Cost vs. Quality Trade-off for High-Density Packaging of Electronic Systems

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH
for the degree of
Doctor of Technical Sciences

presented by
Michael Scheffler
Dipl. Ing. TU Berlin
born June 14th, 1969
citizen of Hanover, Germany

accepted on the recommendation of
Prof. Dr. Gerhard Tröster, examiner
Prof. Dr. Paul Damian Franzon, co-examiner

May 2001
Contents

Abstract III

Zusammenfassung IV

1. Introduction 1
   1.1. High Density Packaging ................................. 1
   1.2. Virtual Prototyping and Trade-off ....................... 5
   1.3. Existing Work on Cost/Quality Trade-offs ............... 8
   1.4. Research Objectives and Novel Contributions .......... 9

2. Calculating Cost and Quality 11
   2.1. Defining Terms ........................................ 11
   2.2. A Process Oriented Cost Model and its Description ...... 13
   2.3. Typical HDP Processes .................................. 13
   2.4. Modeling HDP Processes with Petri Nets ................ 16
   2.5. Specific Firing Rules for a Manufacturing Petri Net ..... 20
       2.5.1. Component ........................................ 21
       2.5.2. Carrier .......................................... 22
       2.5.3. Process Step ..................................... 22
       2.5.4. Assembly Step .................................... 23
       2.5.5. Test Step (2 outputs) ............................. 23
       2.5.6. Test Step (3 outputs) ............................. 24
       2.5.7. Rework or Repair Step ............................. 24
       2.5.8. Sinks ............................................ 24
   2.6. The Electronics Manufacturing Cost Quality Model ...... 25
   2.7. Summary .............................................. 26

3. MOE - Modeling and Analyzing HDP Processes 27
   3.1. The Implementation in MOE .............................. 27
   3.2. Input Parameters for HDP Technologies .................. 31
       3.2.1. Cost Data ......................................... 31
       3.2.2. Yield Data ........................................ 32
       3.2.3. Test Transparency and Fault Coverage ............... 34
       3.2.4. Summary .......................................... 36
   3.3. A Case Study using MOE ................................. 37
   3.4. Performance and Limitations ............................ 41
   3.5. Summary .............................................. 42

4. Search for the Manufacturing Optimum 43
   4.1. Introduction to Multi-Objective Optimization .......... 43
   4.2. Evolutionary Algorithms for Multi-Objective Optimization Problems 48
   4.3. Application of the Evolutionary Algorithm ............... 54
   4.4. Discussion ............................................ 57
   4.5. Summary .............................................. 63

5. Application to Early Design Analysis 64
   5.1. Introduction to the Case Study .......................... 64
   5.2. Optimizing eCard ...................................... 66
   5.3. Results and Discussion ................................ 70
   5.4. Conclusions .......................................... 73
Abstract

In this thesis we present the first approach to concurrently optimize cost and quality aspects of the manufacturing process for electronic products using high-density packaging. Quality in this context defines the ratio of perfectly working units to all delivered units after all phases of a manufacturing test and details how well a product can comply with its specifications.

In order to calculate cost and quality metrics for high-density packaging processes, description models have been developed. The most promising approach found was process-oriented cost modeling, due to its applicability to all existing and future HDP technologies. We further examined different high-density packaging (HDP) processes in order to identify the main “building blocks” for such processes. Since in most cases HDP process interactions are described as flow graphs, we used a Petri net approach, a specific type of graph especially suited for production modeling. This approach has been implemented as graphical description language called MOE. Case studies using MOE have shown that different processes can easily be compared and optimized in terms of cost/quality.

To compare cost/quality results, a ranking scheme had to be found. Selecting the Pareto approach, we pleaded for “true” multi-objective optimization to avoid possible compensation of underperforming objectives and to enable search space exploration. Whereas for a small number of alternatives a “manual” search may be applicable, for most cases an automated method as search support is desirable. We have implemented this method using a multi-objective evolutionary algorithm. The results prove that by using our optimization method significant improvements can be achieved compared to existing search algorithms and solutions. Combining for the first time the Pareto scheme with a cost/quality trade-off, a method has been created to identify turning points where further quality improvements no longer make sense from an economical point of view.

An extension of the proposed methodology to more general problems such as contract manufacturing and system partitioning has been presented. This extension requires the introduction of the additional metrics “product design/manufacturing time” and “production volume”. Whereas volume investigations can be conducted as a parametric sampling of solutions, the production time should instead be included as a third optimization objective. A time calculation model similar to the cost has been outlined. We have demonstrated that these additional metrics can easily be integrated into the existing framework.

Even if no detailed process parameters are available, it is at least possible to determine the Pareto-front type and therefore the inherent optimization potential. The benefits of the proposed cost/quality trade-off methodology have been demonstrated with various examples and a case study on the manufacturing process of a next-generation Smart Card. The case study provided general information on the cost/quality structure of a completely new Smart Card process and evaluated various improvement strategies.
Zusammenfassung


Introduction

This framework describes the development of a trade-off methodology between cost and quality for high-density packaged (HDP) subsystems and modules early in the design cycle. The framework can be divided into two major parts, with the derivation of the cost and quality metrics on the one hand, and the description of the tradeoff methodology on the other.

This chapter introduces the technologies used for HDPs, shows how design concepts have to cope with the special needs of these technologies motivating the need for a cost/quality trade-off, revisits existing work in the cost/quality modeling field, and finally defines the research objectives.

1.1. High Density Packaging

Looking at today's consumer electronics products, such as mobile telephones [1, 2] and personal digital assistants (PDAs) [3], we see that next to improvements in electrical, thermal, and computational performance there is a continuous trend towards miniaturization. This miniaturization is achieved mainly using high-density packaging technologies for the core electronic part of the product, enabling the outer housing to shrink as well. We refer to the core part as the “system”, since it contains the vital functions of an electronic product [4]. A single function of this product is called a “subsystem”. Examples for such functions can be computation, communication, and navigation subsystems [5, 6, 7, 8].

Technologies

Fig. 1.1 summarizes the current number of technology options to implement a miniaturized electronic system. After deciding on the component itself, using either standard devices, application specific ICs (ASICs), or programmable logic, the component has to be connected to other components and/or the outside world. This connection is made by means of packaging and interconnect methods.

However, the fine interconnect structures and pad pitches on ICs cannot be directly transferred to actual printed circuit boards (PCB), see Tables 1.1 and 1.2. Instead, they require a large pitch adaptation area (“fan out”). Therefore, traditional Integrated Circuit (IC) packages, such as Quad Flat Packs (QFPs), are much larger as the IC itself.

HDP on the other hand comprises all technologies that minimize the adaptation area, thus increasing the packaging density of today’s electronics systems. Various HDP technologies are available.

The foremost representatives of HDP technologies are all direct chip attachment technologies where the bare chip is attached to a substrate or a PCB eliminating the intermediate package [11]. Direct chip attachment features interconnection from the chip to the next level using

Wire Bonding (WB), connecting the chip with the next level by means of a small aluminum or gold wire;
1.1. High Density Packaging

Flip Chip (FC), flipping the chip (hence the name) and
- soldering the chip to the next level by means of small bumps, studs, or springs,
- or gluing the chip to the substrate using either isotropic or anisotropic Conductive Adhesives (CA);

Tape Automated Bonding (TAB), contacting the chip to a lead frame that is in turn attached to the next level.

But also

Chip Size/Scale Packages (CSPs), increasing the chip size up to 20%, and other fine pitch packages can be used to miniaturize a system or subsystem. These types of packages do not offer the same degree of density as direct chip attach, but they allow easier handling and (supposedly) higher yield than bare dies, since the package facilitates IC testing. Lately, an especially large number of CSPs packaged on the wafer level have emerged [12]: Super CSP (Fujitsu), UltraCSP (FlipChipTechnologies), ShellCSP (Shellcase), wsCSP (Amkor Anam), µBGA (Tessera), MOST (Form Factor), etc. Whereas all packages fulfill the high-density requirements, they differ in terms of cost and quality, due to their different build-up [12].

Because direct chip attach and fine-pitch components bring the ICs high interconnect density to the next level, this next level also requires high-density interconnect PCBs and substrate types, such as

thinfilm, also referred to as MCM-D(eposited),
ceramics or MCM-C(ermics), and
SBU, or MCM-L(aminite) [13].
1.1. High Density Packaging

While ceramics substrates have still remained a technology on their own, divided into high and low temperature co-fired ceramics (HTCC and LTCC), the distinction between thinfilm and laminate substrates has become somewhat blurred. Recently, laminates have also become used as base material for thinfilm processes, and thinfilm technologies are employed to increase the interconnect density of laminate substrates [14, 15].

An overview on HDP technologies is given in Fig. 1.2, Tables 1.1 and 1.2; detailed information can be found in [16]. From the tables we can see that there is some overlap in the technical properties, so different technologies can be used to fulfill the specifications. Also, next to thermal and electrical properties different processes and technologies come with different cost and yield.

Moreover, additional upcoming technologies driving system miniaturization are the integration of passive components into the substrate (see e.g. [17]), of electro-optical interconnects [18], and of micro-electromechanical (MEMS) devices [19].
1.1. High Density Packaging

Table 1.1: Comparison of interconnect technologies (2000).

<table>
<thead>
<tr>
<th></th>
<th>Wire Bond</th>
<th>Flip Chip</th>
<th>TAB</th>
<th>CSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>min I/O pitch [µm]</td>
<td>50</td>
<td>150</td>
<td>60</td>
<td>400</td>
</tr>
<tr>
<td>min substrate pad pitch [µm]</td>
<td>120</td>
<td>same</td>
<td>200</td>
<td>same</td>
</tr>
<tr>
<td>mounting</td>
<td>serial (2-10 bonds/sec)</td>
<td>parallel</td>
<td>serial/parallel</td>
<td>parallel</td>
</tr>
<tr>
<td>electrical performance L [nH/C [pF]]</td>
<td>1-5/0.2-0.6</td>
<td>0.1-0.2/0.02-0.03</td>
<td>1-3/0.2-0.6</td>
<td>A: 0.3-2.3/ B: 0.05-0.34</td>
</tr>
<tr>
<td>mechanical support/ protection</td>
<td>glob top</td>
<td>underfill</td>
<td>none</td>
<td>underfill?</td>
</tr>
</tbody>
</table>

Table 1.2: Comparison of substrate technologies (2000).

<table>
<thead>
<tr>
<th></th>
<th>SBU</th>
<th>MCM-C/ Ceramics</th>
<th>MCM-D/ Thinfilm</th>
<th>Standard PCB</th>
</tr>
</thead>
<tbody>
<tr>
<td>min design rules [µm]</td>
<td>80/80/120</td>
<td>125/250/200</td>
<td>10/10/30</td>
<td>125/125/650</td>
</tr>
<tr>
<td>no. layers</td>
<td>2*3 + PCB</td>
<td>15-30</td>
<td>2-5</td>
<td>8-30</td>
</tr>
<tr>
<td>dielectrics</td>
<td>Epoxy/BCB</td>
<td>6-10</td>
<td>BCB/PBO</td>
<td>Epoxy</td>
</tr>
<tr>
<td>cr</td>
<td>2.3-4.7</td>
<td>2.7-3.5</td>
<td>4.7</td>
<td></td>
</tr>
<tr>
<td>base material†</td>
<td>FR4</td>
<td>Alumina</td>
<td>high Tg laminate, silicon, metal</td>
<td>FR4</td>
</tr>
<tr>
<td>price (4 layers)</td>
<td>medium</td>
<td>medium</td>
<td>high</td>
<td>lowest</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>≈5 cents/cm²</td>
<td>≈cents/cm²</td>
</tr>
</tbody>
</table>

New Challenges

With this number of technologies, a paradigm shift has recently taken place in electronics manufacturing, from technology-driven to application-driven. So far, many manufacturing companies have offered a specific technology, such as MCM-C including wire bond assembly capabilities or PCB/MCM-L plus soldering [20], accompanied by design capabilities for this technology. Each technology had its own application field:

- thinfilm + flip chip: expensive, high interconnect, used e.g. for mainframe computers (IBM S/360 [21]);
- ceramics: large number of interconnect layers, withstanding high temperatures, offering high dielectrics, used for RF, space, and automotive applications, but also for the early supercomputers;
- laminates: least capable technology in terms of interconnect density and manufacturing precision, but offering lowest cost.

Designers were technology specialists, thriving to improve performance in a certain technology once it had been adopted for a specific product.

But nowadays, the “high walls” between technology choices have become much lower: considering e.g. interconnect density, advanced PCB processes have reached the capabilities formerly left to MCM-D technologies [15]. For digital systems now, the System-on-in-a-package (SOP/SIP), where more than one IC plus other components are integrated into a single package, competes with System-on-a-chip (SOC), where the same functional blocks are integrated on a single piece of semiconductor. When both implementations can deliver the technical performance, “secondary” criteria, such as cost, time-to-market, and quality become predominant [22].
Moreover, with new applications such as PDAs, GameBoys, communication devices as well as computing applications to be integrated in clothing, miniaturization has become a selling factor, and packaging can be the key to system performance, e.g. for RF applications. Both designers and manufacturing companies have to consider new technologies, processes, and design concepts to accomplish these goals, as the customer “does not care for the technology”.

Finally, a recent study claims that small and medium electronics manufacturers will have to broaden their portfolio to include product responsibility in addition to only contract fabrication [23]. For major players this strategy allows for growth also when most of the market is already captive [24].

Summarizing this section we find that

• the number of technology options to implement high-density systems has increased over the last ten years, not only due to the advent of new solutions as e.g. CSPs, but also due to the diminishing capability gap, such as the routing density of SBU advancing to those of ceramics and thinfilm,

• manufacturers are experiencing a paradigm shift from technology sellers to product designers, and in this position they have to consider (at least to some degree) all of the technological options to make sure the most suited technology has been chosen to realize a specific system.

The next section shows how this paradigm shift has been addressed.

## 1.2. Virtual Prototyping and Trade-off

Moving from fabrication of PCBs to fabrication of HDP substrates and from ordinary soldering to direct chip attachment, the complexity grows due to the number of higher order precision processes required. Examples for this increased complexity are the alignment of two PCB layers with 350μm via land compared to mask alignment for thinfilm substrates with 50μm via land, or the placement of 0805 SMDs vs. 0201 SMDs. Moreover, rework and repair of HDP systems is more complicated than for PCBs using standard SMD technology (see [21] for repair of thinfilm substrates). As a consequence, HDP fabrication takes more time and is more expensive than standard PCB/SMD or even through-hole electronic manufacturing. Next to that, the tolerance band for design features (e.g. noise margins) has narrowed due to higher signal frequencies, closer interconnect lines, and the choice of the least expensive technology delivering the required performance. So designers cannot rely upon a design that “has been found to work” instead of analyzing all working conditions. This increase in complexity and cost of manufacturing HDP systems and the decrease in tolerance makes it necessary to switch from a “correct-by-verification” approach to a “correct-by-design” solution [25].

### Virtual Prototyping - Exploration of the Design Space

Thus, already early in the design cycle possible implementations have to be analyzed carefully to select the most promising ones. This problem has been addressed using the concept of “virtual prototyping” [25]:

> “..., which allows designers to define and test system attributes prior to large design or fabrication investments.”

An overview of early design issues to be covered for the analysis of a virtual prototype and their interactions is given in Fig. 1.3 [26].

So far, stand-alone tools have achieved good coverage for most of the problems in Fig. 1.3, but the number of design options spanning the design space makes an analysis of more than one solution cumbersome. Still, it is necessary to explore several design options in parallel in order to find within reasonable time a solution satisfying all design goals. But in general there is no “optimum solution” giving the best performance for all design goals because the features of Fig. 1.3 are not independent from each other. For example, consider that when moving a component on a PCB thermal management might improve, but the system’s size may enlarge. Also, different wafer-level CSP technologies might come with higher reliability, but also higher cost and different form factors.
1.2. Virtual Prototyping and Trade-off

Making Compromises - Concurrent Design

As a result, designers have to make a compromise between concurrent features. They have to trade off:

"Trade-off analysis is the process of comparing performance gains in one region of the design space [covering the issues above] with the associated performance losses in another." [10]

Trade-off is "a balancing of factors all of which are not attainable at the same time; a giving up of one thing in return for another". [4]

Next to the development of functionality, test, manufacturing, and quality have gained increasing attention as dedicated design issues. Choices made early in the design cycle drive (either willingly or not) such aspects as test and manufacturing issues responsible for the majority of the production cost. This is referred to as the 80-20 rule (Fig. 1.4 ([25] after [28] and [27] after [29]).

Early consideration of the issues above is referred to as DfX “Design for X”, where “X” can be replaced by T(est), M(anufacturing), C(ost), and Q(uality) (see e.g. [30, 31]).

The Cost versus Quality Trade-off in the Design Phase

An area that has been rarely addressed is the cost vs. quality\(^1\) trade-off, although several publications have stressed the importance of a quality concept [32, 33]. Most of the concepts, as we will see later, focus only on how to find the most cost effective test concept for a given quality.

Some rules-of-thumb wisdom exists. Fig 1.5(b) gives the qualitative relation of cost and quality, where at a certain (problem specific) level the relation changes its behavior from proportional to exponential, meaning that even large investments result only in very small quality improvement. Cost vs. quality is strongly related to the testability issue. Fig. 1.5(a) illustrates the “rule of ten” stating that test plus repair cost multiplies by ten from level to level. Ambler and Bennett [34] have outlined that this rule is often cited and is perceived to be very mythical, but conceded that there is clearly a nonlinear behavior (Fig. 1.5(b)) in cost vs. quality.

On the other hand there has been significant work done investigating the relationship between test and quality, e.g. [35, 36, 37]. Common sense is that maximum testing and yield results in the highest quality, which is highly desirable.

\(^1\)For a definition of terms see section 2.1.
1.2. Virtual Prototyping and Trade-off

Figure 1.4: 20-80 rule: up to 80% of the entire cost are committed within the first 20% of the design cycle [25, 27].

"One of the most critical goals in business is to get the quality level as close as possible to 100%", [37]

but also with highest cost being less appreciated. Das et al. state that

"a perfect fault coverage is an illusive goal" and there is a "very high incremental cost of either determining a remaining fault to be redundant or finding a test for it." [38].

On the other hand minimum efforts in yield and testing give the lowest cost, but also the lowest quality, being not desirable. Thus, the rules of thumb do not help in answering a specific question on a cost vs. quality trade-off like in this example:

Example 1.1

Figure 1.5: Rules of Thumb [34]: (a) the "Rule of Ten" points out that the later a fault gets caught the more expensive test and repair are; (b) there exists an exponential relationship between quality and test cost for high quality values.
For an electronic subsystem responsible for data readout in high-energy experimental physics [39], the question was whether the existing test concept would fulfill the customer quality specifications and how other concepts (either IddQ, or enhanced final test) would perform. These additional test efforts come with a considerable cost overhead (and in case of IddQ also with a design time delay).

Looking back to Fig 1.5(b), we need to find the turning point where pushing forward quality does not make economically sense.

In the following section we will see how the cost-quality problem has been addressed by existing approaches.

1.3. Existing Work on Cost/Quality Trade-offs

Three different methodology families are involved in the cost-quality problem: virtual prototyping, cost modeling, and quality engineering.

Virtual Prototyping

Various tools support virtual prototyping, i.e. they provide analysis methods to assess (all or part of) the issues in Fig. 1.3. These tools are also called early design analysis (EDA) tools. Among the more academic ones are IMPACT (Georgia Tech) [40], AuDiT (Cornell University) [41], commercial systems comprise HiTEA (MCC) and its successor SavanSys (Savantage Inc., now NuThera Systems), and EDANavigator (Xynetix, now Avant! Corp.). The academic tools consist of less detailed simplified analytical models exemplifying trends; the commercial tools supply a two-step method. Their simple analysis is based on models similar to academic approaches using public or proprietary models as first estimations; for thorough investigations they use refined models and/or interface to commercial tools (e.g. Flowtherm for thermal problems, Ansoft HFSS for 3D Finite Element Modeling (FEM)). None of these virtual prototyping tools addresses the cost vs. quality trade-off problem.

Savansys offers a very detailed cost analysis, but quality is not an analysis feature. Moreover, no dedicated trade-off method is available. Only a “what-if” analysis is possible, meaning “trial-and-error” to find a design balancing the interdependency of the key design issues.

EDANavigator does not incorporate cost modeling but at least offers a weighting of the other analysis results to generate an overall design score while monitoring out-of-specification events.

In addition to that, the tools above focus strictly on either MCMs or ICs and do not permit a combination of both (SOC/SOP) or an exploitation of new aspects such as Integral Passives, or integration of MEMS devices. The simple models they provide are not meant to be adapted.

Cost Modeling

Next to the prototyping tools, we also review the stand-alone tools and approaches. Numerous proprietary cost modeling approaches exist: commercial ones such as SimWitch (MTBA) [42] or Envision (Dow Chemical, commercialized by TechSearch Intl.) [43], or the methods published by Dislis (Known Good Die (KGD) Selection, [44]), Alexander (Surface Mound PCB Design Evaluation, [45]), Werkmann (Cost Effectiveness of Smart Substrates, [46]), and others.

Among these approaches only the one from Dislis covers the cost and quality aspects, but without any trade-off. Moreover, it is tailored to the KGD problem only, and adapting it to other types of processes would require rewriting the entire spreadsheet implementation.

Most of the cost modeling approaches above are realized as lengthy spreadsheets [47, 48, 49] or use a specialized user interface rendering them useless for other than the targeted types of problems. The level of detail and abstraction also spans from ten formulae describing the entire process [45] to seven pages of spreadsheets [47]. So none of the approaches covers the entire chain from early estimation to detailed analysis, which would be necessary to keep consistent models throughout the entire design process while refining data.

---

2 Passive components such as resistors, capacitors, and inductors integrated into a substrate or board.
Quality Engineering

On the other hand, quality engineering (also known as Total Quality Management TQM) also incorporates cost considerations (see e.g. [50, 51, 52]). The total quality cost is considered to be the sum to all appraisal/evaluation costs, the failure costs, and the prevention costs [50, 51]. TQM works on the basis of a predefined quality goal (in our case a specific defect level, see section 2.1 “Defining Terms”), and defines cost penalties when missing this goal, e.g. using the Taguchi model of a parabolic penalty function for a continuous target [53]. Considering discrete events such as “defect/no defect”, a cost penalty for field returns (defective units returned after delivery) is often used [52]. Optimization in any case was done by “trial-and-error”.

The Taguchi model does not help define or trade off a defect level early in the design cycle; it only gives a measure how a predefined quality level could be reached. Moreover, the Taguchi method requires statistical data that is not available early in the design cycle.

The second concept of field return penalty has several drawbacks. First, it is based on a 100% field return rate. Recent customer surveys have revealed that a significant number of customers, rather than complaining and returning the product, instead change the supplier without notice, so field return rates are grossly underestimated. Second, considering customer satisfaction, quality (and its fulfillment) have a very high value [23], and it is very hard to win back a dissatisfied customer. Therefore, translating such dissatisfaction into money can be misleading. This fact is a known drawback of single-objective optimizations [54], since the different objectives have to be forced into one single metrics.

Therefore we consider it to be better to address cost and quality separately during optimization in order to ensure that both goals are sufficiently met.

Summary

We have seen that

- the cost vs. quality trade-off in the early design cycle is not covered by existing solutions,
- even existing cost modeling solutions do not cover all existing HDP technologies, and they are not meant to be adapted to future HDP technologies such as Integral Passives, electro-optical interconnects, and MEMS,
- quality engineering is more dedicated to meeting a predefined quality goal at a minimum cost instead of helping to define that goal,
- although concurrent analysis is available, trade-off is often not more than changing parameters until all specifications are met, i.e. “trial-and-error”.

Identifying these shortcomings, the next section defines the research objectives to address.

1.4. Research Objectives and Novel Contributions

In order to enable a cost vs. quality trade-off for HDP systems, the following tasks have to be accomplished:

1. to develop a model that incorporates all relevant aspects of manufacturing an existing HDP product in order to produce the metrics cost and quality;
2. to make this model scalable to all types of HDP technologies and thus to overcome the limitations of the existing approaches;
3. to automate the cost/quality trade-off and to develop a method that gives decision support in order to identify for the first time quantitatively the turning point shown in Fig. 1.5(b) claimed to exist for a cost-quality trade-off;
4. to show a new approach how the trade-off concept can be used to cope with partitioning concepts (e.g. SOC/SOP) and manufacturing aspects (contract manufacturing).
The remainder of this work is organized as follows:

In Chapter 2 we develop the basic models to calculate cost and quality for HDP systems (objective 1), which in Chapter 3 are applied to HDP production processes and case studies using the Modular Optimization Environment MOE (objective 2). Chapter 4 introduces a method to compare different cost/quality combinations. We demonstrate the need for an automated search methodology finding trade-off points (objective 3). In Chapter 5 we exemplify the benefits of our concept with a case study on the manufacturing process of a Next Generation Smart Card. A generalization of the proposed methodology is shown in Chapter 6 (objective 4). The final chapter is dedicated to the summary of this thesis.
Calculating Cost and Quality

In this chapter we explain the fundamentals of cost and quality modeling, beginning with the explanation of terms used throughout this framework. Then, we analyze existing HDP manufacturing processes and propose process oriented modeling to map the different technologies. Using definitions and descriptions from the graph theory to model processes, we introduce a colored stochastic Petri net modeling approach.

2.1. Defining Terms

Before we start with the development of the models, it is appropriate to define some terms used throughout this framework.

**Unit** (or packet as it is used later in the information processing context) is a single entity of the system or subsystem to be produced.

**Cost or total cost** $c_{total}$ is the sum of all costs associated with the acquisition, production, and distribution of an electronic system. The product’s cost is (hopefully) lower than the specific price the product is sold for. Elements to be included in the total cost are [10]:

- non-recurring design and development costs,
- capital costs, tooling costs, learning curves,
- recurring fabrication and assembly costs of components and packaging,
- repair and rework costs, testing costs,
- support and maintenance cost.

All these contributors can be sorted into two categories: direct and indirect cost.

**Direct cost** $DC$ is the sum of all costs directly associated with a single unit of a product; the sum of all component costs $c_{comp}$ plus the sum of all process costs $c_{proc}$ directly involved in its production. For simplicity’s sake direct cost in this framework will be treated as production quantity independent, although in general component cost changes with the order quantity. As not all units undergo exactly the same processes (e.g. some units require rework), the direct cost is an average number.

**Indirect cost** is the sum of all other costs. This sum has to be broken down into the number of produced units; in explicit terms, the more units produced, the lower the indirect cost per unit is. We divide the indirect cost into non-recurring expenditure and yield loss.
2.1. Defining Terms

**Non-recurring expenditure** \(NRE\) incorporates all capital, tooling, and design costs, plus overhead. The calculation of capital costs or the depreciation amount itself is tax-law dependent and therefore country and company specific. Various procedures exist, such as straight line, exponential progressive or regressive, with different depreciation periods [55] in order to determine the annual NRE.

**Yield loss** \(YL\) or reject cost is the sum of all sorted out or scrapped units throughout the entire production. The yield loss is driven by the yield of the components \(y_{\text{comp}}\) and the processes \(y_{\text{proc}}\) on the one hand and the test transparency on the other.

The division into direct and indirect costs is used widely within several cost modeling approaches, but the exact assignment of the cost elements from the list above differs [27, 45, 56, 57]. We used the division according to Fig. 2.1 into \(DC\), \(YL\), and \(NRE\) to better focus on the separate cost optimization potentials.

![Figure 2.1: Process information required to calculate the three cost categories direct cost, NRE, and reject cost.](image)

**Yield** \(y\) is defined as the ratio of the number of usable items after the completion of production processes to the number of potentially usable items in the beginning of a production \(y_{\text{proc}}, y_{\text{comp}}\) [58]. We refine it to the ratio of defective items/units BEFORE testing to the number of potentially usable items, \(y\) values are \([0, 1]\), also described as being between 0% and 100%.

**Defect level** \(DL\) is defined as the percentage of faulty circuits passing all phases of a manufacturing test [58], i.e. the yield AFTER test, sometimes also referred to as the Field Reject Rate or Process Defect Level. The defect level represents the proportion of a product which may fail because of extrinsic failure (or infant mortality) and is a function of both yield and test transparency. \(DL\) values are also \([0, 1]\), often in the range of parts per million (ppm, \(10^{-6}\)).

**Quality level** \(QL\) or simply quality is \((1 - DL)\), and therefore the ratio of perfectly working units to all delivered units after all phases of a manufacturing test. Quality in the context of TQM also includes the aspects of reliability, maintainability, and availability [52], or the tolerances within a certain parameter can lie. Our definition assigns the defect level as “first order quality” (what is the average ratio with which the product can comply with its specifications). “Second order quality” deals with the tolerances (what is the deviation from this average, leading to process control), “third order quality” includes time (how the average compliance ratio develops over time, implying reliability).

**Test transparency** \(TT\) is the probability \([0, 1]\) of a unit escaping a test. \(TT\) is related to fault coverage and fault clustering [59].

**Fault coverage** \(fc\) is the ratio of the number of detectable faults and the possible number of faults assumed on a fault list, again ranging between 0% and 100%.

In the subsequent sections, these definitions are translated into mathematical formulas.
2.2. A Process Oriented Cost Model and its Description

Cost estimation has long been used in every company, on different hierarchy levels and with different levels of complexity. Table 2.1 lists a number of various scopes and methods where cost estimation is applied.

- Traditional cost accounting is labor oriented, converting processes to labor hours required. Here, indirect cost is associated as burden percentage per labor hour, and the approaches work only when labor is the main cost factor. But in electronics manufacturing most of the cost is associated with the equipment, leading to

- the cost-of-ownership model (COO) for production equipment [60]. But this model is not sensible for material driven processes such as PCB fabrication.

- Activity based cost modeling [27] works on the number of high-level actions as e.g. design processes, purchase orders, and production change overs. The more actions required, the more costly a product will be. Parametric approaches do the same for products, assuming that the cost of a product is only dependent from the number of parts, neglecting other influences such as new technologies or major product changes.

- Technical cost modeling is a process-based model which simulates the manufacturing operations in order to estimate cost [27].

<table>
<thead>
<tr>
<th>Hierarchy Level</th>
<th>Scope</th>
<th>Key Attribute</th>
<th>Popular Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corporate</td>
<td>All company activities (multiple products)</td>
<td>Accurate allocation of overheads to specific products</td>
<td>• Activity based cost analysis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Traditional cost accounting</td>
</tr>
<tr>
<td>Program</td>
<td>Single product</td>
<td>Treats entire product life cycle including HW and SW</td>
<td>• Parametric</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Traditional cost accounting</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>All or part of a single product</td>
<td>HW only</td>
<td>• Cost of Ownership</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Technical Cost Modeling</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Parametric</td>
</tr>
<tr>
<td>Process</td>
<td>Single processing activity</td>
<td>HW only</td>
<td>• Analytical models</td>
</tr>
</tbody>
</table>

Of all the modeling methods presented in Table 2.1, Technical Cost Modeling is the most appropriate for modeling HDP systems. The process-centered approach guarantees that existing HDP technologies can be modeled and new technologies, design and manufacturing concepts can be included when broken down to their respective processes.

2.3. Typical HDP Processes

Before trying to find an appropriate description to model HDP processes, it is opportune to investigate such processes to find common denominators.

From our general description of the HDP technology in Section 1.1, we take three process examples. The first one describes a part of a PCB fabrication process (Fig. 2.2), typical for any kind of substrate processing, the second one details the wire bond - SMD mixed assembly of an RF system (Fig. 2.3), representing the interconnection technologies. The third one is a part of a KGD qualification process (Fig. 2.4). We will not describe the technological issues in detail, we only want to illustrate several process typical properties.

From the example processes we see that:

1. Processes are preferably described as sequential material flows, having (at least) one source and (at least) one sink. Material units flow in the direction indicated by the arrows. The source of a material flow are components.
2.3. Typical HDP Processes

Figure 2.2: Example for HDP processes I. Part of a PCB fabrication process [16].
2.3. Typical HDP Processes

2. While progressing in the flow, processes operate on the components, changing their properties.

3. In a material flow, several incoming components can be combined to a single outgoing component, e.g. see Fig. 2.2 in the upper left corner where several single panels are combined to a stacked one. Here multiple flows are joined. This action requires some sort of synchronization and is called assembly. Only when a specified number of panels have arrived at the lamination machine, the machine starts processing.

4. In some cases (see Fig. 2.4) the flow can switch to only one of the outputs at a time. Following a stochastic distribution for certain events ("error present, error detected") the flow takes one of the branches. In analogy to Fig. 2.4 we call this test.

5. Also in Fig. 2.4, feedback loops can be found, required for rework and repair. Examples for such activities can be repair of open interconnect lines on an MCM-D substrate [21] or replacement of...
malfunctioning components. Once faulty units have been repaired, they have to enter the “normal” flow again. Again, this action requires synchronization.

Several approaches exist to capture general process descriptions, one of them is e.g. the IDEF3 format [63]:

An IDEF3 Process Flow Description captures a description of a process and the network of relations that exists between processes . . .

Whereas IDEF3 itself provides only behavioral relations, in our context we strive for a mathematical model as well. Thus, as a descriptive means for process modeling we use Petri nets, which have been applied to model fabrication processes and data flows for some time [64, 65].

Since Petri nets are a special subset of graphs, the next section introduces some basics of the graph theory and Petri nets.

### 2.4. Modeling HDP Processes with Petri Nets

As previously said, Petri nets are a subset of graphs. Graphs serve as mathematical models to analyze various types of problems, beginning with the ancient famous “Königsberg bridges”, the “Four Color Conjecture” of map coloring, and Kirchhoff’s electrical networks to linear programming, operations research, and scheduling [64, 65, 66, 67, 68, 69]. Below follow some basic definitions of graph theory. Next to these basics, we will extend the definitions to include special properties required to model fabrication processes.

**Definition 2.1**

**Graph.** A graph \( G(V, E) \) consists of a set \( V \) of elements called **nodes** together with a set \( E \) of unordered pairs of the form \((i, j)\) or \((j, i)\), called the **edges** of \( G \); the nodes \( i \) and \( j \) are called the **endpoints** of \((i, j)\) [67]. Other names commonly used for a node are vertex, point, and junction, for an edge line, branch, and arc.

![Graphical representation of edge and node](image)

**Figure 2.5:** Graphical representation of edge and node (left), sample graph (right).

The graphical notation for edges and nodes is shown in Fig. 2.5, together with a very simple graph. Since processes and data flows in general have a specific sequential direction, we need to extend Definition 2.1.

**Definition 2.2**

**Directed Graph.** A directed graph \( G_d(V, E) \) or short digraph consists of a set \( V \) of elements called **nodes** together with a set \( E \) of ordered pairs of the form \((i, j)\), called the **edges** of \( G \); node \( i \) is called the **initial node** and \( j \) is called the **terminal node** [67].

In order to describe a way through a graph, we define a route.

**Definition 2.3**

**Route.** A route \( r(i_1, i_k) \) or sequence in a directed graph \( G_d \) is an edge sequence of length \( k - 1 \) in which the edges along the edge sequence are in the form of

\[
(i_1, i_2), (i_2, i_3), \ldots, (i_{k-1}, i_k),
\]

\( k \geq 2 \), where \( i_1 \neq i_k \), i.e. starting and terminal node are not the same.
Graphs can describe fabrication processes in various levels of detail or granularity (cf. the lamination process in Fig. 2.2 with the test/repair process in Fig. 2.4). Still, as long as two different representations model the same process they are said to be homeomorphic.

**Definition 2.4**

**Homeomorphism.** Two graphs are said to be **homeomorphic** if both can be obtained from the same graph by a sequence of subdivisions of the edges.

This means that using subdivisions and their counterpart reductions (combining multiple edges into a single one), we can transform graphs into each other, increasing or decreasing the level of detail.

All processes have (at least) one starting point and (at least) one terminal point, which we have called source and sink in the previous section. Before defining these two items, we have to introduce the degree of nodes.

**Definition 2.5**

**Outcoming and incoming degrees.** For a directed graph \( G_d \) the number \( d^+(i) \) of edges of \( G_d \) having node \( i \) as their initial node is called the **outgoing degree** of node \( i \), and the number \( d^-(i) \) of edges of \( G_d \) having node \( i \) as their terminal node is called the **incoming degree** of node \( i \).

**Definition 2.6**

**Sources and sinks.** A node \( i \) in \( G_d \) having an incoming degree \( d^-(i) = 0 \) is called a **sink**, a node \( i \) in \( G_d \) having an outgoing degree \( d^+(i) = 0 \) is called a **source**.

**Extension 2.6:** For a production process, two types of sinks are present, "Units to be shipped", and "Units to be scrapped". Components or material are the sources of a production process.

![Example for a digraph](image)

**Figure 2.6:** Example for a digraph, node \( V_1 \) being a source, nodes \( V_6 \) and \( V_{10} \) being sinks.

While graphs as shown in Fig. 2.6 are used for combinatorial problems such as partitioning, map coloring, etc., for synchronization problems and modeling of manufacturing systems, a special type of graph, called Petri net [70, 71], is widely employed. Extending our existing graph definition, we can say that Petri nets contain two different types of nodes: places \( V_P \) or \( P \) and transitions \( V_T \) or \( T \) [65], connected by directed edges. Places and transitions are to be ordered alternatingly (see Fig. 2.7).

**Definition 2.7**

**Petri net.** A marked Petri net \( Z = (P, T, I, O, m) \) is a five tuple where

1. \( P \) is a finite set of places,
2. \( T \) is a finite set of transitions,
3. \( I(T_x) \) is an input function defining the set of directed edges from \( P_x \) to \( T_x \),
4. \( O(T_x) \) is an output function defining the set of directed edges from \( T_x \) to \( P_{x+1} \),
5. \( m \) is a marking vector where the \( k^{th} \) component represents the number \( i \) of tokens in the \( k^{th} \) place. An initial marking is denoted by \( m_0 \).
2.4. Modeling HDP Processes with Petri Nets

**Extension 2.7:** In analogy to Extension 2.6, we define two types of sinks, $P_{\text{ship}}$ and $P_{\text{scrap}}$. Whereas $P_{\text{ship}}$ is unique, multiple instances of $P_{\text{scrap}}$ can be present. Thus, there are two generic routes through a production graph, which we will call $r_{\text{reject}}$ and $r_{\text{ship}}$, ending either in $P_{\text{ship}}$ or $P_{\text{scrap}}$. The route a unit takes during production is influenced by yield and test factors, and we detail this later. A token represents such a unit in the manufacturing flow.

![Petri Net Diagram](image)

**Figure 2.7:** Example for a Petri net, place $P_3$ marked with a token.

Whereas the four-tuple $(P, T, I, O)$ defines the structure, the introduction of tokens marking places and their flow through transitions enable us to describe the discrete-event behavior of the modeled system. The execution rules of a Petri net are called firing rules.

**Definition 2.8**

**General firing rules.** A transition $T_x$ is enabled to fire iff

- all input places $P_{\text{Input}}(T_x)$ of $T_x$ are marked with a sufficient number of tokens defined in $I(T_x)$,
- and the number of tokens defined in $O(T_x)$ can be added to the output places $P_{\text{Output}}(T_x)$ of $T_x$.

Using Definition 2.8, assembly of subunits, e.g. single panels, to larger units, e.g. multi-layer panels, can easily be accomplished. Definition 2.8 also calls for the definition of a place’s capacity.

**Definition 2.9**

**Place capacity.** The capacity $C(P_x)$ of a place $P_x$ defines the maximum number of tokens allowed to reside in place $P_x$. The actual number of tokens in place $P_x$ is $N(P_x)$.

**Extension 2.9:** Unless otherwise noted, we define $C(P_x) = 1$.

Using this notation, we can model almost all transitions of the processes described in Figs. 2.2 to 2.4. The remaining element is the test in Fig. 2.4 that routes a unit either to branch A or branch B, depending on the outcome of a test (either good or faulty). This transition is called a conditional transition [70].

**Definition 2.10**

**Transitions.** In total, there are five possible types of transition combinations in a Petri net [70] (see Fig. 2.8), namely

- automatic or simple transitions (single input place, single output place, single transition): the token is passed from the input place to the output place;
- parallel transition (single input place, single transition, multiple output places): a single token in the input place is multiplied by the number of output places;
- synchronic transitions (multiple input places, single output place, single transition): firing is granted only in case ALL input places contain a sufficient number of tokens; all tokens in the input places are processed to a single token in the output place;
2.4. Modeling HDP Processes with Petri Nets

- a conditional transition (single input place, multiple output places, multiple transitions): the token is passed to ONE of the output places, depending on the fulfillment of the firing condition;

- unconditional or concurrent transitions (multiple input places, single output place, multiple transitions): only ONE of the transitions is allowed to fire.

All other transitions can be constructed using these basic transitions.

![Basic transitions in a Petri net.](image)

In our case of testing the transition decision is related first to the yield of the unit (error present or not) and second to the test transparency (is the error detected or not?), and both variables, yield and test transparency, are of stochastic nature. A graph where the transitions are dependent on stochastic events is called a Markov chain [72]. Simple Markov chains can be modeled using generalized stochastic Petri nets (GSPN), where transitions can operate conditionally. Thus, for the test case we have to implement a specific firing rule that is based on a conditional expression [71].

MOSES [73] e.g., a simulation software developed by the Computer Engineering and Networks Lab of ETH Zurich, implements this conditional expression supplying a global variable that can be observed to make the routing decision. For a production model, the MOSES construct would be rather complicated, since the global variable has to be connected with the actual error status of a component because this error status determines which specific transition (e.g. “unit error-free”) has to be fired. Therefore, a better approach is to “personalize” the so-far indistinguishable tokens with manufacturing information. This information acts like a tag or bar code present in real production supplying details about fabrication history and actual status of a specific unit. The tag makes the error information available not only to select the firing transition, but also to compute the final quality of a unit, satisfying both requirements with the same approach.

By distinguishing among tokens in an ordinary Petri net and modifying the Petri net executing rules, one creates a “colored” Petri net. The “color” is a synonym for a set of attributes to be assigned to a token which may change as the token moves through the net [71]. The information we are interested in is the cost associated with a specific unit and the number of errors or faults present in this unit.

**Definition 2.11**

**Colored token.** A colored token \( \omega \) is defined as a set of variables (“properties”) which can be changed by the input and output functions \( I(T_x) \) and \( O(T_x) \) of a transition \( T_x \).

**Extension 2.11:** In a HDP production flow we define

\[
\omega = \{ \text{cost, error}_1, \ldots, \text{error}_n, \text{max, rework, counter} \}; \quad (2.2)
\]

\( \text{cost} \in \mathbb{R}, \text{all others} \in \mathbb{N} \).

This approach of colored tokens inherently ensures data consistency all over the net and thus the process model.

Definition (2.11) already foresees several error counters to enable only testing of partial aspects of a unit. X-ray testing after micro BGA soldering e.g. only checks for interconnect faults and cannot identify a non-working component.
2.5. Specific Firing Rules for a Manufacturing Petri Net

The number of errors can be extracted from the yield probability distribution for a process or a component, using Monte Carlo sampling.\(^1\) For our type of applications (discrete events: "error/no error") we use the binomial distribution \([75, 76]\), which gives the probability of exactly \(x\) failures in \(n\) trials with the probability of success \(p\), being defined as

\[
\text{Probability}(X = x) = \binom{n}{x} p^x (1-p)^{n-x}.
\] (2.3)

Thus, following Eq. 2.3 and setting \(n\) to one, \(x\) to \(e \in \{0, 1\}\) and \(p\) to the yield \(y\), we generate random numbers \(e\) for \(y\) according to a given probability distribution, representing "error-free" and "erroneous" events. Sampling over a sufficient number of runs \(N_{\text{eff}}\) (which is problem-dependent), the estimate \(\hat{y}\) mimics the desired value for \(y\) (see below).

\[
\hat{y} = \frac{\sum_{e=0}^{N_{\text{event}}} \text{events with } e = 0}{N_{\text{event}}} N_{\text{event}} > N_{\text{eff}}
\] (2.4)

\[
\hat{y} \approx \frac{\sum_{e=0}^{N_{\text{event}}} \text{events with } e = 1}{N_{\text{event}}} N_{\text{event}} > N_{\text{eff}}
\] (2.5)

The final question to be answered in this section is the Petri net analysis method. We are not primarily interested in scheduling and planning, because the process flows as shown in Figs. 2.2 to 2.4 are very simple granting in most cases the properties above, but we want to obtain numerical information stored in the tokens.

Basically, one can distinguish between analytical and numerical methods. In general, for Petri nets analytical methods are available from general graph theory, providing information on reachability, liveness, and deadlocks. We refrain from defining these terms as they are commonly used as in their colloquial sense; for further readings see \([70, 71]\). Discrete event simulation (sometimes referred to as "token game") is a numerical type of method. The idea is simple: just choosing an initial marking, and then using the execution algorithm, i.e. the transition rules, to run the net. Simulation is reported to be time and computation intensive, but for the processes we intend to model the effort stays within reasonable bounds. We will investigate later on this point. First of all, we define the firing rules to operate the Petri net.

2.5. Specific Firing Rules for a Manufacturing Petri Net

In order to model and to calculate manufacturing processes using the Petri net approach, we have to define specific firing rules. Before doing so, we have to investigate how many different rules we require. Fig. 2.9 shows a Petri net for a material flow (cf. \([70]\)). The transitions on the left hand side continue to fire while their output places are empty. The tokens are processed to the sinks on the right hand side, and the net is alive unless the number of tokens in the upper right corner sink has reached the specified number of \(N\). The sink below has infinite capacity.

We now compare the transitions with the typical processing actions undertaken in a manufacturing process as described in Section 2.3. Fig. 2.10 shows the Petri net structure of Fig. 2.9 with transitions renamed to adequate process steps in a manufacturing process, with a summary in Table 2.2.

A parallel transition is not present in this model. This transition is only required when throughput calculations have to be made, modeling two or more identical subunits of a machine. Another possible application area would be disassembly processes which are not within the research focus of this framework. Also, wafer dicing processes are better modeled using a modified simple transition, since the diced component undergoes the same subsequent process chain afterwards. For a wafer dicing process e.g., fanning out to 1024 parallel identical process chains modeling all dies does not contribute to the abstraction of the manufacturing process.

The following definition of the firing rules for each of the transitions is divided into

- firing, i.e. transportation of the token itself (when/how), and

\(^1\)The name Monte Carlo was applied to a class of mathematical methods first used by scientists working on the development of nuclear weapons in Los Alamos in the 1940s. The essence of the method is the invention of games of chance whose behavior and outcome can be used to study interesting phenomena \([74]\). Nowadays, Monte Carlo methods are used to evaluate finite-dimensional integrals, in statistical physics, and for sampling of integral equations.
2.5 Specific Firing Rules for a Manufacturing Petri Net

2.5.1 Component

A component acts as the source of a manufacturing flow. This item is therefore by the component cost $c_{\text{comp}}$ and the component yield $y_{\text{comp}}$. Indirect costs are summed under $NRE_{\text{comp}}$.

Firing: if the capacity of the output place is not exceeded, fire a single token $\omega$ to the output place:

$$N'(P_{Output,T_x}) = N(P_{Output,T_x}) + 1$$

if $N(P_{Output,T_x}) < C(P_{Output,T_x})$.  \hfill (2.6)

Operation on token: generate new token $\omega$ having the following properties:

$$\omega_{\text{cost}} := c_{\text{comp}},$$

$$\omega_{\text{error}} := \text{RND}(1, y_{\text{comp}}),$$

$$\omega_{\text{rework counter}} := 0.$$  \hfill (2.7) \hfill (2.8) \hfill (2.9)

RND(n,p) calculates a random fault tag $\in \mathbb{N}$ according to the binomial distribution given in Eq. 2.3. When $n$ is 1 (single drawing), the fault tag is $\in \{0, 1\}$. $p$ is specified by the yield of the process or the component. With a 90% yield, from every 100 produced units 90 are fault-free, and ten units contain an error.
Table 2.2: Assignment of Petri net transitions to basic manufacturing flow items.

<table>
<thead>
<tr>
<th>Petri Symbol</th>
<th>Transition type</th>
<th>Process element</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Petri Symbol]</td>
<td>fire only</td>
<td>Component, Carrier</td>
<td>A fire-only transition can be interpreted as a component entering the manufacturing flow.</td>
</tr>
<tr>
<td>![Petri Symbol]</td>
<td>simple</td>
<td>Process, Rework</td>
<td>A simple transition is the model for a process where a unit is processed from the input of a machine to the output. The process operates on the unit, i.e. on the information variables in the token.</td>
</tr>
<tr>
<td>![Petri Symbol]</td>
<td>concurrent</td>
<td>feed back Process</td>
<td>In a concurrency situation, two units compete on the utilization of a single machine, in our case with a feedback loop present. The transition has to guarantee that no deadlock and no endless looping occurs.</td>
</tr>
<tr>
<td>![Petri Symbol]</td>
<td>conditional</td>
<td>Test</td>
<td>A conditional transition selects a transition to fire, mimicking a test station selecting the next process, depending on the actual device under test.</td>
</tr>
<tr>
<td>![Petri Symbol]</td>
<td>synchronous</td>
<td>Assembly</td>
<td>Like an assembly station, this transition awaits the specified number of components (= tokens) to be sent out together. Only if the specified number of components is available, the process is completed.</td>
</tr>
</tbody>
</table>

2.5.2. Carrier

The carrier is a special type of component, presenting an MCM or PCB carrier substrate in an HDP manufacturing flow. The carrier cost can either be specified explicitly or be calculated as a product of area and cost per area. A carrier is described either by the total carrier cost \( c_{\text{carrier}} \) or the carrier cost per area \( c_{\text{carrier,area}} \) and the carrier area \( A_{\text{carrier}} \). This implementation provides the opportunity to explore different manufacturing options demanding different carrier sizes. Additionally, the carrier yield \( y_{\text{carrier}} \) and possible setup costs \( NRE_{\text{carrier}} \) have to be specified. The carrier also acts as a flow source item.

**Firing:** if the capacity of the output place is not exceeded, fire a single token \( \omega \) to the output place:

\[
N'(P_{\text{output},T_x}) = \begin{cases} 
N(P_{\text{output},T_x}) + 1 & \text{iff } N(P_{\text{output},T_x}) < C(P_{\text{output},T_x}) \\
N(P_{\text{output},T_x}) & \text{else}
\end{cases}
\]  
(2.10)

**Operation on token:** generate new token \( \omega \) with the following properties:

\[
\begin{align*}
\omega_{\text{cost}} & := c_{\text{carrier,area}} \cdot A_{\text{carrier}}; \\
OR & := c_{\text{carrier}}; \\
\omega_{\text{error}} & := RND(1, y_{\text{carrier}}), \\
\omega_{\text{rework counter}} & := 0.
\end{align*}
\]  
(2.11) \hspace{1cm} (2.12) \hspace{1cm} (2.13) \hspace{1cm} (2.14)

2.5.3. Process Step

A process is characterized by its cost per repetition \( c_{\text{proc}} \), the yield per repetition \( y_{\text{proc}} \), and the number of repetitions \( n_{\text{rep}} \) (e.g. the number of bonds when wire bonding a component). Machine depreciation and tooling costs are included in \( NRE_{\text{proc}} \).

**Firing:** if the capacity of the output place is not exceeded, subtract a single token \( \omega \) from the input place and fire to the output place:

\[
\begin{align*}
N'(P_{\text{input},T_x}) & = N(P_{\text{input},T_x}) - 1, \\
N'(P_{\text{output},T_x}) & = N(P_{\text{output},T_x}) + 1 \\
& \text{iff } N(P_{\text{output},T_x}) < C(P_{\text{output},T_x}).
\end{align*}
\]  
(2.15) \hspace{1cm} (2.16)
2.5. Specific Firing Rules for a Manufacturing Petri Net

2.5.4. Assembly Step

We describe an assembly step by its total number of input places \( k \), the number of tokens \( n_i \) in input place \( i \) required to fire, the cost of the assembly process \( \text{assembly} \), and the assembly yield \( \text{assembly} \). Again, machine depreciation and tooling costs are included in \( \text{NRE}_{\text{assembly}} \).

Firing: if the capacity of the output place is not exceeded, subtract a specified number of tokens \( n_i \) from every input place \( P_{\text{Input},i} \) and fire a single token to the output place:

\[
\begin{align*}
N'(P_{\text{Input},i}) &= N(P_{\text{Input},i}) - n_i \forall P_{\text{Input},i}, \\
N'(P_{\text{Output},o}) &= N(P_{\text{Output},o}) + 1, \quad \text{if} \quad N(P_{\text{Output},o}) < C(P_{\text{Output},o}).
\end{align*}
\]

\( n_i \) denotes the number of components of a single type to be assembled together, e.g. 50 resistors, or two SRAMs.

Operation on tokens: sum up all \( \omega'_{\text{cost}} \) over the specified number of tokens \( n_i \) per input place over the specified number of input places \( k \) forming a new token; continue as above with all other features of \( \omega' \):

\[
\begin{align*}
\omega'_{\text{cost}} &= \sum_k \sum_{n_i} \omega_{\text{cost}} + \text{assembly}, \\
\omega'_{\text{error}} &= \sum_k \sum_{n_i} \omega_{\text{error}} + \text{RND}(1, y_{\text{assembly}}) \forall \text{error}, \\
\omega'_{\text{rework counter}} &= \sum_k \sum_{n_i} \text{rework counter}.
\end{align*}
\]

2.5.5. Test Step (2 outputs)

Two output places are available: \( P_{\text{output OK}, o} \) and \( P_{\text{output FALSE}, o} \). This test step is defined by the test cost \( c_{\text{test}} \) and the test transparency \( TT \), which according to Eq. 3.10 can be described using the fault coverage \( FC \) and the clustering factor \( n_0 \). \( \text{NRE}_{\text{test}} \) describes the test machine cost and the test development cost.

Firing: determine output place: when no error is present, fire to \( P_{\text{output OK}, o} \), otherwise determine if a test escape has happened by checking the test escape tag. \( \text{RND}(1, TT) \) calculates this random test escape tag \( \in \{0, 1\} \) according to the binomial distribution given in Eq. 2.3. If no test escape is has happened, fire to \( P_{\text{output FALSE}, o} \):

\[
\begin{align*}
\text{if} \quad (\omega_{\text{error}} = 0) \text{ OR } ((\omega_{\text{error}} > 0) \text{ AND } (\text{RND}(1, TT) > 0)) \\
\text{then fire token to } P_{\text{output OK}, o}, \quad \omega'_{\text{rework counter}} := 0 \\
\text{else fire token to } P_{\text{output FALSE}, o}
\end{align*}
\]

\[
\begin{align*}
N'(P_{\text{Input},i}) &= N(P_{\text{Input},i}) - 1, \\
N'(P_{\text{Output},o}) &= N(P_{\text{Output},o}) + 1, \quad \text{if} \quad N(P_{\text{Output},o}) < C(P_{\text{Output},o}).
\end{align*}
\]

Operation on tokens: the cost for testing a unit has to be added. In case the unit passes the test, the rework counter is set to 0 (see above):

\[
\omega'_{\text{cost}} = \omega_{\text{cost}} + c_{\text{test}}.
\]
2.5.6. Test Step (3 outputs)

Three output places are available: \(P_{\text{output}0},T_e\), \(P_{\text{outputScrap}},T_e\), and \(P_{\text{outputREWORK}},T_e\). Also this test step is defined by the test cost \(c_{\text{test}}\) and the test transparency \(TT\). Moreover, \(n_{\text{max rework}}\) has to be specified. \(N_{\text{REtest}}\) describes the test machine cost and the test development cost.

**Firing:** determine output place: when no error is present, fire to \(P_{\text{output}0},T_e\), otherwise determine if a test escape has happened checking the test escape tag. \(RND(1,TT)\) calculates this random test escape tag \(G \in \{0,1\}\) according to the binomial distribution given in Eq. 2.3. If the test escape marker is not set, check the rework counter. If the maximum number of reworks possible \(n_{\text{max rework}}\) has been exceeded, fire to \(P_{\text{outputScrap}},T_e\), else fire to \(P_{\text{outputREWORK}},T_e\):

\[
\text{if } (\omega_{\text{error}} = 0) \text{ OR } ((\omega_{\text{error}} > 0) \text{ AND } (RND(1,TT) > 0)) \text{ then fire to } P_{\text{output}0},T_e, \omega_{\text{rework}} := 0 \text{ else fire to } P_{\text{outputREWORK}},T_e, \text{ else fire to } P_{\text{outputScrap}},T_e
\]

\[
N'(P_{\text{input}},T_e) = N(P_{\text{input}},T_e) - 1 \quad (2.27)
\]
\[
N'(P_{\text{output}},T_e) = N(P_{\text{output}},T_e) + 1 \quad (2.28)
\]
\[
\text{if } N(P_{\text{output}},T_e) < C(P_{\text{output}},T_e)
\]

**Operation on tokens:** the cost for testing a unit has to be added. In case the unit passes the test, the rework counter is set to 0:

\[
\omega_{\text{cost}}' = \omega_{\text{cost}} + c_{\text{test}} \quad (2.29)
\]

2.5.7. Rework or Repair Step

For a rework step, such as in Fig. 2.4, we need to know the rework success rate \(y_{\text{rework}}\), the damage rate \(y_{\text{damage}}\), and the cost of the rework \(c_{\text{rework}}\). \(N_{\text{REreWork}}\) denotes the repair machine cost and the setup cost.

**Firing:** if capacity of output place is not exceeded, subtract a single token \(\omega\) from the input place and fire to the output place:

\[
N'(P_{\text{input}},T_e) = N(P_{\text{input}},T_e) - 1 \quad (2.30)
\]
\[
N'(P_{\text{output}},T_e) = N(P_{\text{output}},T_e) + 1 \quad (2.31)
\]
\[
\text{if } N(P_{\text{output}},T_e) < C(P_{\text{output}},T_e)
\]

**Operation on token:** the cost for the rework step has to be added, and the rework counter is increased. Depending on the rework success flag \(RND(1,y_{\text{rework}}) \in \{0,1\}\), the errors in \(\omega_{\text{error}}\) are removed and new possible errors added, which are caused by damage during rework:

\[
\omega_{\text{cost}}' = \omega_{\text{cost}} + c_{\text{rework}} \quad (2.32)
\]
\[
\omega_{\text{error}}' = \omega_{\text{error}} \cdot RND(1,y_{\text{rework}}) + RND(1,y_{\text{damage}}) \quad (2.33)
\]
\[
\omega_{\text{rework counter}}' = \omega_{\text{rework counter}} + 1 \quad (2.34)
\]

2.5.8. Sinks

As already defined in Def. 2.6, two types of sinks exist: “Units to ship” \(P_{\text{Ship}}\) and “Units to scrap” \(P_{\text{Scrap}}\). These sinks are the output places of the last transitions in the flow and therefore do not require firing rules. For \(P_{\text{Ship}}\) a capacity \(C(P_{\text{Ship}})\) is specified, which is the total number of units to be produced. When the number of tokens \(N(P_{\text{Ship}})\) equals the capacity, no more tokens in this sink will be accepted and the net operation stops.
Summary
So far we have defined all structural information and all firing rules necessary to operate a manufacturing process as a Petri net. In the next section, we develop the mathematical calculation for the two metrics cost and quality and see how they can be computed using a Petri net-like model.

2.6. The Electronics Manufacturing Cost Quality Model

The total cost to manufacture a unit is the total cost necessary to produce a given number of units divided by this given number of units to ship (Equation 2.35). Using our definition from section 2.1, the total cost necessary can be broken into the three categories

- direct cost (DC),
- non-recurring expenditure (NRE), and
- reject cost or yield loss (YL).

To obtain the total cost the DC cost of all units shipped have to be summed, as well as the YL cost of all units scrapped. To this, we have to add the NRE sum of all processes. All units shipped are defined as \( N(P_{\text{ship}}) \). In analogy, the number of rejected units is denoted \( N(P_{\text{scrap}}) \), leading to Equation 2.36.

\[
\text{Costs to manufacture electronic systems} \quad \text{Given no. of systems shipped} \\
\sum_{N(P_{\text{ship}})} DC + YL + \sum_{\text{all proc}} NRE.
\]

Next to the final cost per unit, our second metrics is the quality level of this unit, the percentage of shipped units being error-free. Again, using the definition given in section 2.1, the quality level is the ratio of fully working units that passed the test and the total number of units that passed this test (Equation 2.37):

\[
\text{Quality Level}_{\text{unit}} = \frac{\text{No. shipped, fully working Units}}{\text{No. shipped Units}}.
\]

We will now show how Eqs. 2.36 and 2.37 can be calculated using a Petri net as defined in the previous section. The term \( \sum_{\text{all proc}} NRE \) in Eq. 2.36 does not require any information on process relations, as simply the NRE of all processes in the manufacturing flow have to be added. This can be done offline.

\[
\sum_{N(P_{\text{ship}})} DC \quad \text{traces the direct cost of the processes involved and the number of units required to produce a given number of systems to ship. This is done in our model using the specific variables in the tokens, in this case } \omega_i(\text{cost}). \text{ The number of shipped units is simply the total number of tokens collected in the sink } P_{\text{ship}}. \text{ Thus, } \sum_{N(P_{\text{ship}})} DC \text{ becomes}
\]

\[
\sum_{N(P_{\text{ship}})} DC = \sum_{\text{proc involved}} DC_{\text{proc}} \\
= \sum_{i=1}^{\omega_i(\text{cost})}.
\]
In analogy to Eq. 2.39 we define
\[
\sum_{N_{scrap}} YL_{scrap} = \sum_{\text{proc involved}} YL_{\text{proc}} = \sum_{N(P_{scrap})} \sum_{i=1} \omega_i (\text{cost}). \tag{2.40}
\]

Now, all necessary information for the calculation of Eq. 2.36 can be extracted from a generic Petri net, leading to

\[
\text{Total Cost}\_\text{Unit} = \frac{1}{N(P_{\text{ship}})} \cdot \left( \sum_{i=1}^{N(P_{\text{ship}})} \omega_i (\text{cost}) + \sum_{\forall P_{\text{scrap}}} \sum_{i=1}^{N(P_{\text{scrap})}} \omega_i (\text{cost}) + \sum_{\text{all proc}} NRE \right) . \tag{2.41}
\]

Eq. 2.37 can be expressed by the ratio of the number of tokens that are routed to the sink \( P_{\text{ship}} \) containing no errors and the total number of tokens in \( P_{\text{ship}} \), transforming into

\[
\text{Quality Level}\_\text{Unit} = \frac{1}{N(P_{\text{ship}})} \cdot \left( N(P_{\text{ship}}) | \omega_i (\text{error}) = 0 \right) . \tag{2.42}
\]

2.7. Summary

In this chapter we have identified a process-oriented modeling approach as the most suitable way to map HDP processes. Using three example HDP processes, we determined the basic items needed to describe such processes. Then, we laid the foundations to our manufacturing cost/quality model using a conditional stochastic Petri net.

The subsequent chapter shows how these foundations are transferred to a graphical process description language.
In this chapter we present the implementation of the model proposed in Chapter 2.

Section 3.1 shows the translation of the specific firing rules into the graphical description language MOE. The following section details how to measure and to generate input figures to compute cost and quality. Section 3.3 gives an example of a case study using MOE. Using this case study, the modeling and the parameterization process is explained. In section 3.4 some investigations on the computational performance of MOE and model limitations are presented.

3.1. The Implementation in MOE

A simple approach to adopt the structures and firing rules presented in the previous chapter would have been to use standard Petri net modeling tools such as MOSES [73] or DSPNexpress [77]. These tools would have provided a process description platform, though they still had to be customized to accomplish the firing rules described in the previous section.

Instead, we adopted a graphical user interface called CostAS [78] where an easy implementation of the firing rules was possible. An additional advantage of adopting CostAS is the full control over input and output interfaces, as we see later in this work. Moreover, in order to hide the modeling complexity from the end user, a HDP design engineer, a dedicated graphical description language following an intuitive manufacturing flow-like description is preferable to an abstract Petri net.

MOE (Modular Optimization Environment) [79] is divided into

a **front end**, responsible for model construction, parameterization, and result analysis, and

a **back end**, responsible for the stochastic Petri net analysis running the “token game”.

Fig. 3.1 shows the interaction between the front end and back end of MOE. The model to be analyzed is described in the front end using flow items, then the front end generates an ASCII model description which is parsed to the back end simulation program “simengine”. This program runs the “token game” for the Petri net structure using the parameters specified and generates another ASCII file containing the results. These results in turn are parsed to the front end where they are displayed in the result window.
3.1. The Implementation in MOE

Figure 3.1: Interaction between front end and back end of MOE. The front end, written in the interpreter language tcl/tk provides the graphical user interface to enter the model structure and the parameter data. When starting a calculation, a ASCII file <netdata> is generated containing all information required to conduct the simulation. The simulation is conducted by a C program called “simengine”, which is available as executable to speed up calculation. “simengine” again writes the results into an ASCII file and parses them to the front end. Here, the results can be displayed for further analysis. With a <netdata>-file, a simulation can be executed without requiring the front end.

Front End

The graphical process description integrated in MOE’s front end focussed on five basic descriptive flow item classes (the transitions described in section 2.5):

- components (and their subclass carriers), the sources of a flow
- simple processes (and their subclass reworks),
- assemblies (joining two to five different streams),
- tests (switching between two or three output streams), and
- result containers (with their subclasses SCRAP and sensors); result container and SCRAP are the sinks of the manufacturing flow.

The general usage of MOE is described in full length in the user manual [80]; therefore, we only give a summary and highlight the issues that are important for the latter part of this work.

Flow Items

Flow items can be selected from the “Item” menu entry. The process structure is constructed by connecting the input and output buffers of the flow item. These buffers represent the input and output places of the Petri...
3.1. The Implementation in MOE

net transitions. MOE checks for consistency by ensuring that all input and output buffers are connected. An overview of the flow items is given in Fig 3.2. The top row shows (from left to right) a component, carrier (special form of a component), simple process, process with feedback, and a rework (special form of the simple process). Each item is uniquely identified by an ID number in its upper half. A description tag helps the user to distinguish the different items. Input buffer connections are denoted by light gray triangles, output buffer connections by dark gray triangles. Component and carrier items are the sources of a flow. The middle row shows a 2-input assembly, 4-input assembly, 2-output test, 3-output test, and a 2-input/2-output test. Process, assembly, and test items can be transformed dynamically, as we will see below. In the bottom row, a sensor (allowing for intermediate result checks), a scrap and a collector are depicted. A sensor has no influence on the tokens or packets. Scrap and collector are the sinks of the flow, and only a single collector is allowed to appear.

**Figure 3.2**: Overview of MOE flow items: top row: (left to right) component, carrier, simple process, process with feedback, rework; middle row: 2-input assembly, 4-input assembly, 2-output test, 3-output test, and 2-input/2-output test; bottom row: sensor, scrap, and collector.

After constructing the flow itself, each flow item has to be customized by entering item-specific data. Moreover, for process, assembly, and test, the number of inputs and outputs can be changed dynamically, as can be seen in Fig. 3.3.

The item-specific parameters can either be entered as numbers (denoted as “CONSTANT”) or variables (denoted as “GLOBAL”) from a predefined list in the batch parameter dialog (see Fig. 3.3(d)). Using the batch parameters, a sweep over two independent base variables (here: “yield1” and “fc1”) can be accomplished to investigate the influence of parameter changes.

**Back End**

The full model description of the <*.flw>-file is stripped from the graphical information down to the pure Petri net structure. The resulting <netdata>-file contains only information on which input is connected to which output and on the parameter values. This description is translated to a pointer chain, so the subsequent token game can operate only on small pointers instead of large variable constructs. This implementation has been chosen to speed up the evaluation process. For the same reason, the back end is available as a fully executable program (compared to the front end, which has to be translated by a run time interpreter).

**Results**

Then, the results are displayed in the “result window”, depicted in Fig. 3.4. In the upper left corner either the average cost, the average quality, or the number of the tokens passing by can be chosen for display. In the upper right corner, the flow item to check (either a sensor or the collector) has to be specified. The window shows the chosen variable on the y-axis, and the value range for the main base variable (here: “yield1”).
3.1. The Implementation in MOE

Figure 3.3: Entering item-specific parameters: either numbers (identified as "CONSTANT") or variables (denoted as "GLOBAL") can be specified. The numbers for the variables are to be specified in the batch parameter dialog (Fig. d), depending on the main (here: "yield1") and the second (here: "fc1") base variable. Note that for assembly items the number of inputs can be customized dynamically (Fig. c), and for processes and test items special in/outputs can be specified (Figs. a & b).
3.2. Input Parameters for HDPTechnologies

After setting up the model structure and before running the net, the input parameters for a specific fabrication model have to be specified. Doing so, for every flow item the respective descriptive data detailed in subsections 2.5.1 to 2.5.7 have to be identified. All this data can be divided into three categories: cost-related, yield-related, and fault coverage-related.

3.2.1. Cost Data

Which figures and information to include to compute the cost to manufacture an electronic system has already been shown in Fig. 2.1. For each process item, we now have to identify the direct cost $DC$ and the non-recurring expenditure $NRE$.

When using external contract manufacturers, $DC$ and $NRE$ can be found in the quotation, otherwise $DC$ has to be extracted from material and labor costs. Machine related cost (setup and depreciation) plus general manufacturing overhead go into $NRE$. Whereas material and labor costs are straight-forward to determine [81], especially the computation of the depreciation and overhead costs is much more arguable and therefore company specific.

Figure 3.4: MOE result window: upper left corner shows result selection between cost and quality; the graph itself plots the range of the main base variable (here: “yield”) versus the result (here: Average Cost). Cost results are depicted as three-part bar, detailing contributors direct cost “DC” (bottom), yield loss “YL” (light gray, middle), and “NRE” (dark gray, top).
Considering overhead, some companies organize all business units as cost centers, where every cost center has to pay for using central services, other companies provide these central services “for free” to their internal departments adding an overhead percentage to the final cost. Examples for such central services are e.g. building rent, insurance, consumables as water and electricity, management, marketing and sales activities. In our case studies, we prefer to identify a specific amount of money to be shared, denoted as NRE, instead of using an overhead scaling factor.

Various mechanisms also exist for depreciation, since depreciation is strategy and tax dependent [55]. Issues are e.g. the duration in years to depreciate capital investment and the method to calculate the annual amount to depreciate. For all of our examples we have used the simple straight line depreciation, dividing the total amount to depreciate by the number of years to obtain the annual depreciation cost.

When comparing process models, the analyst has to ensure that the data has been extracted using identical definitions. In order to provide the reader with an idea on the cost range, Table 3.1 gives an overview on HDP technology costs. The data has been collected from various sources and discussions with colleagues and must therefore be interpreted with caution, as preconditions are not the same. In general, the data has been calculated for large quantities, so setup costs can be neglected. Moreover, the data originates from various years, so especially the substrate and IC cost data might not be up-to-date. Also, testing and repair costs are difficult to analyze, since both names are general labels, and no exact description of the work undertaken was available.

Table 3.1: Examples for cost data of HDP technologies in comparison to other technologies.

<table>
<thead>
<tr>
<th></th>
<th>Cost</th>
<th>Year</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMD placing and soldering</td>
<td>&lt;0.01$/unit</td>
<td>1998</td>
<td>[82]</td>
</tr>
<tr>
<td>WB</td>
<td>0.0017$/pin</td>
<td>1998</td>
<td>[11]</td>
</tr>
<tr>
<td>TAB</td>
<td>0.007$/pin</td>
<td>1998</td>
<td>[11]</td>
</tr>
<tr>
<td>FC</td>
<td>0.0014-0.0083$/pin</td>
<td>1998</td>
<td>[11]</td>
</tr>
<tr>
<td>MCM-C</td>
<td>0.2$/(cm²-layer) LTCC</td>
<td>1993</td>
<td>[27]</td>
</tr>
<tr>
<td></td>
<td>0.12$/(cm²-layer) HTCC</td>
<td>1993</td>
<td>[27]</td>
</tr>
<tr>
<td>MCM-D</td>
<td>&gt;6$ cm⁻²</td>
<td>1993</td>
<td>[27]</td>
</tr>
<tr>
<td></td>
<td>1-2$ cm⁻³</td>
<td>2000</td>
<td>[14]</td>
</tr>
<tr>
<td>MCM-L/SBU</td>
<td>0.2$ cm⁻²</td>
<td>1993</td>
<td>[27, 83]</td>
</tr>
<tr>
<td>PCB</td>
<td>&lt;0.01$/(cm²-layer)</td>
<td>1993</td>
<td>[27]</td>
</tr>
<tr>
<td>GaAs IC</td>
<td>70$/cm²</td>
<td>2000</td>
<td>[84]</td>
</tr>
<tr>
<td>CMOS IC</td>
<td>50$/cm²</td>
<td>1996</td>
<td>[85]</td>
</tr>
<tr>
<td>Testing</td>
<td>1-20$/min</td>
<td>1997</td>
<td>[86, 87]</td>
</tr>
<tr>
<td>Repair</td>
<td>1-20$/min</td>
<td>1997</td>
<td>[86]</td>
</tr>
<tr>
<td>Setup/tooling Cost</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9x9in² thin film mask</td>
<td>&gt;1'000$</td>
<td>2000</td>
<td>NDA</td>
</tr>
<tr>
<td>Custom CSP package</td>
<td>30'000-40'000$</td>
<td>2000</td>
<td>NDA</td>
</tr>
<tr>
<td>Stencil solder screen printing</td>
<td>≈500$</td>
<td>2000</td>
<td>NDA</td>
</tr>
</tbody>
</table>

3.2.2. Yield Data

Similar to cost estimates, yield assumptions extrapolate from former experience to future products. For simple repetitive processes, such as wire bonding, statistics suffice to extract yield data because the process itself remains almost constant within its suitable process window. Thus, from previous bonding experiments we can forecast the number of faulty bonds to be expected based on a given number of bonds.

Process and Product Specific Parameters

For more diverse processes, such as substrate fabrication, a product specific parameter also has to be taken into account. For substrate fabrication, the process-specific parameter is the defect density \( D_0 \), and the product-specific parameter is the critical area \( A_C \) (the area where the center of a defect of size \( x \) is located in order to cause a functional error). These two factors have to be combined in a suitable yield model.
The most known yield models are the Poisson yield model and the negative binomial yield model. While the Poisson model (Eq. 3.1) assumes that the spatial distribution of defects is random, the negative binomial model (Eq. 3.2) is based on the assumption that the likelihood of a defect occurring at a given location increases linearly with the number of events already occurred, introducing a clustering factor \( \alpha \).

\[
\text{Poisson: } Y = e^{-A_C D_0}
\]

\[
\text{Negative binomial: } Y = \left(1 + \frac{A_C D_0}{\alpha}\right)^{-\alpha},
\]

where \( A_C \) is the average critical area susceptible to defects, \( D_0 \) is the average defect density, and \( \alpha \) is a clustering factor, experimentally found to be ranging between 0.3 and 5.0 [88].

Since the Poisson model does not take into account that neighbored defects can damage the same substrate/die, it gives in some cases an over-pessimistic yield estimate (see Fig. 3.5). A comprehensive introduction comparing various yield models and giving sufficient references for further readings can be found in [58, 89].

Once the process-specific defect density \( D_0 \) and its clustering factor \( \alpha \) have been determined, the critical area extraction remains the most complicated part. Focusing on spots defect yield loss only, basic failure models for interconnect substrates are primarily based on line-line shorts and line opens\(^1\). Whereas the calculation of the “open” failure critical area is straightforward, depending on the total interconnect length, the computation of “short” critical area requires more effort:

- Monte Carlo based methods. Williamson et al. [90] already applied this method to MCM substrates. The algorithm places random defects on the layout, checking for errors to extract the critical area. This is an accurate but computationally intensive process.

\(^1\)For semiconductor structures, where transistors are constructed using a three-dimensional build-up, more primary failure modes have to be taken into account.
• Geometrical methods. Presented by Allan and Walton [91], they use “bloating” of layout structures to extract the critical area, either used for primitive cells or for entire layouts. Although this algorithm is efficient, scaling with \(O(n \cdot \log n)\) (\(n\) the number of structures), it still requires several hundreds of seconds on a Sun Ultra 1 workstation for 1,000 SRAM cells in an array.

• Virtual [92] or equivalent [93] layouts, i.e. statistical images of a real layout, help in extracting critical area for large circuits, but induce a higher degree of estimation error.

• Finally, pattern oriented methods tile the layout into primitives and extract the critical area for each tile. Depending on the tiling algorithm, different levels of accuracy can be achieved, but execution times of up to several hours even for a moderate number of transistors (<10⁵) are reported [94, 95]. This type of algorithm pays off for regular designs, such as crossbars [96].

Table 3.2: Examples for yield data of HDP technologies in comparison to other technologies.

<table>
<thead>
<tr>
<th>Defect density (IC)</th>
<th>Yield</th>
<th>Year</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>WB</td>
<td>99.997-99.993% per bond</td>
<td>2000</td>
<td>[99]</td>
</tr>
<tr>
<td>TAB</td>
<td>99.95% per bond</td>
<td>1993</td>
<td>[27]</td>
</tr>
<tr>
<td>Soldering SMDs/FC</td>
<td>99.999-99.9985% per bond</td>
<td>2000</td>
<td>[99]</td>
</tr>
<tr>
<td>ICs (first pass)</td>
<td>50-85%</td>
<td>1998</td>
<td>[100]</td>
</tr>
<tr>
<td>(tested)</td>
<td>up to 99.99%</td>
<td>1995</td>
<td>[44]</td>
</tr>
</tbody>
</table>

\(\dagger\): based on a critical defect size <100nm

Table 3.2 provides some basic data and generic yield numbers. In general, yield data for interconnect substrates, PCBs, and ICs is found to range from 50 to 99.99%. Prototyping yield is reported to be as low as 50-60%, while massively improving during the yield ramp-up phase, getting close to 100%. Next to the actual yield stage a component is in, another key factor is the IC or substrate complexity in terms of gates, feature size, and interconnect density. The more complex a substrate or IC, the lower the slope of the ramp-up curve is.

3.2.3. Test Transparency and Fault Coverage

The fact that there might be erroneous units having passed the test undiscovered leads to the question of how to quantify this event.

Defect Level Models

The quality level \(QL = 1 - DL\) has been defined above in section 2.1. \(DL\) is therefore a function of stochastic variables as yield of components and processes \(y\) and transparency \(TT\) (Eq. 3.4). The probability of a unit containing an error is \(1 - y\), the probability of passing a test \(TT\) (per definition), and the probability of passing the test at all is the sum of the probabilities of the uncorrelated events “unit contains no error” \(y\) and “unit contains error but passes test” \((1 - y) \cdot TT\).

\[
Quality = 1 - DL
\]

\[
= 1 - \frac{\text{Probability \{unit passes test|contains error\}}}{\text{Probability \{unit passes test\}}}
\]

\[
= 1 - \frac{(1 - y) \cdot TT}{y + (1 - y) \cdot TT} \quad (3.4)
\]

Equation 3.4 is not the only possible formula for \(DL\). Several models exist to link test transparency to quality (or defect level) [59], and the most simple relation, being developed by Wadsack, is given in Eq. 3.5, where \(f_c\) denotes the fault coverage:
3.2. Input Parameters for HDP Technologies

\[ DL = (1 - y)(1 - fc). \]  

(3.5)

Williams and Brown suggest another relation (Eq. 3.6):

\[ DL = 1 - y^{(1 - fc)}. \]  

(3.6)

Both equations are based on the assumptions that faults occur independently. Agrawal and Seth [35] have shown that fault clustering (multiple logical faults caused by a single fault, and multiple faults located on the same unit) decrease the test transparency probability, therefore presenting the model in Eq. 3.7. \( n_0 \) indicates the fault clustering determined by exhaustive failure analysis and curve fitting.

\[ DL = \frac{(1 - y)(1 - fc)e^{-(n_0-1)fc}}{y + (1 - y)(1 - fc)e^{-(n_0-1)fc}}. \]  

(3.7)

\[ DL \bigg|_{n_0=1} = \frac{(1 - y)(1 - fc)e^0}{y + (1 - y)(1 - fc)e^0} \approx \frac{(1 - y)(1 - fc)}{y + (1 - y)(1 - fc)}. \]  

(3.8)

(3.9)

For \( n_0 = 1 \) Eq. 3.7 becomes Eq. 3.8, which can be approximated by the Wadsack model (Eq. 3.9). The reason is that for a very small number of defects, in the range of \( 10^{-6} \), an equiprobability can be assumed. A comparison of the different models is shown in Fig. 3.6. Setting

\[ TT = (1 - fc)e^{-(n_0-1)fc}, \]  

(3.10)

Equation 3.7 equals Equation 3.4, modeling the probability for an erroneous unit to escape the test. Again, this probability can be simulated using Monte Carlo sampling. The number of errors already present in a unit can be used for \( n_0 \) representing the clustering effect.

![Figure 3.6: Comparison of various defect level vs. fault coverage models.](image)

**Fault Coverage for ICs**

To calculate the fault coverage of a specific test for ICs (fraction of failures detectable by the test to all failures included in the fault model), a MIL-STD-883 Procedure 5012 exists (“Fault Coverage Measurement...”)
for Digital Microcircuits”), for comments see [101]. In principle to determine \( f_c \) for any type of test, (1) one has to choose a fault model and to calculate the total number of possible faults. Whereas this strategy is straightforward for ICs and a simple stuck-at fault model, it gets very intensive including transition, state coupling and 1-to-n coupling faults, leaving the fault calculation task very often to automated test pattern generators (ATPGs). For a system test determination of all possible faults becomes even more complicated because simple fault models as described above are no longer available. Next (2), the faults have to be partitioned into fault classes (where every fault creates identical behavior), and for every class a representative fault has to be chosen; the result is the final detectable fault universe. Finally (3), the fault coverage of a specific test can be determined either by an exhaustive simulation of all faults or by fault sampling (see again [101]). ATPGs do these steps all together generating test patterns and checking the respective fault coverage. The test itself can be conducted using either fully external equipment or partly built-in self test (BIST) strategies (or a combination of both). Reachability of a system’s or IC’s logic states and observability of its internal nodes is ensured using Boundary Scan architecture (aka Scan Bus or IEEE Std 1149.1 (JTAG)) [102]. Compared to fully external test equipment, the use of BIST or Boundary Scan can have a higher degree of observability of internal nodes in a system. Therefore, BISTs can improve the fault coverage significantly while lowering the cost for external test equipment. For pure digital logic ICs, fault coverages close to 100% have been reported (see e.g. [103]).

**Fault Coverage for Assembly Processes**

For the electronics assembly process, no ATPGs are available. The fault universe consists of component-related errors (misplacement, wrong part, wrong direction, defective part, etc.) and process-related errors (tomb-stoning of SMDs, weak soldering, solder residues, etc.). Available test strategies are Automated Optical Inspection (AOI), In-Circuit Test (ICT), X-ray test, and Functional Test (FCT) [104]. AOI and X-ray test can only check for more mechanical oriented faults, while ICT and FCT also check electrical aspects. FCT might also use IC BIST and Boundary Scan when available. A very good example of a hierarchical test architecture taking advantage of IC level test structures can be found in [103]. Consequently, the combination of both mechanical and electrical test strategies yields in higher fault coverage [104]. AOI is also used in substrate fabrication [105], leading to a significant improvement in electrical yield. In this context, AOI is even used as an intermediate process step.

Examples for \( f_c \) are difficult to compare, as they depend on defect probability and the fault universe. Again to provide an idea, the following fault coverages for testing in an assembly process are given in Table 3.3 [104].

<table>
<thead>
<tr>
<th>Failure Type</th>
<th>AOI</th>
<th>ICT</th>
<th>FCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly</td>
<td>60%</td>
<td>96%</td>
<td>86%</td>
</tr>
<tr>
<td>Components</td>
<td>19%</td>
<td>89%</td>
<td>82%</td>
</tr>
<tr>
<td>Soldering</td>
<td>96%</td>
<td>92%</td>
<td>83%</td>
</tr>
<tr>
<td>Mechanical</td>
<td>85%</td>
<td>0%</td>
<td>30%</td>
</tr>
</tbody>
</table>

**Table 3.3: Fault coverages of test types in electronics assembly processes [104].**

**3.2.4. Summary**

We have to point out that data collection is laborious work, since most of the information required is considered to be company confidential and therefore is not freely available. The purpose of this section was to provide the reader with some ideas on the structure of MOE input data and with some examples of the data itself, in case no problem-specific data would be available.

In order to demonstrate the modeling of processes and the trade-off process itself, we show an example in the following section.
3.3. A Case Study using MOE

To give the reader an impression of how MOE can be used for manufacturing analysis and how the results look, we present a case study on evaluating the benefits of using CSPs instead of bare dies [106].

**Example 3.1** The GPSMS1® of μ-blox ag is a fully self-contained global positioning system (GPS) receiver multichip module (MCM) at a single operating voltage of 3.3V. Based on the SiRFstar/LX® chip set, the module provides complete GPS signal processing from antenna input to serial data output. The GPSMS1® fits into the form factor of a PLCC84 package (plastic leaded chip carrier), allowing for dual side assembly. On the top side, the RF front end chip, antenna connector, and additional components are mounted in surface-mount technology (partially housed), whereas on the bottom side microprocessor, DSP correlator, and memory are mounted as bare dies, wire bonded and overmolded [6].

During recent years, CSPs have shown to be a viable alternative to bare dies due to their ease in handling, better obtainability, and easier mounting procedure. As the market is ever demanding for form factor shrinking and cost reduction, an investigation was due on the potential benefits of CSPs for the next generation of GPSMCMs.

![Figure 3.7](image)

(a) existing wire bonded GPSMS1®  
(b) study using CSPs with 0.65mm pitch

**Figure 3.7:** Two different implementation of the μ-blox GPSMS1® (bottom view); footprints are marked white and black, respectively; all measures in mm.

We first analyzed the footprints and the routing congestion of the GPSMS1® module [106]. Based on the footprint models developed in [9] it turned out that by using CSPs with ball pitches down to 0.65mm it was possible to loosen the substrate design rules. This possibility offers a chance for cost reduction, also due to improved manufacturing yield, but possible savings had to offset investments into custom CSPs and an expensive redesign. Fig. 3.7 shows the footprint comparison of the two alternatives. Thus, the task was to compare two different manufacturing strategies for the GPSMS1®.

**Example 3.1 (continued)** A flow description for the manufacturing process is shown in Fig. 3.8. The components enter the flow at the material supply. First, the SMDs on the front side are attached to the laminate carrier, and the populated carrier undergoes a reflow soldering, followed by a post-cleaning step. Then, the carrier is flipped and the dies are attached to the back side. Preceded by a plasma cleaning step, the dies are gold wire bonded to the substrate, and the wires undergo an immediate electrical test. Not explicitly shown is a removal and repair of defective units. The functional units are then glob-topped. Additional SMD components on the back side are attached and again reflowed. The panels are separated, and an RF protection lid is mounted and soldered in hot air. The leadframe is attached and soldered with a hot bar in order not to damage the already assembled components. The module then undergoes the final test program and is delivered to the customer, which will use the module as a common SMD component, soldering it e.g. to a daughter card.

Since a model is only intended to be a simplified abstraction of the system that includes enough detail to derive the desired performance measures [107], not all process steps described above have to be included in
3.3. A Case Study using MOE

Figure 3.8: Flow description of the GP’s MS1® assembly process [61].

Table 3.4: Mapping of flow description items to MOE model.

<table>
<thead>
<tr>
<th>Flow Description Fig. 3.8</th>
<th>MOE ID Fig. 3.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material Supply</td>
<td>ID1, ID0, ID13, ID4</td>
</tr>
<tr>
<td>SMD Assembly, Reflow Soldering, Cleaning</td>
<td>ID7</td>
</tr>
<tr>
<td>Die Attach, Plasma Cleaning</td>
<td>ID2</td>
</tr>
<tr>
<td>Au-Wire Bonding</td>
<td>ID6</td>
</tr>
<tr>
<td>Electrical Test</td>
<td>ID14, ID15</td>
</tr>
<tr>
<td>Epoxy Glob Top, SMD Assembly, Reflow, Panel Separation, RF Lid, Hot Air Soldering, Leadframe Mounting, Hot Bar Soldering</td>
<td>ID8</td>
</tr>
<tr>
<td>100% Test Program Label</td>
<td>ID3</td>
</tr>
</tbody>
</table>

our model of the fabrication process. Instead, it is more important to stress the basic skeleton of the process. Therefore, test and rework steps have to be included in any case. Subsequent process steps can be combined into a single one provided that

- each cost contribution is small compared to the component cost;
- each yield is above 99.99% AND the number of repetitions is one;
- none of the process steps is under special investigation;
- the defect type (such as “component” or “interconnect”) is the same.

Example 3.1 (continued) The translation of the process described above can be found in Fig. 3.9. Compared to the detailed description above, we have integrated the cleaning steps into the assembly processes. Since the process part under interest affects only the back side (as the bare dies located there could be replaced with CSPs), all front side assembly steps are combined into “Solder Front (ID7)” and “Metal Cap (ID8)”. IDs 0, 13, and 4 represent the sources for the substrate, the DSP, the microcontroller, and the memory dies. Test steps are ID14 and ID3, respectively.

In analogy to Fig. 3.9, a fabrication model using other substrate design rules and CSPs also had to be developed, depicted in Fig. 3.10.
Example 3.1 (continued) For the alternative CSP-based process, custom CSP fabrication steps have to be inserted (IDs 6 and 12), while wire bonding and bond test/rework could be removed. The front side is now assembled (ID7) following the back side.

In our GPS example, the cost data was available due to quotations or existing data, the yield figures had to be partially estimated.

Example 3.1 (continued) In Table 3.5, part of the case study's cost data can be found, emphasizing the differences between the existing and the new implementation. The main cost advantage of the CSP solution is caused by the lower backside assembly cost (by saving the cost for wire bonding and bond testing) and by the lower yield loss. On the other hand, the component cost is higher, and some investments for custom CSPs have to be incurred. What remained unchanged are all the cost/yield data for front side assembly, capping, final test, and investment for the processor mask ROM and the substrate. The cost of the substrate itself goes down with the loosened design rules, as well as the yield improves from 0.999 to 0.9999. However, the cost for a substrate redesign has to be taken into account.

For the second scenario, keeping 80/80/250 rules and operating with a 0.5mm CSP pitch, substrate cost and yield remain the same as for the wire bond case; the other data applies as for the 0.65mm CSP solution.

For the entire case study, the fault coverage of the test steps has been set to 100% in order to concentrate on potential cost savings, thus suppressing a possible quality impact. Therefore, also the final quality is 100%.

The cost calculation results for a depreciation period of two years and a target quantity of 100'000 units per year can be found in Fig. 3.11.
3.3. A Case Study using MOE

Figure 3.10: Alternative fabrication flow: manufacturing of a GPS front end using CSPs and 100μm/100μm/350μm design rules.

Example 3.1 (end) The cost saving of the (100/100/350 rules; 0.65mm pitch)-CSP solution are in total about 6%. The direct cost reduction contributes around 3%, but the NRE is about 0.5% higher. The reduced yield loss has the highest contribution to the improvement. The (80/80/250 rules; 0.5mm pitch)-CSP solution results only in a cost saving of around 4% compared to the existing solution. Moreover, loosening the design rules widens the substrate supply channels. Thus, the cost savings justify employing a CSP packaging solution for a next redesign of μ-blox GPS MCM.

Table 3.5: Partial data for the cost/yield calculation; IC cost data is confidential and therefore given only in arbitrary cost units a.u.; NRE cost considers only additional cost compared to the existing solution.

<table>
<thead>
<tr>
<th></th>
<th>wire bond 80/80/250 (existing solution)</th>
<th>CSP 0.65mm 100/100/350</th>
<th>CSP 0.5mm 80/80/250</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DC [a.u.]</td>
<td>y [%]</td>
<td>DC [a.u.]</td>
</tr>
<tr>
<td>DSP</td>
<td>8.50</td>
<td>99.9</td>
<td>10.50</td>
</tr>
<tr>
<td>μC</td>
<td>5.75</td>
<td>99.9</td>
<td>7.75</td>
</tr>
<tr>
<td>SRAM/Flash</td>
<td>6.50</td>
<td>99.8</td>
<td>7.15</td>
</tr>
<tr>
<td>Substrate</td>
<td>4.25</td>
<td>99.9</td>
<td>3.25</td>
</tr>
<tr>
<td>Assembly</td>
<td>18.20</td>
<td>99.9</td>
<td>12.90</td>
</tr>
<tr>
<td>Back side</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without wire bond</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wire bond</td>
<td>99.99</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
3.4. Performance and Limitations

Using MOE, it was possible to set up the two different process flows of Figs. 3.9 and 3.10 in the case study within 5 min each, entering the data took about the same time. The simulation with 15'000 iterations took about 0.85 sec on a SUN Ultra, on a LINUX 586-portable about 1.5 sec. The Monte Carlo simulation needs a certain number of iterations until it is stable. The settling for a complex model of more than 20 flow items is shown in Fig. 3.12, presenting the three results direct costs $DC(\ast)$, yield loss $YL(\square)$, and defect level $DL(\times)$. As can be seen, the values for $DC$ and $DL$ settle very fast, after approx. 150 iterations. The NRE cost can be calculated offline and therefore does not influence the settling time. The yield loss $YL$ needs up to 500 iterations and is finally oscillating within a variance band. This effect is due to the regular arrival of faulty units in the SCRAP collectors, causing a jump in $YL$. Throughout the next iterations, when no faulty units are detected, $YL$ decreases as it is shared among more good units.

The choice of Monte Carlo simulation makes the calculation more computation intensive, but within reasonable bounds. Other tools, such as SavanSys (cf. [10]) being based on recursive formulae, require more time for process modeling, and the computation time is about the same order of magnitude. SimWitch [42], also featuring process oriented modeling, operates in the same order of magnitude. Only the spreadsheet based tools like Envision [43] have a shorter computation time, but require approx. ten times longer for model changes.

Model limitations

Using the Petri net approach and its extension of colored tokens, theoretically every production process can be modeled [71]. Using the MOE graphical language (the flow items shown in Fig. 3.2), it is possible to directly model any manufacturing process that does not require a parallel transition (cf. Fig. 2.8). This type of transition has not been implemented yet as MOE was aiming for a “construction”, not a “destruction” process.

The concept of a parallel transition could indeed be helpful when e.g. calculating wafer processing...
operations ending with a dicing process step. As already mentioned, from an abstraction point of view it does not make sense to split such a single wafer process into several hundred parallel die handling processes. These die handling processes are better modeled as single sequential flow. A possible implementation could be a simple process subtracting only a single token from its input place, while firing a number of tokens equivalent to the number of dies to its single output place. The operation on the tokens would translate wafer cost and yield to die cost and yield. Yet, the actual solution (also adopted by SimWitch and Envision) is to model the wafer manufacturing process separately and to take the results of cost and yield as an input for a simple process step in the die handling process.

As a manufacturing model can be adopted by all products using the same type of fabrication process, a back link from MOE to other analysis tools such as JavaCAD [9] and the Xynetix Navigator [108] is provided. Within these tools it is possible to define where components are processed, to update design specific cost and yield information of the model, to start the calculation, and to read the results into the analysis tool for further examination.

3.5. Summary

In this chapter we have presented the implementation of our HDP process model as a graphical cost modeling tool MOE. MOE is, next to cost calculation, able to compute the defect level or quality of an electronic product based on yield and test efficiency information. Due to its graphical user interface and its standardized building blocks, MOE is especially suitable for comparing two implementations using different manufacturing processes, as shown in Section 3.3.

In the following chapter we see how the quality factor influences trade off decisions, how to rank different solutions, and what to do to find an optimum setup.
In the previous chapter we have presented a model to calculate cost and quality aspects at the same time. When conducting a parameter variation analysis or comparing different models, the question of how to rank and order various solutions arises. This problem will be tackled in section 4.1, by introducing the Pareto concept as the preferable ranking scheme. Because some cost-quality trade-off problems are located in a large trade-off space, covering more than 100,000 different parameter or model combinations, a dedicated search methodology is required. Section 4.2 briefly reviews standard search and optimization procedures and presents evolutionary algorithms as the method of choice. In section 4.3 the implementation of this search algorithm together with the MOE calculation is presented. We analyze Pareto stability problems and different types of Pareto fronts in section 4.4 and conclude with a chapter summary.

4.1. Introduction to Multi-Objective Optimization

Trade-off investigations, in our case the cost-quality trade-off, can be considered as multi-objective optimization problems.

Definition 4.1

**Single-objective Optimization Problem.** In a general single-objective optimization problem, for a decision vector \( x \) the size \( n \) and a target function \( f \), an objective value \( y \) has to be found satisfying the following condition:

\[
\minimize \quad y = f(x),
\]

where \( x = (x_1, x_2, \ldots, x_n) \in X \),

\( y \in Y \).

**Multi-objective Optimization Problem.** In a general multi-objective optimization problem, for a decision vector \( x \) the size \( n \) and a target function vector \( f \) the size \( k \), an objective vector \( y \) has to be found satisfying the following condition:

\[
\minimize \quad y = f(x) = (f_1(x), f_2(x), \ldots, f_k(x)),
\]
4.1. Introduction to Multi-Objective Optimization

where \[ x = (x_1, x_2, \ldots, x_n) \in X, \]
\[ y = (y_1, y_2, \ldots, y_k) \in Y. \]

\( X \) is called the decision space, \( Y \) the objective space.

"True" Multi-Objective or "Quasi-Single" Objective?

For multi-objective problems hardly any true multi-objective optimization technique had been available until recently [54]. What has been done so far is to aggregate multi-objective problems into single-objective target functions ("quasi-single"), e.g. using weighting factors for the specific objectives [109]. Another approach was the dedicated translation of multiple objectives into a single one, such as test escapes to cost [52] and time-to-market to cost [110]. The main advantage in doing so is that various solution techniques for single-objective problems do exist (see section 4.2).

On the other hand, the drawbacks when solving a multi-objective problem using "quasi-single"-objective optimization are also numerous:

- In [54] it has been shown that in some cases not all solutions will be detected.
- While setting the weights, implying "decision making before search" [54], a dedicated knowledge of the search space is required to generate these weights, or various runs are necessary to assess the impact of the weights.
- Aggregating implicitly into a single objective allows compensation of a specific underperforming objective, which is often not desired. Consider the case when for a product low quality comes with a very attractive cost. Measures preventing this result complicate the optimization procedure.
- Moreover, to obtain a Pareto-front solution set and to make a trade-off decision, several independent runs are necessary where synergies cannot be exploited [111]. Single results do not permit a design parameter space exploration to be conducted, which is one of the main goals of virtual prototyping. Problems as in Example 1.1 cannot be solved.

Especially the last issue makes the case for a "true" multi-objective optimization in order to support trade-off investigations. Doing so, we need a specific ranking mechanism for multi-objective optimizations in order to compare different solutions.

Ranking of Solutions

Whereas in a single-objective optimization two decision vectors, e.g. \( \mathbf{a} \) and \( \mathbf{b} \in X \), can be ranked in a strict order according to the single objective function \( f \), resulting in either \( f(\mathbf{a}) \leq f(\mathbf{b}) \) or \( f(\mathbf{b}) \leq f(\mathbf{a}) \); for multi-objective problems only partially ordering schemes are present, as introduced by Wilfredo Pareto (see [112]).

Consider the example in Fig. 4.1, where the objectives "defect level" and "total module cost" are plotted, both to be minimized. Depicted are ten solutions in the objective space (\( \mathbf{a} \) to \( \mathbf{j} \)), with \( \mathbf{g} \) as reference (circled). \( \mathbf{a, b} \) are better than (or superior to) \( \mathbf{g} \) in terms of both cost and defect level. They are said to "dominate" \( \mathbf{g} \).

**Definition 4.2**

**Dominance.** A vector in the objective space \( \mathbf{a} \) dominates (\( \succeq \)) another vector \( \mathbf{g} \) iff \( \mathbf{a} \) is at least as good or better as \( \mathbf{g} \) in all objectives. For a target minimization function this translates to

\[ \mathbf{a} \succeq \mathbf{g} : f(\mathbf{a}) \leq f(\mathbf{g}) \ \forall i \in \{1, 2, \ldots, k\} : a_i \leq g_i. \quad (4.3) \]

In analogy, \( \mathbf{h, i, j} \) are inferior to \( \mathbf{g} \) (or are dominated) because they are inferior in both objectives cost and defect level. \( \mathbf{c, d, e, f} \) are said to be indifferent to \( \mathbf{g} \) because they are partially superior and partially inferior. For example, \( \mathbf{c, f} \) offer lower total cost, but at a higher defect level, and for \( \mathbf{c, d} \) it is vice versa.

**Definition 4.3**

**Indifference.** A vector in the objective space \( \mathbf{c} \) is indifferent (\( \sim \)) to another vector \( \mathbf{g} \) iff \( \mathbf{c} \) is in some criteria better than \( \mathbf{g} \) and worse in others. For a target minimization function this translates to

\[ \mathbf{c} \sim \mathbf{g} \iff f(\mathbf{c}) \not\leq f(\mathbf{g}) \land f(\mathbf{g}) \not\leq f(\mathbf{c}). \quad (4.4) \]
4.1. Introduction to Multi-Objective Optimization

Figure 4.1: Possible relations of solutions in the objective space, $g$ as reference (circled). Goal is minimization of both cost and defect level.

Taking all dominating points in an objective space as supporting points, we obtain a Pareto front. For the example from Fig. 4.1, the vectors $a, b, c, d, e, f$ form such a front (see Fig. 4.2).

**Definition 4.4**

**Globally optimal Pareto set or Pareto front.** A set $P$ of Pareto-optimal vectors in $X$ with $P \subset X$ is called globally optimal, iff

$$\forall a \in P : \exists x \in X : x \preceq a.$$

(4.5)

Figure 4.2: A set of dominating points form a Pareto front.

Using the Pareto ranking concept, our problem from Example 1.1 can be solved.

**Example 4.1** (cf. Example 1.1)

Among the components/subsystems that can tolerate only a very low number of defect parts shipped are electronics for physics experiments or space subsystems. Typically the defect level is in the range of ppm as these blocks are vital for system functionality or difficult and costly to replace. Of course, these systems
are also sensitive with respect to cost. For an electronic readout subsystem e.g., the question was whether the existing test concept would fulfill the customer quality specifications and how other test concepts (IddQ (quiescent leakage current) [113], enhanced final test) would perform [114].

Figure 4.3: Readout subsystem: The data arrives at 40MHz rate compressed and digitized. The data is first linearized (in the LIN stage) including an adder for offset correction and a multiplier for gain adjustment. Next, the data is stored in a pipeline stage (PIPE) during first-level trigger loop latency. Then, if triggered, it is written to a so-called derandomizer (DER) event buffer. A subsequent filter (LVL2) contains FIR filters and statistic operators. In the trigger path the data from different channels are first applied to an adder circuit (ADDER) and then to another filter ASIC (LVL1) which extracts the energy and bunch crossing information formatting the signal according to the trigger system requirements (trigger primitives). As the trigger path is more dedicated to a special experiment, the ADDER and LVL1 ASIC are located outside the MCM to increase reusability [114].

The readout subsystem is depicted in Fig. 4.3, and it consists of five LPD and one LVL2 custom ASICs. The subsystem is located in the data acquisition and trigger path in CERN’s Atlas experiment, its task is the preprocessing of event data and activation of the latter trigger stage in case of interest. For more detailed information see [103].

Example 4.1 (continued) Various options for additional test efforts (better test coverage for the LPD die and/or the LVL2 die due to IddQ, improved final test coverage) are available. But the additional effort comes with a considerable cost penalty. Numbers can be found in Table 4.1, where the increase of the test cost caused by higher test coverage is illustrated. The options above can also be combined, and an overview of the scenarios under investigation is given in Table 4.2. In addition to the four simple combinations of IddQ and enhanced test (A, C, D, F), IddQ for only the LPD was also considered (B, E).

Table 4.1: Cost and fault coverage data (cost in arbitrary units a.u.): the increase of fault coverage due to IddQ and extended test has a cost penalty.

<table>
<thead>
<tr>
<th>Fault coverage</th>
<th>Chip test cost w/ IddQ 99.7%</th>
<th>Final test cost extended 97%</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPD chip (yield 60%)</td>
<td>8 a.u.</td>
<td>9 a.u.</td>
</tr>
<tr>
<td>LVL2 chip (yield 75%)</td>
<td>8 a.u.</td>
<td>9 a.u.</td>
</tr>
<tr>
<td>Final test</td>
<td>10 a.u.</td>
<td>15 a.u.</td>
</tr>
</tbody>
</table>

A MOE model, shown in Fig. 4.4, has been developed for this problem. On the right hand side, the components LPD and LVL2 enter the manufacturing flow (IDs 0 and 9), are tested (ID4), eventually scrapped.
4.1. Introduction to Multi-Objective Optimization

Table 4.2: Test scenarios: a combination of the three options available.

<table>
<thead>
<tr>
<th></th>
<th>LPD</th>
<th>LVL2</th>
<th>Final Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>IddQ</td>
<td>normal</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>IddQ</td>
<td>normal</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>normal</td>
<td>extended</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>IddQ</td>
<td>normal</td>
<td>extended</td>
</tr>
<tr>
<td>F</td>
<td>IddQ</td>
<td>extended</td>
<td></td>
</tr>
</tbody>
</table>

(ID10) and mounted (ID1) onto a thinfilm substrate (produced in ID2). The chips are wire bonded, and the populated substrate is connected to a laminate carrier, acting as mechanical carrier and 2nd level BGA package. All secondary packaging activities (connection, housing) are concentrated in this block ID13. The final test is represented in ID6. Based on this model and the numbers in Table 4.1, the results for cost and defect level for the scenarios A to F have been extracted.

Figure 4.4: MOE model for the cases from Table 4.2; only the underlying fault coverages for LPD test (ID4), LVL2 test (ID10), and final test (ID6) change according to Table 4.1.

Example 4.1 (end) Fig. 4.5 presents the results for a given cost-test coverage relation, where scenarios A to E turn out to be Pareto points.

- Solution A is the status-quo point with the actual IC wafer level test coverage and final test coverage.
- Solutions B to E represent the respective tradeoff for a dedicated defect level. Solution F can be dropped because E gives a better solution in terms of cost.
- Proceeding from B to E, a defect level reduction can be achieved while increasing the total module cost.

Trading off the results we see that the simplest approach (extending the final test time and therefore test coverage) already gives a remarkable defect reduction. Introducing IddQ for the lower-yield LPD chip even provides better performance at minimal additional cost. The use of IddQ for the LVL2 chip improves the defect level only minimally, so this action could be left away to spare engineering resources to redesign the chip. Combining the strategies B and D to E gives again an improvement, reaching the minimum defect level border for the case.
Figure 4.5: Pareto results for Example 4.1: “A” represents the status-quo; all further measures undertaken drive down the defect level while increasing the cost. “F” offers no defect improvement over “E” and is not a Pareto point.

Now, setting a maximum acceptance value for cost and/or for test escapes, the final test setup can be chosen. Setting the defect level to 1000ppm and 230a.u., solutions B and C are eligible.

However, even for a medium-complex manufacturing process, the number of design options can easily extend the capability of manual optimization.

Example 4.2
Consider the example of a GPS subsystem fabrication process, similar to that used in Section 3.3. In this simplified example, only two IC components are required (one RF chip and one digital correlator). Still, we can influence the yield of the RF die (y1), the correlator die (y2), the repair process step (y3), and the substrate (y4). Together with the fault coverage of three test steps (incoming test RF die f1, incoming test correlator f2, final test f4), we have a total of seven parameters.

For each of the seven parameters \{y1,y2,y3,y4,f1,f2,f4\}, we have a set of eight applicable values, leading us to \(8^7 = 2097152\) possible combinations.

For the simple readout electronics case in Example 4.1 with a countable number of choices, the Pareto search could still be done manually by evaluating all possible combinations, but for such a high number of alternatives as in Example 4.2 a more structured search mechanism is required. This mechanism is described in the following section.

4.2. Evolutionary Algorithms for Multi-Objective Optimization Problems

Various solution methods are available for single/multi-objective optimization problems. Among such solution techniques are analytical methods, numerical methods, e.g. linear programming techniques [115], or heuristic approaches. Analytical solutions are preferable when the target function is two-fold differentiable and continuous, numerical methods require at least a closed formula to be approximated. Since neither a general target function nor a closed formula is available to describe the cost-quality objectives, we have to concentrate on the heuristic methods.
4.2. Evolutionary Algorithms for Multi-Objective Optimization Problems

Table 4.3: General scheme of SA-like heuristic optimization methods in pseudo code.

0 begin
1 generate a starting solution (stochastically)
2 repeat
3 generate another solution in the neighborhood
   of the previous solution
4 compare the target functional values
5 if the new solution is better or
   within an acceptable threshold
6 then accept,
7 otherwise reject it
8 reduce deterioration threshold
9 until stop criterion is fulfilled
10 print out results
11 end

Heuristics

A heuristic algorithm is a deterministic and often iterative method, targeting to find for a given problem within limited time one or more solutions that are as good as possible without guaranteeing to find a global optimal solution [111].

Typical representations of the heuristic search algorithms class are Simulated Annealing (SA), Hill Climbing (HC), Threshold Accepting (TA), etc. All these class members follow a scheme described in Table 4.3. Since these methods are tailored for single-optimization problems because they do not include a multi-objective ranking scheme (and since we have ruled out to transform the cost-quality trade-off into a single-objective problem), SA-like algorithms are not eligible. Instead, we move to Evolutionary Algorithms.

Evolutionary Algorithms

Evolutionary Algorithms (EA) are based on the Darwinist notion of a population development by means of variation and selection: Due to stochastic deviation of their composites (chromosomes or genes), individuals have a different degree of fitness with respect to their environment. When it comes to reproduction, usually the fitter individuals are preferred (selected). Nissen [111] divides Evolutionary Algorithms into four main categories:

- Evolution Strategies (ES) as developed by Schwefel/Rechenberg for optimization of a turbine body,
- Evolution Programming (EP) by Fogel, Owens, and Walsh, used in the search for finite state machines,
- Genetic Algorithms (GA) by Holland created for adaptive systems, also beyond biological applications, and
- Genetic Programming (GP) by Koza, as a subclass of GA, becoming ever more popular for developing computer source code.

The distinction between these categories is not easy, and borders become somewhat blurred. As we only focus on the composites of an individual, its genes, and there is no cohesive behavior of the solution population, our algorithm detailed below belongs to the Genetic Algorithm class.

1 In [116] Fogel suggests that evolutionary algorithms reject the notion that meaningful fitness can be assigned to individual genes in isolation and instead emphasize the total cohesive behavior of an individual or reproducing population. Selection is implemented as a culling force that eliminates relatively poor solutions rather than as a promoter that makes additional copies of above-average solutions.
4.2. Evolutionary Algorithms for Multi-Objective Optimization Problems

Table 4.4: General scheme of Evolutionary Algorithms in pseudo-code.

0 begin
1 initialize starting population set $S_{pop}$ the size $N_{pop}$
2 goto 7
3 repeat
4 select parents according to their fitness $F$
5 generate offspring (using crossover and/or mutation with probability $p_{cross}$ and $p_{mut}$)
6 generate new population from offspring (with/out parents)
7 assign fitness $F$ to population members
8 update Pareto set $S_{pareto}$ from $S_{pop}$
9 until max number generations $n_{maxgen}$ or stop criterion
10 print out result $S_{pareto,final}$
11 end

It is well beyond the aim of this work to review all basics and aspects of Evolutionary Algorithms. See [111, 116, 117, 118, 119] for further readings. We only present the general scheme of EAs, which is listed in pseudo-code in Table 4.4. The interaction of this algorithm with the MOE tool is depicted in Fig. 4.6.

In the following sections we will detail the specific encoding, variation, and selection criteria used in our search algorithm.

Encoding

One of the core parts in GAs is to find a suitable encoding of the problem. Our optimization problem belongs to the class of real parameter optimizations [120], i.e. we have a given number of parameters (e.g. $\{y_1,y_2,y_3,y_4,f_{c1},f_{c2},f_{c4}\}$) forming our decision vector according to Def 4.2. This decision vector is also called individual $i$ in the EA context, and each parameter can adopt either a value from a given range or from a set of values (see e.g. Equation 4.6).

$$y_i = \{real\_value_0, \ldots, real\_value_n\}$$

Example $\{0.50; 0.55; 0.61; 0.67; 0.72; 0.78; 0.84; 0.90\}$ (4.6)

In principle, two alternative codings are available: transformation of the real parameters to a bitstring representation or the use of real number representation, as e.g. suggested in [119] (list of real parameters). The advantage of the bitstring encoding is the availability of numerous recombination and mutation operators; the disadvantage could be the high degree of perturbation, when disrupting a multi-bit represented parameter by recombination [120]. Perturbation means that the recombined offspring will not inherit any features assigned with the disrupted (and therefore destroyed) parameter value (see Fig. 4.9 below).

Another concern with bitstring encoding is the disproportional search space coverage: in general, to effectively cover the decision space, each parameter value (and therefore each location in the space), should have an equal probability to be selected. But when considering the standard bit-inversion mutation operator, the principle above is violated. Take the example of a three-digit bitstring with the actual binary value of $\{101\}$: although this bitstring represents eight different bit combinations, each one encoding a real value, by one-bit inversion only the three binary numbers $\{001,110,111\}$ can be reached. Other combinations require two or three bits to be inverted and are therefore much less likely to be selected. Countermeasures can prevent this, but they make the mutation process more complex.

On the other hand, at least the recombination disruption can be seen as an effective means to extend the parameter space exploration by “jumping” into distant regions. In fact, this exploration capability is one of the advantages of EA approaches compared to SA-type ones, which could be trapped in local extrema more easily.

In our work we decided to use the bitstring representation, as it is the most flexible one, and the above mentioned drawbacks can be overcome by choosing suitable recombination/mutation operators.
Figure 4.6: Interaction between the EA algorithm and the MOE tool. First, a model-specific parameter set is encoded and initialized into a population \( S_{\text{pop,0}} \). Every individual is translated into a netdata-file to extract the objective values cost and defect level. Based on these values, fitness assignment is done and the Pareto set \( S_{\text{Pareto}} \) is updated. If the maximum number of generations has not been exceeded, a new generation is created by means of selection and replication, and the loop continues with the objective extraction.
4.2. Evolutionary Algorithms for Multi-Objective Optimization Problems

A standard GA framework [121] was adopted and we encoded the parameters under investigation as a one-chromosome individual. The chromosome is a chain of $n_{\text{param}}$ genes, coding the parameters \{par, \ldots, par_{n_{\text{param}}}\}. Each parameter or "gene" is coded as a bitstring with a chosen resolution $n_{\text{res}}$. An example for the model shown in Fig. 4.11 is given in Fig. 4.7. In this example a resolution of $n_{\text{res}} = 3$ bits has been selected, permitting $2^{3} = 8$ real values to be encoded.

<table>
<thead>
<tr>
<th>y1</th>
<th>y2</th>
<th>y3</th>
<th>y4</th>
<th>fc1</th>
<th>fc2</th>
<th>fc4</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>100</td>
<td>01</td>
<td>010</td>
</tr>
<tr>
<td>gene1</td>
<td>gene2</td>
<td>gene3</td>
<td>gene4</td>
<td>gene5</td>
<td>gene6</td>
<td>gene7</td>
</tr>
<tr>
<td>011</td>
<td>110</td>
<td>100</td>
<td>011</td>
<td>010</td>
<td>100</td>
<td>110</td>
</tr>
</tbody>
</table>

Figure 4.7: Coding the parameter set of Example 4.2 as a one-chromosome individual; parameters are lined in an arbitrary sequence; each parameter is coded by a 3-bit gene selecting a value from its value list as in Eq. 4.6 (bit values in third line chosen as example).

So, in the first step of the scheme in Table 4.4 a population $S_{\text{pop}}$ the size $N_{\text{pop}}$ of bitstrings the length $n_{\text{param}} \cdot n_{\text{res}}$ is generated and initialized randomly.

**Objective Extraction and Fitness Assignment**

After decoding a bitstring individual to a specific set of real parameter values, these parameter values are fed into a MOE <netdata>-file describing the manufacturing model under investigation. With the <netdata>-file, a MOE simulation run is started, and the objective results "cost" and "defect level" for the fed parameter combination are extracted and returned to the GA framework (see Fig. 4.6).

The fitness assessment is done using the Strength Pareto Evolutionary Algorithm (SPEA) [54]. SPEA has been compared to various other multi-objective EAs and has proven to outperform them [122].

SPEA uses the concept of elitism, where the best performing individuals (the actual Pareto points of every population) are stored in an external set $S_{\text{Pareto}}$, the size of $N_{\text{Pareto}}$. Now, aiming for fitness minimization, the fitness $F(i)$ of an individual $i \in S_{\text{Pareto}}$ is the ratio of individuals $j \in S_{\text{pop}}$ dominated by $i$ to the total number of individuals (Eq. 4.7) [54].

$$F(i) = \frac{\left| \{ j \mid i \preceq j \} \right|}{N_{\text{pop}} + 1} \quad \text{ (Fitness Pareto individuals)}$$

$$F(i) \in [0, 1] \text{ in } \mathbb{Q}.$$  

The fewer individuals there are in a certain niche, the better the fitness value. This strategy propels the search into less explored regions of the decision space.

On the other hand, the fitness $F(j)$ of an individual $j \in S_{\text{pop}}$ is the sum of the fitnesses of all Pareto individuals $i \in S_{\text{Pareto}}$ dominating it, plus one to make sure all ordinary individuals have a lower fitness than the Pareto individuals (Eq. 4.8).

$$F(j) = 1 + \sum_{i \mid i \preceq j} F(i) \quad \text{ (Fitness population individuals)}$$

$$F(j) \in [1, N_{\text{pop}}] \text{ in } \mathbb{Q}.$$  

A high number of Pareto individuals dominating this ordinary individual reduces its fitness value. The advantage of this approach is to better maintain the diversity of the population. An example for the fitness calculation using the objective vectors $a\cdot j$ from Fig. 4.1 is given in Fig. 4.8.

After the fitness assignment, the Pareto set $S_{\text{Pareto}}$ is updated, removing eventually now dominated individuals and adding new non-dominated ones. Once this step has been accomplished, one generation has been completed, and the loop in Fig. 4.4 can start again till the until-condition is fulfilled.
4.2. Evolutionary Algorithms for Multi-Objective Optimization Problems

Figure 4.8: Fitness calculation of SPEA: the lower the value, the better the fitness $F()$. Pareto points dominating more population individuals, e.g. a, b get a worse fitness value than those covering a less explored region, such as d, f. From the normal population, individuals (k, m) close to those marginal points are promoted in order to support them “boldly to go where no individual has gone before”.

<table>
<thead>
<tr>
<th>Crossover CO1</th>
<th>Crossover CO2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent i</td>
<td>0 1 1 1 1 0 1 0 0 1 1 0 1 0 1 0 0 1 1 0</td>
</tr>
<tr>
<td>Parent j</td>
<td>0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 1 0</td>
</tr>
<tr>
<td>Child m</td>
<td>0 1 1 1 1 0 0 1 1 0 0 0 1 1 1 0 1 1 0</td>
</tr>
<tr>
<td>Child n</td>
<td>0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 1 1 0</td>
</tr>
</tbody>
</table>

Figure 4.9: Child m inherits its first and third part from parent i, the second part from parent j, child n vice versa. Crossover CO1 is performed along the gene boundaries (\(\mid\)), crossover CO2 breaks a gene apart.

Creating the Offspring: Selection and Replication

After updating the Pareto set and normal population, two individuals $i, j \in S_{pop} \cap S_{Pareto}$ are arbitrarily drawn, and the individual with the better (here: smaller) fitness is copied to the subsequent population $S_{pop}$, until again $N_{pop}$ individuals are selected (“tournament selection”).

From $S_{pop}$ now, again in a random drawing two individuals $i, j$ are selected for recombination. These individuals $i, j$ act as “parents”, inheriting part of their genetic information to their “children” $m, n \in S_{pop}$. This process is called “crossover”, and typical representations are 1-point, 2-point, and n-point crossovers. The number denotes the total of “cuts” where the parents exchange genetic information. An example for a 2-point crossover is given in Fig. 4.9.

The two types of crossover exemplified in Fig. 4.9 – along gene boundaries (CO1), or breaking genes (CO2) – illustrate the options available. CO1 crossovers mimic the true real number encoding, as existing “building blocks” survive the crossover operation, thus inheriting their benefits to the offspring. Thus, “building blocks” of an individual having a high fitness are preserved and not disrupted. On the other hand, a greater degree of exploration comes with a CO2 crossover. Both types of crossovers have been tested in the algorithm, and the CO1 operator supporting the “building block” hypothesis has been found to outperform the CO2 operator. The crossover probability $p_{cross}$ selects the gene boundaries used for crossovers.
4.3. Application of the Evolutionary Algorithm

Whereas crossover can be seen as the “coarse” search algorithm mixing existing genes for fast search space exploration, mutation operates in the neighborhood of an existing solution. The purpose of this “fine” search algorithm is to introduce “new” (i.e. not yet present) bitstring combinations and therefore new gene sequences. These new sequences

- are the only means to generate new possible solutions at all (which is not possible with the non-disruptive CO1 crossover operator because it keeps the existing gene sequences and only changes their combination),
- therefore help maintaining population diversity,
- and search for improvements in the neighborhood.

Bit-oriented operators perform mutation by flipping bits within a gene with a certain probability \( p_{mut} \). As already mentioned above, the problem with this operator is that flipping most- or least-significant bits would have had very different impact in the resulting number, and some parameter values requiring multiple bit inversions would have a very low probability.

In order to cover the entire sub-space, instead of simple bit inversion a uniform probability distribution is used. When a gene of an individual \( S'_{pop} \) is chosen stochastically for mutation with the mutation probability \( p_{mut} \), a uniformly distributed random number between 0 and \( (2^{res} - 1) \) is generated, representing the list number of the real parameter, and converted to a bit string.

When this process is accomplished moving all individuals with or without mutation from \( S'_{pop} \) to \( S''_{pop} \), the loop in Fig. 4.6 will continue with extraction of objectives and fitness assignment as described above.

4.3. Application of the Evolutionary Algorithm

We will now revisit Example 4.2. Remember, in the presented example, only two IC components are mounted (one RF chip and one digital correlator).

Example 4.3 (cf. Example 4.2)

For small, hand-held, low-power GPS systems, a reduction of size, cost, and power consumption is mandatory. MCM technology can be used to achieve these sophisticated constraints by offering a higher packaging density and better interconnect solutions. A functional schematic of the GPS front end is shown in Fig. 4.10. The operation is roughly as follows: After external filtering, the GPS signal passes via a matched impedance line to a low-noise amplifier (LNA), and is band-pass (BP) filtered at 1.575GHz to reject the image frequency. Using a voltage controlled oscillator (VCO) to feed the input signal, the reference signal is downconverted via intermediate frequencies (IF) to the base band. After A/D conversion, the signal undergoes the selection in the correlator and the subsequent stages [123].

The correlator IC is rerouted on wafer level to transform a peripheral pad pitch of 100µm into an area array pad pitch of 350µm, suitable for flip chip mounting. Both chips are tested before mounting, with the possibility to rework the correlator rerouting. These ICs are mounted onto a thinfilm substrate, which is in turn mounted to a laminate carrier. The module is then encapsuled and undergoes a final functional test.

Example 4.3 (continued) In Fig. 4.11 the MOE manufacturing model of this GPS front end is depicted. Components are an RF die (upper left, ID0) and a correlator die (medium left, ID13). The RF die with yield \( y_1 \) undergoes a pre-screening (ID3) with fault coverage \( fC1 \); the correlator \( y(2) \) is rerouted (ID14) for flip chip attach and afterwards optically inspected \( fC2 \), ID15). In case of an error, this chip can be repaired once with the success rate \( y_3 \) (ID12). Both dies are attached (ID5) onto the thinfilm substrate (upper right corner with \( y_4 \), ID1), and then the entire system undergoes a functional test before shipping \( fC4 \), ID9).

The values chosen for the parameters \( \{y_1, y_2, y_3, y_4, fC1, fC2, fC4\} \) in Example 4.3 are shown in Table 4.5. Additionally, when changing a yield or fault coverage parameter, dependent parameters, such as component cost or test cost, have to be altered as well. Thus, for each parameter in \( \{y_1, y_2, y_3, y_4, fC1, fC2, fC4\} \) there is a dependent parameter following the syntax <mainPar.dependentPar>.

In order to compare the EA search approach to “benchmark solutions”, we also calculated the results of three typical parameter combinations, marked in Table 4.6:
Figure 4.10: The GPS demonstrator: After external filtering, the GPS signal passes via a matched impedance line to a LNA, and is BP-filtered at 1.575GHz to reject the image frequency. Using a VCO to feed the internal reference, the signal is downconverted via intermediate frequencies to the base band. After A/D conversion, the signal undergoes the selection in the correlator and the subsequent stages. Pictures courtesy SUMMIT consortium [123].
4.3. Application of the Evolutionary Algorithm

Table 4.5: Parameter values used in Example 4.3. For each main parameter there is a dependent parameter detailing the cost relation when changing yield and fault coverage (values arbitrarily chosen).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Yield and fault coverage</th>
<th>Low</th>
<th>Average</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>y1</td>
<td>0.5</td>
<td>0.56857</td>
<td>0.63428</td>
<td>0.71428</td>
</tr>
<tr>
<td>y2</td>
<td>0.5</td>
<td>0.56857</td>
<td>0.63428</td>
<td>0.71428</td>
</tr>
<tr>
<td>y3</td>
<td>0.67142</td>
<td>0.70571</td>
<td>0.74286</td>
<td>0.78571</td>
</tr>
<tr>
<td>y4</td>
<td>0.67142</td>
<td>0.70571</td>
<td>0.74286</td>
<td>0.78571</td>
</tr>
<tr>
<td>fc1</td>
<td>0.5</td>
<td>0.56857</td>
<td>0.63428</td>
<td>0.71428</td>
</tr>
<tr>
<td>fc2</td>
<td>0.5</td>
<td>0.56857</td>
<td>0.63428</td>
<td>0.71428</td>
</tr>
<tr>
<td>fc3</td>
<td>0.67142</td>
<td>0.70571</td>
<td>0.74286</td>
<td>0.78571</td>
</tr>
<tr>
<td>fc4</td>
<td>0.67142</td>
<td>0.70571</td>
<td>0.74286</td>
<td>0.78571</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Yield and test cost in [s/u]</th>
<th>Low</th>
<th>Average</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>y1_cost</td>
<td>50</td>
<td>35.5714</td>
<td>25.6142</td>
<td>17.2857</td>
</tr>
<tr>
<td>y2_cost</td>
<td>50</td>
<td>35.5714</td>
<td>25.6142</td>
<td>17.2857</td>
</tr>
<tr>
<td>y3_cost</td>
<td>100</td>
<td>61.4286</td>
<td>67.1428</td>
<td>72.8571</td>
</tr>
<tr>
<td>y4_cost</td>
<td>100</td>
<td>61.4286</td>
<td>67.1428</td>
<td>72.8571</td>
</tr>
<tr>
<td>fc1_cost</td>
<td>50</td>
<td>35.5714</td>
<td>25.6142</td>
<td>17.2857</td>
</tr>
<tr>
<td>fc2_cost</td>
<td>50</td>
<td>35.5714</td>
<td>25.6142</td>
<td>17.2857</td>
</tr>
<tr>
<td>fc3_cost</td>
<td>100</td>
<td>61.4286</td>
<td>67.1428</td>
<td>72.8571</td>
</tr>
<tr>
<td>fc4_cost</td>
<td>100</td>
<td>61.4286</td>
<td>67.1428</td>
<td>72.8571</td>
</tr>
</tbody>
</table>

The “low” combination corresponds to low testing efforts and low-cost low-yield components, giving the highest overall defect level, possibly at the lowest cost (we will cope with that point later on). On the other hand, the “high” value represents the opposite with maximum test effort and high-quality components, yielding the lowest defect level. The “low” and “high” combinations are two often undesired extremes when it comes to cost effectiveness and quality standards. Thus, we included a combination of “average” parameter values well, picking a medium value from every parameter range.

Example 4.3 (continued) The parameter set \([y_1, y_2, y_3, y_4, fc_1, fc_2, fc_3]\) is encoded as a bitstring individual (cf. Fig. 4.7), and the search algorithm is conducted following the EA scheme in Table 4.4. The metrics “cost” and “defect level” are computed by inserting the parameter values decoded from the bitstring individuals into the MOE model shown in Fig. 4.11. From these metrics, the fitness assessment and the evolutionary process as described in the previous sections are performed for a specified number of generations (see Fig. 4.6).

Fig. 4.12 shows the joint results from three runs. As one can see, the high-average-low results are situated almost on a straight line moving towards the process optimum. The “low”-value (filled circle) is
4.4. Discussion

Table 4.6: Parameter values from Table 4.5: Benchmark combinations “low”, “average”, and “high” marked.

<table>
<thead>
<tr>
<th></th>
<th>LOW</th>
<th>AVERAGE</th>
<th>HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>y1</td>
<td>0.5</td>
<td>0.55714</td>
<td>0.61428</td>
</tr>
<tr>
<td>y2</td>
<td>0.5</td>
<td>0.56857</td>
<td>0.63714</td>
</tr>
<tr>
<td>y3</td>
<td>0.875</td>
<td>0.73064</td>
<td>0.77426</td>
</tr>
<tr>
<td>y4</td>
<td>0.875</td>
<td>0.73064</td>
<td>0.77426</td>
</tr>
<tr>
<td>f1</td>
<td>0.5</td>
<td>0.55714</td>
<td>0.61428</td>
</tr>
<tr>
<td>f2</td>
<td>0.5</td>
<td>0.56857</td>
<td>0.63714</td>
</tr>
<tr>
<td>f4</td>
<td>0.875</td>
<td>0.73064</td>
<td>0.77426</td>
</tr>
</tbody>
</table>

Table 4.7: Decoded parameters for the Pareto front; trade-off point is marked.

<table>
<thead>
<tr>
<th></th>
<th>y1</th>
<th>y2</th>
<th>y3</th>
<th>y4</th>
<th>f1</th>
<th>f2</th>
<th>f4</th>
<th>cost (a.u.)</th>
<th>defect level DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>y1</td>
<td>0.9</td>
<td>0.82</td>
<td>0.99</td>
<td>0.0</td>
<td>0.5</td>
<td>0.69</td>
<td>0.69</td>
<td>8998.86</td>
<td>0.0178</td>
</tr>
<tr>
<td>y2</td>
<td>0.9</td>
<td>0.73</td>
<td>0.99</td>
<td>0.0</td>
<td>0.91</td>
<td>0.69</td>
<td>0.69</td>
<td>9034.68</td>
<td>0.0086</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.69</td>
<td>0.99</td>
<td>0.0</td>
<td>0.91</td>
<td>0.69</td>
<td>0.69</td>
<td>9038.93</td>
<td>0.0084</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.69</td>
<td>0.99</td>
<td>0.0</td>
<td>0.98</td>
<td>0.69</td>
<td>0.69</td>
<td>9012.74</td>
<td>0.0006</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.69</td>
<td>0.99</td>
<td>0.0</td>
<td>0.98</td>
<td>0.73</td>
<td>0.73</td>
<td>9064.02</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.69</td>
<td>0.99</td>
<td>0.0</td>
<td>0.98</td>
<td>0.77</td>
<td>0.77</td>
<td>9113.58</td>
<td>0.0044</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.69</td>
<td>0.99</td>
<td>0.0</td>
<td>0.98</td>
<td>0.82</td>
<td>0.82</td>
<td>9166.78</td>
<td>0.0034</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.69</td>
<td>0.99</td>
<td>0.0</td>
<td>0.98</td>
<td>0.95</td>
<td>0.95</td>
<td>9318.07</td>
<td>0.0014</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.69</td>
<td>0.99</td>
<td>0.0</td>
<td>0.98</td>
<td>0.99</td>
<td>0.99</td>
<td>9367.91</td>
<td>0.0008</td>
</tr>
<tr>
<td></td>
<td>0.84</td>
<td>0.69</td>
<td>0.99</td>
<td>0.0</td>
<td>0.98</td>
<td>0.99</td>
<td>0.99</td>
<td>9518.82</td>
<td>0.0002</td>
</tr>
</tbody>
</table>

dominated by the average values from Table 4.5 (outline square), and both are surpassed by the “high” parameters (filled square).

Albeit the “high” combination gives the lowest defect level (while the “low” combination gives the highest one), lower cost is obtainable at the price of a small increase in the defect level.

The Pareto front (small filled squares) is located in the lower left area of Fig. 4.12(a), and a zoom of this area is shown in Fig. 4.12(b). We pretend that all Pareto points fulfill the general quality requirements, and choose the point {0.912 (a.u., 5800 ppm)} as trade-off point, marking it with a circle. Thus, compared to the “high” point with a cost of 10’500 a.u. and a defect level < 1000 ppm, we achieve a 16% cost reduction.

In a final step we have to analyze and to interpret the results from the genetic optimization run. Table 4.7 shows the decoded parameters of the Pareto front.

**Example 4.3 (end)** From Table 4.7 we can see that high yield values for the RF die (y1) and the correlator IC (y2) are mandatory. When looking back on Table 4.5 and comparing the cost penalty for yield and fault coverage increase, the benefits are best for these two components. The influence of the rework step (y3) is gradually turned down while the final fault coverage fC increases, although the cost penalty is four times higher when increasing fC. Obviously, the high component yield together with a good test renders it superfluous to do some repair and to invest in retesting the repaired component. High substrate yield (y4) is also imperative. The final test efficiency increases permanently, driving the defect level down the Pareto front. An interesting point is that in the higher defect level regions the low fC value is first compensated by increasing the preliminary test fC due to its better cost-performance ratio. Then the final test comes into play. Thus for some applications, where only minor defect level improvements are required, a preliminary test might already suffice to move to an acceptable defect level.

4.4. Discussion

Does the evolutionary algorithm always result in a Pareto front like the one depicted in Fig. 4.12? And what about the stability of the trade-off point? We will investigate these two issues in the remainder of this
4.4. Discussion

(a) The Pareto front clearly outperforms the average parameter combination (EA parameters are population size $N_{\text{pop}} = 100$, Pareto set size $N_{\text{Pareto}} = 10$, $n_{\text{maxgen}} = 100$ iterations, crossover probability $p_{\text{cross}} = 0.8$, and mutation probability $p_{\text{mut}} = 0.1$).

(b) Zoom of (a); trade-off point marked with a circle.

Figure 4.12: Results for Example 4.3: comparison of the Pareto front to high-average-low parameter combinations; the defect level is to range between 0 and 1.
section.

Pareto Types and Population Shapes

For another optimization problem, a Pareto front like the one depicted in Fig. 4.13(a) has been observed. Thus, the question is

- how many different Pareto front types do exist,
- whether we can deduce information on the Pareto front type before using the EA optimization method, and
- whether this can help reduce the computational effort.

In order to investigate the general result types for an EA run, we introduce the shape of the population as criterion.

We define the population shape as two tangential lines including all population individuals and originating near or on the trade-off point. The Pareto front from the example in Fig. 4.13(b) is shown again in Fig. 4.13(b), with the population shape marked. In this figure, the Pareto front is identical with the population shape.

For a more general perception, two angles $\alpha, \beta$ are introduced to describe the inclination of the population shape (see Fig. 4.14(a)). Both angles in Fig. 4.14(a) are considered positive, and thus four general types of fronts can be distinguished:

- $\alpha > 0, \beta > 0$: a “large” Pareto front (Fig. 4.14(a)),
- $\alpha > 0, \beta < 0$: mini-Pareto type 1, where only the lower part of the population shape constitutes the Pareto front (Fig. 4.14(b)),
- $\alpha < 0, \beta > 0$: mini-Pareto type 2, where only the upper part of the population shape constitutes the front (Fig. 4.14(c)), and
- $\alpha < 0, \beta < 0$: only a single optimal point (Fig. 4.14(d)).

Pareto fronts like in Figs. 4.14(a) and 4.14(b) have already been shown previously. The only hint about the Pareto type we can have before calculating the front are the “high-parameter” and “low-parameter” points. As already said, the “low-parameter” point always has the highest defect level
4.4. Discussion

Figure 4.14: Types of Pareto fronts: dotted lines indicate the shape of the population, solid lines the Pareto front. In a) the Pareto front and the population fall together, in the other cases only part of this shape builds the Pareto front. In d) there is even only a single optimal point. From the location of the high-low points no conclusion on the type of Pareto front can be drawn.
(since every change leads to higher yield and better fault coverage and therefore to a decreasing defect level), and the "high-parameter" point has the lowest defect level (as every change leads to a degradation). On the other hand, no information can be deducted on the cost of these two points. Investment into higher yield and better fault coverage increases the direct cost, but this could be compensated by a reduced yield loss, depending on the actual parameter values. So, the "low" point can exhibit an even higher total cost than the "high" point (Fig. 4.12(a)) or not (Fig. 4.13(a)). Since the "low-high" pattern of the "large Pareto front" and the "mini-Pareto 2" and the pattern of the "single optimal point" and "mini-Pareto 1" can be the same (Fig. 4.14), no rash conclusion about the Pareto type can be drawn.

**Stability Analysis**

Another concern is the stability of the trade-off point (marked with circles in Figs. 4.12 and 4.13(a)).

**Definition 4.5**

**Stability of Trade-off Point.** The stability of a cost-defect level trade-off point, which is located at \(\{c_{\text{trade-off}}, D_{\text{trade-off}}\}\) in the objective space, is defined as variation in the objective space \(\{c_{\text{trade-off} ± Δc}; D_{\text{trade-off} ± ΔD}\}\) due to parameter variations in the decision space \(\{y_{1}\text{trade-off} ± Δy_{1}, y_{2}\text{trade-off} ± Δy_{2}, \ldots, y_{4}\text{trade-off} ± Δy_{4}\}\}. The higher \(Δc \cdot ΔD\), the worse the stability.

In real-world optimization problems using statistical parameters in the decision vector, it is very probable that some parameters have a higher statistical variance than others and that therefore some trade-off points are more stable than others. In order to quantify the susceptibility to this parameter variation, we perform a stability analysis.

The usual procedure would be to calculate the gradient of every single parameter. But due to the large number of parameters, the result would be difficult to analyze, and moreover, the parameters are not independent of each other. Instead, we move to a "nearest neighbor" analysis, where we take a subpopulation consisting of parameters adjacent to those of the trade-off point (see Table 4.8). The assumption is that every parameter of the trade-off point could only change to its adjacent higher or lower value. So taking Table 4.5 as an example, the value of \(y_{1} = 0.90\) could only move to 0.842858. In our example, only 7\(^{2}\) individuals (= parameter combinations) had to be evaluated. For a higher number, a joint Pareto/reverse Pareto search\(^{2}\) would give the shape of the stability subpopulation.

**Table 4.8: Nearest neighbor population for the trade-off point in Example 4.3: the parameter values of the trade-off point marked with thick boxes, neighbors with thin boxes.**

<table>
<thead>
<tr>
<th></th>
<th>y1</th>
<th>0.5</th>
<th>0.55714</th>
<th>0.61428</th>
<th>0.67142</th>
<th>0.72857</th>
<th>0.78571</th>
<th>0.842858</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>y2</td>
<td>0.5</td>
<td>0.56857</td>
<td>0.63714</td>
<td>0.70571</td>
<td>0.77428</td>
<td>0.842858</td>
<td>0.842858</td>
<td>0.842858</td>
<td>0.9</td>
</tr>
<tr>
<td>y3</td>
<td>0.68702</td>
<td>0.73064</td>
<td>0.77426</td>
<td>0.81788</td>
<td>0.86150</td>
<td>0.8926</td>
<td>0.8926</td>
<td>0.8926</td>
<td>0.9</td>
</tr>
<tr>
<td>y4</td>
<td>0.68702</td>
<td>0.73064</td>
<td>0.77426</td>
<td>0.81788</td>
<td>0.86150</td>
<td>0.8926</td>
<td>0.8926</td>
<td>0.8926</td>
<td>0.9</td>
</tr>
<tr>
<td>c1</td>
<td>0.5</td>
<td>0.55714</td>
<td>0.61428</td>
<td>0.67142</td>
<td>0.72857</td>
<td>0.842858</td>
<td>0.842858</td>
<td>0.842858</td>
<td>0.9</td>
</tr>
<tr>
<td>c2</td>
<td>0.5</td>
<td>0.56857</td>
<td>0.63714</td>
<td>0.70571</td>
<td>0.77428</td>
<td>0.842858</td>
<td>0.842858</td>
<td>0.842858</td>
<td>0.9</td>
</tr>
<tr>
<td>c4</td>
<td>0.68702</td>
<td>0.73064</td>
<td>0.77426</td>
<td>0.81788</td>
<td>0.86150</td>
<td>0.8926</td>
<td>0.8926</td>
<td>0.8926</td>
<td>0.9</td>
</tr>
</tbody>
</table>

We then introduce the rectangle enveloping the subpopulation as stability criterion; the larger the size of this rectangle (i.e., \(Δc \cdot ΔD\)), the less stable the point is. For a three-dimensional objective space, this criterion would be extended to a box containing all subpopulation members.

The analysis is exemplified using the trade-off point from Example 4.3. This point is compared to the the two neighbors points \(T1, T2\) on the Pareto front (see Table 4.9).

The results of this stability analysis are given in Fig. 4.15. Comparing the enveloping rectangles, the one of the originally chosen trade-off point presents the smallest area, giving the best stability of all candidates.

**Numerical Resolution**

The last issue to be tackled is the numerical resolution of the computations. When analyzing the Pareto points in Table 4.7, we see that for the second and third points, although \(y3\) is reduced, an improvement in

\(^{2}\) highest cost; highest defect level
Figure 4.15: Comparison of the stability of possible trade-off points (indicated by circles); the values of the envelope corner points are shown, demonstrating that the chosen trade-off point b) exhibits the lowest stability risk.
Table 4.9: Possible trade-off points for stability analysis (cf. Table 4.7).

<table>
<thead>
<tr>
<th></th>
<th>y1</th>
<th>y2</th>
<th>y3</th>
<th>y4</th>
<th>fc1</th>
<th>fc2</th>
<th>fc4</th>
<th>cost</th>
<th>defect level</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0.90</td>
<td>0.98</td>
<td>0.69</td>
<td>0.99</td>
<td>0.90</td>
<td>0.91</td>
<td>0.69</td>
<td>0.95</td>
<td>0.0084</td>
</tr>
<tr>
<td>Trade Off</td>
<td>0.90</td>
<td>0.98</td>
<td>0.69</td>
<td>0.99</td>
<td>0.90</td>
<td>0.98</td>
<td>0.69</td>
<td>0.91</td>
<td>0.0058</td>
</tr>
<tr>
<td>T2</td>
<td>0.90</td>
<td>0.98</td>
<td>0.69</td>
<td>0.99</td>
<td>0.90</td>
<td>0.98</td>
<td>0.73</td>
<td>0.96</td>
<td>0.005</td>
</tr>
</tbody>
</table>

The defect level can be observed. This effect can also be noticed in the same table for the last and the last but one Pareto point: \( y_1 \) reduces from 0.99 to 0.84, but the overall defect level improves. The reason is that in the forth position of the defect level the numerical noise limits are reached (all input parameters are only specified to the sixth decimal position). For further increase in numerical resolution, the decimal places of the input parameters would have to be increased.

4.5. Summary

In this chapter we have presented an introduction to multi-objective optimization and motivated why this type of optimization is preferable for design space exploration. The Pareto ranking scheme is used for result representation.

Whereas for a small search set the quest for these trade-off points can be done manually, for larger sets a structured algorithm is required. Evolutionary Algorithms have been chosen as an effective means to conduct a search for Pareto optimal solutions in a large decision space for a manufacturing setup, and we have shown the successful adaptation of such an algorithm to our cost-quality problem.

The application of this search algorithm to a GPS production has been used to illustrate the procedure to select a trade-off point, the foremost target of the virtual prototyping. An important point is the post-processing and interpretation of the decoded results. The discussion on Pareto front types showed that the result type cannot be anticipated. Stability issues for trade-off points have been tackled and a figure-of-merit has been introduced to assess different candidates.

In the following chapter we will now apply our methodology to the manufacturing optimization of a Smart Card.
In this chapter we apply our cost-quality trade-off methodology to the manufacturing process of an e-business smart card, in the following referred to as eCard. The purpose of this application example is to illustrate data collection and identification of available optimization options early in the design, in contrast to the previous GPS example, where theoretical incremental improvements of cost and yield/fault coverage in an existing production have been assumed. Also, this time the considerations include rising NRE costs, i.e. additional equipment investment required to achieve a higher yield or fault coverage.

The chapter starts with an introduction to the eCard, the following section presents the manufacturing model used. Based on a stability analysis, optimization potentials are identified, analyzed, and finally discussed.

Note: Since the work underlying this chapter has been done under a non-disclosure agreement for an external partner, all cost data is anonymized as a.u. (arbitrary cost units).

5.1. Introduction to the Case Study

Defined at its highest level, a smart card, as it currently exists, is a credit-card sized plastic card with an embedded computer chip. The chip can either be a microprocessor with internal memory or a memory chip with non-programmable logic. The chip connection is either via direct physical contact or remotely via a contactless electro-magnetic interface. A microprocessor chip can add, delete and otherwise manipulate information in its memory. It can be viewed “as a miniature computer with an I/O port, operating system and hard disk” [124].

Nowadays, even more capabilities than only data processing can be made available to the customer in order to facilitate e-business and to make new applications happen. But these capabilities require additional services. Therefore, the eCard with its enhanced functionality has been developed, comprising

- location awareness provided by a GPS system, and
- authentication by means of an automatic fingerprint identification system (AFIS).

A schematic picture of a standard so-called ISO ID1 card (according to ISO 7810) with a thickness of 0.76mm ± 0.08mm, is shown in Fig. 5.1 [125]. The location of the functional blocks is indicated.

The RF GPS block incorporates the RF chip GRF (including mixer and VCO for downconverting the incoming signal), a low noise amplifier (LNA), an external clock plus passive components. The antenna is
produced separately and inserted during assembly. The digital GPS part consists of the correlator chip GSP, a microcontroller µC, a Flash EEPROM and an SRAM for program storage and memory. The functional block “Fingerprint” comprises the sensor itself, plus an ARM7 microprocessor, and again program storage and memory.

Technological Issues

So far existing Smart Cards included only single chip applications, mostly an 8-bit µC with additional security features [125]. This µC is inserted into empty space on the top side of the card body, as submodule in either chip-on-flex, TAB, or lead-frame technology (see Fig. 5.2). The process is delicate in terms of process tolerances and yield, but well-established and under control.

For the eCard, the manufacturing complexity is significantly increased from inserting one IC to seven ICs, rendering the chip-on-flex approach improbable. The multiple dies require the existence of a carrier substrate in order to benefit as much as possible from a standard electronic system assembly process. Any additional handling and processing steps will increase system cost due to experimental setups required, followed by a lengthy qualification procedure.

Since, to the best of our knowledge, a system with this complexity has not been integrated so far into a credit card using card manufacturing processes such as moulding and lamination\(^1\) [126], the developers are

\(^1\)Currently, there do exist “cards” that are enclosed in a metal housing and are several millimeters thick.
interested in an assessment of the cost structure, which is presented in the remainder of this chapter.

5.2. Optimizing eCard

First, cost/yield information for medium quantities for the targeted manufacturing process has to be acquired. Based on these numbers, a primary analysis must be conducted to determine the cost/quality range. In case cost/quality requirements are not met, improvement strategies have to be developed and assessed.

The eCard Manufacturing Model

The eCard requires several IC components to be interconnected.

- **GPS**: signal processor GSP, RF chip GRF, µC, SRAM, Flash, antenna, passive components;
- **Fingerprint**: sensor, µC, including Flash and SRAM.

For the fabrication, a sequential approach has been chosen, where first all electronic components are mounted onto an ultra-thin substrate using standard electronic assembly processes (chip attach, wire bonding, SMD placing, soldering). Onto this populated substrate the antenna is attached and the primary software initialization is done. Then, the substrate surface is brought to an even level by means of injection moulding, and finally the top and bottom foils are laminated (see Fig. 5.3). Especially the lamination process is reported to be very cumbersome and can have a yield of down to 60% [126].

Fig. 5.4 shows the manufacturing model for the eCard. For the functional prototype, both wire bonding of bare dies and soldering of SMD passive components and CSPs are used. First, in ID0 the board is produced, undergoes a short/open test, and is prepared in ID3 for soldering (solder screen printing). Then, the solderable IC components are attached to the board, followed by a cleaning step (not shown). The bare die components (in ID4, ID5, and ID6), either tested (TEST A & B) or already delivered as KGDs, are now wire bonded to the partially populated board. TEST C is the first opportunity to inspect the module, primarily to check and repair the wire bonds and solder joints (REWORK C). The bare die components are then glob topped for protection.

Meanwhile, the antenna has been produced (ID20) and tested (TEST D). The antenna is now mounted onto the board and its feed connected to the module. Subsequently, the fingerprint is attached to the build-up. TEST and REWORK E provide a functional test and the last opportunity to replace circuitry before the non-reworkable card processes start.

First, the build up is injection moulded to provide an even level on the top. Then, lamination of the bottom and top foil (partially with openings) finishes the manufacturing process. TEST F is the final factory test before the eCard is shipped for personification at a card issuer company.
5.2. Optimizing eCard

Table 5.1: Production data for the eCard, 10'000 units p.a.; components/processes with yield ranges are marked with thick boxes. Cost is denoted in arbitrary units (a.u.) due to confidentiality reasons.

<table>
<thead>
<tr>
<th>Process</th>
<th>MOE</th>
<th>Yield (%)</th>
<th>FC (%)</th>
<th>Cost [a.u.]</th>
<th>NRE [a.u.]</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Substrate</td>
<td>ID0</td>
<td>99</td>
<td>23</td>
<td>5'000</td>
<td></td>
<td>multi-project wafer</td>
</tr>
<tr>
<td>LNA</td>
<td>ID4</td>
<td>80-90</td>
<td>10</td>
<td>20'000</td>
<td></td>
<td>multi-project wafer</td>
</tr>
<tr>
<td>GRF</td>
<td>ID5</td>
<td>99</td>
<td>12</td>
<td>0</td>
<td></td>
<td>Bare Die</td>
</tr>
<tr>
<td>pC</td>
<td>ID6</td>
<td>99</td>
<td>12</td>
<td>10'000</td>
<td></td>
<td>Bare Die</td>
</tr>
<tr>
<td>Memories</td>
<td>ID12</td>
<td>99</td>
<td>30</td>
<td>0</td>
<td></td>
<td>Std CSFs</td>
</tr>
<tr>
<td>GSP</td>
<td>ID13</td>
<td>99.9</td>
<td>17</td>
<td>0</td>
<td></td>
<td>Custom CSP</td>
</tr>
<tr>
<td>ARM7</td>
<td>ID14</td>
<td>99</td>
<td>15</td>
<td>100'000</td>
<td></td>
<td>Custom CSP</td>
</tr>
<tr>
<td>Antenna (ANT)</td>
<td>ID20</td>
<td>70-85</td>
<td>16</td>
<td>15'000</td>
<td></td>
<td>no ESD protection</td>
</tr>
<tr>
<td>Fingerprint (FP)</td>
<td>ID24</td>
<td>80-90</td>
<td>19</td>
<td>0</td>
<td></td>
<td>multi-project wafer</td>
</tr>
<tr>
<td>S/O Test</td>
<td>ID3</td>
<td>90</td>
<td>8</td>
<td>10'000</td>
<td></td>
<td>DC test</td>
</tr>
<tr>
<td>TEST A</td>
<td>ID7</td>
<td>50</td>
<td>3</td>
<td>15'000</td>
<td></td>
<td>DC test</td>
</tr>
<tr>
<td>TEST B</td>
<td>ID8</td>
<td>50</td>
<td>3</td>
<td>15'000</td>
<td></td>
<td>DC test</td>
</tr>
<tr>
<td>TEST C</td>
<td>ID16</td>
<td>30</td>
<td>5</td>
<td>5'000</td>
<td></td>
<td>bond only</td>
</tr>
<tr>
<td>REWORK C</td>
<td>ID17</td>
<td>30</td>
<td>5</td>
<td>5'000</td>
<td></td>
<td>bond only</td>
</tr>
<tr>
<td>TEST D</td>
<td>ID21</td>
<td>40</td>
<td>7</td>
<td>15'000</td>
<td></td>
<td>return loss</td>
</tr>
<tr>
<td>TEST E</td>
<td>ID25</td>
<td>30</td>
<td>15</td>
<td>5'000</td>
<td></td>
<td>feed, FP</td>
</tr>
<tr>
<td>REWORK E</td>
<td>ID27</td>
<td>70</td>
<td>20</td>
<td>50'000</td>
<td></td>
<td>medium functional</td>
</tr>
<tr>
<td>TEST F</td>
<td>ID30</td>
<td>70</td>
<td>20</td>
<td>50'000</td>
<td></td>
<td>medium functional</td>
</tr>
<tr>
<td>Die Attach W/B</td>
<td>ID3</td>
<td>99.9-999</td>
<td>4</td>
<td>15'000</td>
<td></td>
<td>TII DNs</td>
</tr>
<tr>
<td>Soldering</td>
<td>ID11</td>
<td>99.9-999</td>
<td>3</td>
<td>25'000</td>
<td></td>
<td>ca. 20</td>
</tr>
<tr>
<td>Mount SMD</td>
<td>ID15</td>
<td>99.9</td>
<td>23</td>
<td>0</td>
<td></td>
<td>250 DNs</td>
</tr>
<tr>
<td>Glob Top</td>
<td>ID18</td>
<td>99.9</td>
<td>2</td>
<td>0</td>
<td></td>
<td>250 DNs</td>
</tr>
<tr>
<td>Mount Antenna (MNT)</td>
<td>ID23</td>
<td>99.9</td>
<td>5</td>
<td>7'000</td>
<td></td>
<td>250 DNs</td>
</tr>
<tr>
<td>SW Init</td>
<td>ID33</td>
<td>99.9</td>
<td>2</td>
<td>15'000</td>
<td></td>
<td>250 DNs</td>
</tr>
<tr>
<td>Injection Moulding (MLD)</td>
<td>ID28</td>
<td>80-85</td>
<td>6</td>
<td>60'000</td>
<td></td>
<td>250 DNs</td>
</tr>
<tr>
<td>Lamination T/B (LAM)</td>
<td>ID30</td>
<td>80-85</td>
<td>2</td>
<td>5'000</td>
<td></td>
<td>250 DNs</td>
</tr>
</tbody>
</table>

Production Data

Table 5.1 presents the basic data for the eCard manufacturing flow. There are seven components/process steps where we had to estimate the yield data based on prototyping experience and existing products resulting in data ranges, namely for the components LNA, GRF, Antenna, and Fingerprint, and for the processes Mount Antenna, Injection Moulding, and Lamination. These yield ranges are marked with boxes. Due to these ranges, the result will also have a tolerance range in terms of cost and defect level, posing an economic risk to the company selling this card.

Risk Analysis

In order to assess this risk, we need to perform an analysis estimating the cost-quality tolerance ranges, similar to the stability analysis in the previous chapter. The risk analysis gives an idea of how cost and quality of the eCard prototype will change due to the yield variations in Table 5.1. To do so, we sampled every yield range with its minimum, average, and maximum value (e.g. for the LNA 80%, 85%, 90%) and calculated the worst and the best case with minimum and maximum yield for each parameter. For illustration purposes, we also generated 200 yield combinations for the above mentioned components and processes, being a 10% subset of the $3^n = 2187$ possible combinations resulting from three applicable values for seven parameters \{LNA, GRF, ANT, FP, MNT, MLD, LAM\}.

In Fig. 5.5 we find the results of this risk analysis. In the x-direction the total module cost is shown (all cost in arbitrary cost units a.u.), in the y-direction the card defect level. The samples build a straight line, ranging from (375.85 a.u.; 0.2589) (best case) to (521.59 a.u.; 0.712) (worst case) with a medium of (449.05 a.u.; 0.4921). This straight line is caused by the yield loss; for lower yield figures more rejects are produced by TEST F, increasing the overall cost, and more faulty modules escape the final test, thus increasing the defect level.

From this data we can see that the yield ranges from Table 5.1 result in a significant span of cost and quality, and that with the actual test measures only an intolerable defect level is achievable. In the following section possible improvement concepts are outlined.
Figure 5.4: NOE manufacturing model for the eCard. In I10 (top left) the board is produced, tested, and prepared for soldering (ID3 and ID15). Now, the bare die components (in ID4, ID5, and ID6, left) are wirebonded to the partially populated board. TESTC is the first opportunity to inspect the module, primarily to check and repair wirebonds and solder joints. The bare die components are then gold leaded for protection purposes. Then, the antenna is mounted onto the board and its feed is connected to the module (ID23). Subsequently, the fingerprint plus a mechanical support is attached to the build-up. TEST and REWORK C provide a functional test connected to the module (ID23). Now, the baredie components are wirebonded to the partially populated board. TESTC is the first opportunity to inspect the module, primarily to check and repair wirebonds and solder joints. Then, the fingerprint plus a mechanical support is attached to the build-up. TEST and REWORK C provide a functional test connected to the module (ID23).
5.2. Optimizing eCard

In order to drastically improve the defect level of the prototype eCard, either the yield of the processes or the fault coverage of the test steps have to be improved. Moreover, a test’s/rework’s effectiveness always has to be checked. Thus, for the tests/rework points A to E omitting the step is an option. An exception is the final TEST F which cannot be omitted. Below all improvement options are listed.

**TEST A (LNA):** Currently, only a wafer-level direct current (DC) test is assumed to check the LNA’s functionality. Performance test would require a temporary carrier and RF measurement equipment to achieve a better fault coverage.

**TEST B (GRF):** The GRF chip is currently delivered untested from the supplier, but also here a simple DC test could act as an incoming inspection. For a full performance test passive components and a qualified antenna are required, either on the partially assembled board or on a temporary carrier, rendering this alternative rather expensive.

**TEST C, REWORK C (RF part GPS):** Here, an initial test of the GPS radio part is possible. Connectivity and performance with a qualified EXTERNAL antenna could be tested. Either a simple or extended RF test is applicable, using identical equipment, but differing in test time (and therefore test cost). For the rework process, the first stage is to check/replace the bond interconnects. The second repair stage also involves component replacement, e.g. the GPS oscillator.

**TEST D (Antenna):** In the first pass test the return loss can be measured; advanced testing requires dedicated RF equipment.

**TEST E, REWORK E:** The fingerprint IC is functionally tested before shipping, but since no ESD protection is present, there might be some pixel malfunctioning, calling for an incoming inspection. But not all 256x256 pixels are required for the fingerprint acquisition, so to lower the component cost, a fingerprint chip without all pixels functional might be applicable. Also, a connectivity test via scan bus or BIST for the digital part can be included, as well as a functional test of the GPS system together with the antenna to check the feed connection. A more rigid test incorporates at-speed and performance tests. This is the last possibility to identify and replace a malfunctioning component.
5.3. Results and Discussion

Rework only replaces/reworks the fingerprint and/or antenna feed; a step beyond would require failure analysis and component replacement.

**TEST F**: The final functional test offers increasing test coverages: start with “system wake-up”, proceed with performance tests. Test equipment remains the same; the fault coverage rises with test time.

Yield improvement of the components is a second important strategy, although considered to be more complicated. This improvement requires interaction with the component supplier.

Yield LNA: A first improvement could be obtained when moving from multi-project processing to a single wafer, for even higher yield a custom CSP is required.

Yield GRF: For the GRF, higher yield can only be achieved when packaged in a CSP, with a subsequent burn in step.

Yield Antenna ANT: Higher yield is only possible with more accurate processing.

Yield Fingerprint FP: Either ESD screening or temporary packaging will improve the yield of the fingerprint.

Moreover, investment in the “weak” processes Mount Antenna ANT, Injection Moulding MLD, and Lamination LAM will raise the respective yields as well.

A summary of all possible improvements is listed in Table 5.2. Compared to Table 5.1, for all critical-yield components/processes we have set the values to the lower border, so that for every yield improvement a penalty has to be paid.

In order to determine the optimum combination of all possible strategies, the search algorithm from the previous chapter is applied to screen the search space of $3^{15} = 14,348,907$ possible solutions of now 15 parameters \{LNA, GRF, ANT, FP, MNT, MLD, LAM, feA, feB, feC, rewC, feD, feE, rewE, feF\} with three applicable values.

5.3. Results and Discussion

Fig. 5.6 presents the combined result of several EA optimization runs using the methodology described in chapter 4, showing

- the initial population (+),
- the final population for comparison purposes (×),
- the final Pareto front (+, connected with lines),
- the “min-medium-max” results (■),
- plus the risk region from Fig. 5.5 (dashed box).

The results form a “normal” Pareto front according to Fig. 4.14(a).

As already observed in previous examples, the Pareto front builds a connection between the “min”- and the “max”- combination result, outperforming the “min”- and the “med”- point by cost and defect level. The “max”- combination again gives the best (=lowest) defect level, but provides only minimal improvement compared to the best Pareto front solution.

Table 5.3 shows the decoded parameters of the Pareto points in Fig. 5.6. A preferable trade-off point would be at {468.67 a.u.; 0.1212} (marked) because at this point the slope of defect level decrease has leveled off. For all Pareto solutions, the optimization algorithm favors highest component yields instead of rigid testing. This can be explained by the good yield improvements for the components and processes at rather low cost, compared to investment into higher test effort. The three “steps” observed in Fig. 5.6 where the defect level reduces correspond very well with the progress of the final test coverage $f_{C\beta}$ towards 90%.

Table 5.3 shows the decoded parameters of the Pareto points in Fig. 5.6. A preferable trade-off point would be at {468.67 a.u.; 0.1212} (marked) because at this point the slope of defect level decrease has leveled off. For all Pareto solutions, the optimization algorithm favors highest component yields instead of rigid testing. This can be explained by the good yield improvements for the components and processes at rather low cost, compared to investment into higher test effort. The three “steps” observed in Fig. 5.6 where the defect level reduces correspond very well with the progress of the final test coverage $f_{C\beta}$ towards 90%.

In addition to that, the antenna pre-test is also favorable for improving the maximum achievable yield for this component, again fostered by the comparably low cost effort for high $f_{C\beta}$ (see Table 5.2).

Neither of the rework processes rewC and rewE has been selected. The reason can be found in the rather small number of systems to be produced (10,000), where only a small number of units would arrive
5.3. Results and Discussion

Table 5.2: Parameters for improvement concepts eCard, 10’000 units p.a.; processes/components from Table 5.1 not shown remain unchanged. Cost is denoted in anonymous units (a.u.) due to confidentiality reasons.

<table>
<thead>
<tr>
<th>Process</th>
<th>MOE</th>
<th>Yield [%]</th>
<th>FC [%]</th>
<th>Cost [a.u.]</th>
<th>NRE [a.u.]</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>LNA</td>
<td>ID4</td>
<td>85</td>
<td>10</td>
<td>20'000</td>
<td>20'000</td>
<td>MPW* single wafer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>95</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>99</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRF</td>
<td>ID5</td>
<td>80</td>
<td>12</td>
<td>0</td>
<td>100'000</td>
<td>Bare Die packaged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>95</td>
<td>14</td>
<td></td>
<td>100'000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>99</td>
<td>15</td>
<td></td>
<td>100'000</td>
<td></td>
</tr>
<tr>
<td>Antenna (ANT)</td>
<td>ID20</td>
<td>75</td>
<td>16</td>
<td>15'000</td>
<td>15'000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>80</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>85</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fingerprint (FP)</td>
<td>ID24</td>
<td>85</td>
<td>19</td>
<td>10'000</td>
<td>20'000</td>
<td>ESD screen temp packaged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>90</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>95</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mount Antenna (MNT)</td>
<td>ID23</td>
<td>80</td>
<td>5</td>
<td>7'000</td>
<td>7'000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>90</td>
<td>7</td>
<td></td>
<td>7'000</td>
<td></td>
</tr>
<tr>
<td>Injection Moulding (MLD)</td>
<td>ID28</td>
<td>80</td>
<td>6</td>
<td>60'000</td>
<td>60'000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>87.5</td>
<td>10</td>
<td></td>
<td>60'000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>95</td>
<td>14</td>
<td></td>
<td>60'000</td>
<td></td>
</tr>
<tr>
<td>Lamination T/B Foil (LAM)</td>
<td>ID30</td>
<td>60</td>
<td>2</td>
<td>5'000</td>
<td>5'000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>72.5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>85</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST A (fcA) (LNA)</td>
<td>ID7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15'000</td>
<td>no test DC temp carrier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>3</td>
<td></td>
<td>50'000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>90</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST B (fcB) (GRF)</td>
<td>ID8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15'000</td>
<td>no test DC temp carrier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>3</td>
<td></td>
<td>60'000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST C (fcC)</td>
<td>ID16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>40'000</td>
<td>no test simple GPS extended GPS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>3</td>
<td>40'000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>80</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REWORK C (rewC)</td>
<td>ID17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5'000</td>
<td>bond only comp replace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>3</td>
<td></td>
<td>25'000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>90</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST D (fcD) (Antenna)</td>
<td>ID21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5'000</td>
<td>measure return loss full</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>3</td>
<td>25'000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>90</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST E (fcE)</td>
<td>ID25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15'000</td>
<td>interconnect digital</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
<td>0</td>
<td></td>
<td>25'000</td>
<td>functional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>70</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REWORK E (rewE)</td>
<td>ID27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5'000</td>
<td>feed, FP comp replace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>15</td>
<td></td>
<td>25'000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>70</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST F (fcF)</td>
<td>ID30</td>
<td>70</td>
<td>15</td>
<td>50'000</td>
<td>50'000</td>
<td>functional simple</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80</td>
<td>20</td>
<td></td>
<td></td>
<td>functional medium</td>
</tr>
<tr>
<td></td>
<td></td>
<td>90</td>
<td>23</td>
<td></td>
<td>50'000</td>
<td>functional extended</td>
</tr>
</tbody>
</table>

* MPW: multi-project wafer
Figure 5.6: Results of the EA card production optimization: Key factor for the improvement was primarily investment into qualified, i.e. tested, components, significantly improving the overall defect level, comparable to the "best case" from the risk analysis. For further defect level improvements, a cost premium has to be paid: the Pareto front exhibits three "steps" indicating the increase in final fault coverage $f_{CA}$. The trade-off point is circled where defect level decrease levels off: This point has a cost similar to the center point of the risk analysis, but with only 25% the defect level. Close to the Pareto front, there also exist other individuals that are less optimal in terms of cost and defect level, but might be easier to implement in reality. The use of a Pareto tolerance band instead of a simple front could help keep track of those solutions, but the matter is subject to further research (EA parameters are population size $N_{Pop} = 200$, Pareto set size $N_{Pareto} = 20$, $n_{maxgen} = 100$ iterations, crossover probability $p_{cross} = 0.8$, and mutation probability $p_{mut} = 0.1$).
5.4. Conclusions

Table 5.3: Parameter structure for eCard Pareto front of Fig. 5.6; possible trade-off point is marked. Note how cost rises and defect level decreases with the step-like change of final fault coverage $f_{cA}$.

<table>
<thead>
<tr>
<th>Yield Test/Rework Results</th>
<th>Cost</th>
<th>Defect Lvl</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.99 0.99 0.85 0.95 0.95 0.85 0 0 0 0 0 0 0 0 0</td>
<td>371.31</td>
<td>0.3407</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>371.97</td>
<td>0.2821</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>375.63</td>
<td>0.2788</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>377.98</td>
<td>0.2703</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>381.6</td>
<td>0.2677</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>395.98</td>
<td>0.2667</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>397.3</td>
<td>0.2521</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>404.29</td>
<td>0.2547</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>409.23</td>
<td>0.2165</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>413.4</td>
<td>0.2128</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>426.3</td>
<td>0.2015</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>432.97</td>
<td>0.2013</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>437.37</td>
<td>0.1399</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>446.12</td>
<td>0.1393</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>450.26</td>
<td>0.1323</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>456.32</td>
<td>0.1296</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>468.67</td>
<td>0.1212</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>479.07</td>
<td>0.1203</td>
</tr>
<tr>
<td>0.99 0.99 0.85 0.95 0.9 0 0 0 0 0 0 0 0 0 0</td>
<td>491.88</td>
<td>0.1195</td>
</tr>
</tbody>
</table>

in the rework cycles, rendering the investment in a rework station expensive. For all other tests, if a test has been chosen, only the lower fault coverage has been selected. This can be explained with the continuously increasing NRE cost, next to the increasing direct cost, doubling the expenses for quality improvements.

Then, instead of switching to the higher $f_{cA}$-value for the same test to improve the defect level, another test gives a better cost/defect level, traveling down the Pareto front. This behavior is due to the mathematical optimization, selecting strictly the best value for the Pareto set. As one can see from the final population (∝ in Fig. 5.6), there do exist individuals (=solutions) close to the final Pareto front offering an inferior, though comparable, cost and quality, which might be easier to implement than the Pareto solutions. One answer to this notion could be the logging of all population and Pareto individuals within a certain distance of the Pareto front to keep track of such “sub-optimal”, yet better implementable solutions.

Comparing the trade-off setup with the initially targeted one in Table 5.1, we see that first efforts should concentrate on a full antenna test $f_{cD}$ and an (at least) moderate functional test before lamination $f_{cE}$. Further improvements can be achieved by pre-testing of components, and rework steps can only be seen as “ultima ratio”.

5.4. Conclusions

In this chapter we have shown the application of our trade-off methodology to an industrial early-design analysis for the production of a new type of Smart Card. The results confirm the rule-of-thumb to work as much as possible with KGD-like components instead of finding and fixing faults later in the process (cf. Fig. 1.5(a)). A reason for this fact is that, while improvement in fault coverage had to be paid with increases in both direct cost and NRE, a yield improvement required mostly only a (smaller) raise of the direct cost. Also, the maximum yield reachable was in most cases higher than the maximum fault coverage in range (see Table 5.2). Still, in a local environment striving for medium defect level improvements to meet specifications, it can make more sense to introduce additional, new test steps than to push the fault coverage of existing tests.

Summarizing the findings we can say that since this specific type of smart card production has not yet been fabricated, our analysis gave some insight into the fabrication cost structure, providing an initial cost/defect level risk range and an estimate for a trade-off point. We have to point out that even with the found Pareto-optimal parameters or under “max” parameter conditions the defect level is still unacceptably high even for prototyping production (every eighth prototype would fail although passing all tests). Currently, the best strategy seems to be to work with known-good components as much as possible. Even with the current implementation, the yield loss has a share of 25% to 30% of the total cost (NRE approx. 10%), leaving plenty of room for cost optimization. But before developing future production strategies,
the estimated yield and fault coverage figures have to be corroborated by measurements. The elimination or at least mitigation of damage caused by "weak" process steps would be a key point to achieve better defect level results, namely injection moulding (ID28) and foil lamination (ID30). SOC, i.e. integration of the multiple processing components into a single piece of silicon including memory, might be an option to reduce the number of subsequent packaging processes, thus increasing yield and therefore quality. If these actions still do not lead to an acceptable quality level, the general specifications driving the eCard thickness and therefore requesting the weak steps ID28 and ID30 for card production have to be re-examined. A possible solution could be moving towards more rigid housing alternatives, thus omitting ID28 and ID30, while changing e.g. to a wireless card interface instead of requiring compatibility with existing terminal devices.

In the subsequent chapter we show how the presented approach has to be extended to cope with such issues as fabrication and partitioning concepts.
Investigating Fabrication and Partitioning Issues

In this chapter we outline additional manufacturing choices to be addressed, such as external contract manufacturing (ECM) vs. in-house manufacturing (IHM) and system-on-a-package (SOP) vs. system-on-a-chip (SOC), and how they influence cost and quality issues. Based on these influences we detail how the presented modeling language has to be extended to cope with these more general problems.

Also, we show how the trade-off concept and the stability investigations from chapter 4 can be used for trend analyses when detailed data is not available.

6.1. Even more Manufacturing Choices

The number of manufacturing options is not only driven by the number of technologies but also by different production and design concepts. Thus, in addition to the purely technical options described in chapter 1, recently also more fabrication- and partitioning-related issues have surfaced:

- Should a company set up/use an own fabrication line in-house, or should it subcontract external manufacturing partners?
- System-on-chip (SOC) vs. System-on-a-package (SOP)? Should all functional blocks be mapped onto a single IC? Or should a set of sub-ICs form the desired functionality? What is a suitable implementation for an electronic system? When using SOC, the right IC technology has to be chosen; when opting for SOP, the question is the right packaging technology.

Similar to the cost/quality trade-off, no general wisdom exists to pick the best solution. Internal or external manufacturing, SOC or SOP have significant impacts on test, the SOC strategy will limit the availability of components, the cost structures and lead times are different, etc.

We will outline in the following subsections, what impact these fabrication/partitioning concepts have.

6.1.1. External Contract Manufacturing

Lately, the business model of outsourcing production facilities has gained significant interest. The idea behind the outsourcing concept is to minimize “dead”, i.e. bound capital investment and to be independent from manufacturing technologies while offering “customer solutions” instead.
6.1. Even more Manufacturing Choices

Front end...

In the area of semiconductor design and manufacturing, companies using this model are called “fabless companies”, and their counterparts doing production “foundries”\(^1\).

**Fabless (without fab)** refers to the business methodology of outsourcing the manufacturing of silicon wafers, which several semiconductor companies have adopted (as of Q4 2000, at least 48 companies constitute the FABLS stock index). Fabless companies focus on the design, development and commercialization of their products and outsource (at least a significant share) of their production to silicon wafer manufacturers, or foundries.

**Foundry** is a service organization that caters to the processing and manufacturing of silicon wafers. A pure-play foundry is a company that focuses 100 percent of its efforts on this service and offers no end products. These companies typically develop and own the process technology or partner with another company for it. Some companies offer 100 percent wafer manufacturing services and others offer foundry services to supplement their company’s own requirements.

This business model is especially attractive for start-up companies, but also well-known players such as Xilinx, Adaptec, etc. follow the fabless road. These companies can directly benefit from semiconductor technology advances without having to invest in expensive cleanroom or production technology.

... and Back end

For packaging/system design and production, the outsourcing concept has existed even longer, and the partners are called design houses/fabless original equipment manufacturers (OEMs) and external contract manufacturing (ECM) companies. For fabless OEM companies, such as u-blox, the benefits are the same as for semiconductor fabless companies: not to be determined to use a specific technology. See the two scenarios for the GPSMS1 from Example 3.1: a change from wire bonding to CSPs would not result in bound capital investment for wire bonders that are no longer required.

The disadvantage of external contract manufacturing is that not everything is well-controllable in the manufacturing process, including production timing, quality, and cost. Apart from that, of a product’s entire value chain only a part is created by the fabless company, and therefore that company has to share the profit with the contract manufacturers. This fact can drive the need to set up an own manufacturing line or to develop and to produce own components (u-blox: own GPS chip set [127]).

This question of “we don’t want to rely on others” vs. “let’s focus on our core competences” is a strategic decision, sometimes also a historic one (when the company started with its own production), and will not be changed lightheartedly, see the example of Flextronics taking over Ericsson’s mobile phone production [24]. But the question can rise again with every new product that does not exactly match existing business and manufacturing areas.

Comparing IHM and ECM concepts by parameters according to section 3.2, namely direct cost, NRE, and yield/test coverage, we can say that internal and external (fabless) manufacturing have different cost aspects: fabless companies have higher direct cost, whereas manufacturing companies have a much higher annual depreciation. Thus, own manufacturing sites pay off especially for higher production volumes. Regarding yield and test issues, in principle internal and external manufacturing can achieve the same results, perhaps with internal performing better in testing since there is a closer coupling to the product knowhow.

On the other hand, contract companies have more resources and a higher level of knowhow available to achieve better results or to produce in a shorter time. Thus, in general (except for in-house “hot runs”) external manufacturers have a time lead over internal forces, especially when a production line is to be established (u-blox: although moving towards own component development, a production cooperation is set up with chip manufacturer Atmel [127]). Regarding information accuracy, cost, quality, and production time efforts can be controlled more easily within the own company, whereas external improvements are more difficult to achieve. Thus, for external manufacturing the precision of all cost, timing, and quality data is lower.

\(^1\)see www.fsa.org for semiconductor fabless companies
6.1. Even more Manufacturing Choices

A summary of the considerations above is shown in Table 6.1. Accurate data has to be extracted from specific quotations. Comparing external and in-house manufacturing, a MOE production model, such as the one shown in Fig. 3.9, does not have to be changed since the process steps to be undertaken remain the same. Usually, an in-house production model will be more detailed since more process details are available or can be obtained by measurements.

**Table 6.1:** General comparison of external contract manufacturing (ECM) to in-house manufacturing (IHM).

<table>
<thead>
<tr>
<th></th>
<th>Direct cost</th>
<th>Yield/Testing</th>
<th>NRE</th>
<th>Design plus manufacturing time</th>
<th>Data accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECM (compared to IHM)</td>
<td>↑</td>
<td>→</td>
<td>↓</td>
<td>( \wedge )</td>
<td>( \wedge )</td>
</tr>
</tbody>
</table>

↑: increase (large), \( \wedge \): increase (small), ↓: decrease (large), \( \wedge \): decrease (small), →: as is

6.1.2. SOP vs. SOC

Similar to contract manufacturing, partitioning issues, i.e. the grouping of functional units to blocks to be implemented in the same technology or device as described in Fig. 1.1, are also more of strategic and/or historic nature, since a change has significant impact on lead times and cost structures. The most recent “buzz word” gaining public interest is “System-on-a-chip (SOC) vs. System-on/in-a-package (SOP)” [22].

In the long run, an SOC shall include all functional units of a system, so far manufactured in various semiconductor technologies, into a single piece of IC. SOP instead proposes the manufacturing of every functional block with the best-suited technology leaving system integration to the surrounding package level.

**Low-complexity SOC**

Partitioning trade-offs of functional blocks into one piece or into many pieces of silicon have already been the focus of many investigations, such as the “tiled silicon” approach [128]. Whereas this approach affected only a small number of companies with freely partitionable logic and/or memory as for massively parallel computing, today SOC is attractive for many companies with self-developed ASICs wanting to incorporate external DSP and µC cores. Thus, intellectual property issues also have to be considered. These ASIC/DSP devices are also the primary application fields identified by [22] which will act as SOC enabler to reduce risks as legal and testing problems. We will denote them as “low-complexity” SOC, since they do not require a significant technology leap, including only external cores of the same technology, embedded memory, RF, and/or FPGA technology. “Low-complexity” SOC is expected to be established in the next 5 years [129].

**High-complexity SOC**

“High-complexity” SOCs require mixing systems that are much less related than “low-complexity” SOCs, including logic and MEMS, sensors, or electro-optical technologies, where integration procedures still have to be developed. Challenges comprise

- special wiring and metalization for high-performance analog and RF circuitry;
- special upper-layer processes and passivation for integrated MEMS;
- special localized processing for integrated optical sensors;
- use of discretionary implants for critical low-power regions of the chip, e.g. for dual threshold.

Although the names are new, the concepts are not completely. The SOP is more or less the continuation of the multichip module (MCM) technology. Jack Balde (IPC) during the HD Int’l 2000 in Denver: “Whenever somebody says SOP, you should wash out his mouth because what he really means and should say instead is MCM!”.
6.1. Even more Manufacturing Choices

Table 6.2: Added process complexity (number of mask level) for SOC technologies, based on CMOS logic [129].

<table>
<thead>
<tr>
<th>Technology</th>
<th>Logic*</th>
<th>SRAM</th>
<th>Flash</th>
<th>DRAM</th>
<th>CMOS RF</th>
<th>FPGA</th>
<th>MEMS</th>
<th>FRAM</th>
<th>Chem Sensors</th>
<th>Electro Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electro Opt</td>
<td>5-8</td>
<td>6-9</td>
<td>9-12</td>
<td>9-13</td>
<td>8-12</td>
<td>7-10</td>
<td>7-18</td>
<td>9-13</td>
<td>7-14</td>
<td>0</td>
</tr>
<tr>
<td>Chem Sensors</td>
<td>2-6</td>
<td>3-7</td>
<td>6-10</td>
<td>6-11</td>
<td>5-11</td>
<td>4-8</td>
<td>4-16</td>
<td>6-11</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FRAM</td>
<td>4-5</td>
<td>3-4</td>
<td>7-9</td>
<td>2-3</td>
<td>9-10</td>
<td>6-7</td>
<td>9-15</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEMS</td>
<td>2-10</td>
<td>3-12</td>
<td>6-14</td>
<td>6-15</td>
<td>5-15</td>
<td>4-12</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPGA</td>
<td>2</td>
<td>2-4</td>
<td>4-6</td>
<td>3-7</td>
<td>5-7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMOS RF</td>
<td>3-5</td>
<td>5-9</td>
<td>6-9</td>
<td>6-10</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAM</td>
<td>4-5</td>
<td>3-4</td>
<td>5-9</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flash</td>
<td>4</td>
<td>3-4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRAM</td>
<td>1-2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*: Standard CMOS processes require from 10 (two metal layers) to 26 (seven metal layers) lithography steps [130].

Due to these challenges, the advent of “high-complexity” SOCs is not expected in the near future. Quality will be an issue with SOC especially when integrating highly different technology cores, since new fault models and new testability concepts have to be developed [129]. Also full test vector supply could reveal more of the intellectual property than the core designer is willing to give away.

Extrapolating SOC Data

Unlike for the previous ECM vs. IHH considerations, for SOCs no simple quotations can be used to obtain cost and yield data. Though massively higher integration and improved processes have mitigated the yield problems compared to the wafer scale integration (WSI) days, yield still matters when incorporating e.g. RF or MEMS functionality into CMOS. Currently, no realistic yield data is available. A hint for the complexity increase inherent to SOC is the addition of mask levels (see Table 6.2).

State-of-the-art cost interpretation of the SOC approach is to incorporate additional features into the standard CMOS process [129]. A basic model to generate numbers for cost, yield, and processing time is therefore tied to the number of mask levels and some scaling factors compared to this standard CMOS process.

The yield $y_n$ of a new product, either using a new technology mix or incorporating more functional blocks, can be derived from the yield $y_c$ of an existing product by the following extrapolation (Eq. 6.1, [131]).

$$ y_n = \frac{\gamma_{\text{yield}}}{y_c} = \frac{n_{\text{masks,n}}}{n_{\text{masks,c}}} \cdot \gamma_a \cdot \gamma_s \cdot \gamma_c, $$

where $n_{\text{masks,c}}$: number of masks existing product,
$n_{\text{masks,n}}$: number of masks new product,
$\gamma_a$: area scale factor,
$\gamma_s$: sensitivity factor, $f(\text{feature size})$,
$\gamma_c$: complexity factor, $f(\text{no. of gates})$.

Similar considerations lead to an estimate for the cost $c_n$ of a new SOC device, based on the cost of an existing CMOS device $c_c$ (Eq. 6.3):

$$ c_n = \frac{\gamma_{\text{cost}}}{c_c} = \frac{n_{\text{masks,n}}}{n_{\text{masks,c}}} \cdot \gamma_a \cdot \gamma_d, $$

where $\gamma_d$: cost scaling factor.
where $n_{\text{masks},e}$: number of masks existing product,

$n_{\text{masks},n}$: number of masks new product,

$\gamma_a, \text{cost}$: area scale factor,

$\gamma_d, \text{cost}$: design factor, $f$ (feature size, routing density),

Again using the direct cost/yield/NRE parameters for comparison, SOC has a significantly higher investment cost with possibly lower cost on the system level (reduction of packaging steps, reduction of substrate/carrier sizes, lower system power consumption), driving down the system’s direct cost. Instead, SOP will bear higher cost per unit with lower investment. Considering production time, SOC fabrication has higher lead times compared to SOP with off-the-shelf components, but in case an ASIC has to be fabricated anyway, the lead times are comparable. Again, the basic considerations are summarized in Table 6.3.

<table>
<thead>
<tr>
<th>Table 6.3: Influence of SOC vs. SOP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct cost</td>
</tr>
<tr>
<td>low-complex SOC</td>
</tr>
<tr>
<td>high-complex SOC</td>
</tr>
</tbody>
</table>

$\uparrow$: increase (large), $\nearrow$: increase (small), $\downarrow$: decrease (large), $\rightarrow$: decrease (small), $\rightarrow$: as is

When evaluating SOC vs. SOP, the manufacturing model has to be adapted to include the new SOC processing steps and other process steps influenced by the SOC approach (different design route, new packaging concept).

6.2. Extending the Model

In principle, all four different concepts (IHM, ECM, SOC, SOP) can be mapped onto a MOE production model for evaluation, since all processes could be broken down to their basic process steps. But from the considerations above, we see that two important aspects have to be included to make a fair comparison:

- **annual production quantity**: the volume or number of units to be produced;
- **production time**: how long it takes to design, qualify, and manufacture an electronic product, including all lead times.

Generally, the two scenarios IHM and SOC are more suitable for high-volume applications, and in general have a longer lead time. Therefore, for short product cycles/prototype volume products companies prefer to use external manufacturing facilities and off-the-shelf components in an SOP; for products with longer life cycles and/or driving mass volumes the setting up of an own fabrication line (IHM) and investment into own components (e.g. SOC) pay off. Similar to the cost/quality tradeoff, no information exists on when to switch between the alternatives.

In the following subsections we introduce ways to incorporate volume and production time aspects into our MOE model in order to determine switching conditions.

6.2.1. Economics of Scale

Since the production volume itself is not to be optimized, being instead a specification, it is more appropriate to include volume as a parameter, sampling rather discrete values than sweeping the entire range. When analyzing different production volumes, an existing process can be scaled to the volume range of interest and then be compared to another process.

We assume that changing the volume of a given process has only minimal influence on the objectives quality and production time, but its main impact on cost, namely

- **direct cost**, e.g. material cost, is driven down with higher order purchase quantities and therefore with the production volume,
6.2. Extending the Model

- NRE cost, i.e. high investment, design, and ramp up cost is spread among a higher number of units, again driving the total cost down.

The yield loss is not affected by the volume since after a setup period it is independent from the number of produced units, as we have seen in Fig. 3.12. The production volume in a MOE simulation is determined by the number of units in a collector item, which is the number of units to be shipped \( N(P_{ship}) \). This feature is already implemented in the existing version.

An example for the two-dimensional cost/quality objective space can be found in Fig. 6.1, where we have increased the annual production volume of the eCard study in chapter 5 from 10'000 units to 200'000 units p.a. Although the supporting points of the two Pareto fronts look slightly different (which can also be explained by the higher number of Pareto points conceded at the search), the general behavior and the conclusions drawn from the parameters in section 5.3, which test to focus on first, do not change. The left shift of the front results from the lower NRE-per-unit cost.

![Figure 6.1: When increasing the production volume in the eCard study by factor 20, the general behavior of the Pareto front does not change. The left shift of the front results from the lower NRE-per-unit cost.](image)

6.2.2. Including Product Development and Manufacturing Time

In order to include product development and manufacturing time into the set of optimization goals, similar to cost and quality, in the MOE model time information has to be collected throughout the entire production process.

**Definition 6.1**

**Product Development and Manufacturing Time PT.** The product development and manufacturing time \( PT \) incorporates all the process time spans required to develop and to manufacture a HDP electronic product. The process time spans can be divided into two categories:

- independent, having no other predecessor process than the beginning of production itself; examples for independent time spans are e.g. design times, component lead times, process setup times, etc.; independent times are called lead times.

- dependent, where another process has to be completed before this process can start. Dependent time spans comprise all manufacturing and testing times and are called manufacturing times.
6.2. Extending the Model

Every process step is characterized by both, lead time \(t_{\text{lead,proc}}\) and manufacturing time \(t_{\text{manu,proc}}\).

Definition 6.1 follows the standard project planning procedure adopted by commercial planning tools such as Microsoft® Project, defining an earliest starting time or predecessors for every task. A “task” can be translated into a “process” in our context. The simplest way to collect manufacturing time information is to add a time attribute to the colored tokens \(\omega\) (cf. Def. 2.11 and Eq. 2.2), leading to Eq. 6.5 for a time-enhanced token \(\omega^+\):

\[
\omega^+ = \{\text{cost}, \text{time}, \text{error}, \ldots\}; \text{cost, time} \in \mathbb{R}, \text{all others} \in \mathbb{N}.
\] (6.5)

This manufacturing time information will be processed following similar firing rules as described in section 2.5. Whereas for subsequent processes the manufacturing times will be added, for assembly steps only the longest path has to be considered (Eq. 6.6):

Assembly step: \(\omega^+_{\text{time}} = \max_{\text{inputs}} (\omega_{\text{time},i}) + t_{\text{manu,assembly}}\)

other processes: \(\omega^+_{\text{time}} = \omega_{\text{time}} + t_{\text{manu,other processes}}\). (6.6)

The lead times will be evaluated separately, as these activities can be undertaken in parallel, and therefore no synchronization is necessary (similar to the NRE cost considerations). Therefore, in analogy to Eqs. 2.35 and 2.37, we define the total product development and manufacturing time \(PT\) as in Eq. 6.8, leading to Eq. 6.9:

\[
PT = \text{Manufact. Time}_{\text{all Proc}} + \text{Lead Time}_{\text{all Proc}} = \frac{1}{N(P_{\text{ship}})} \sum_{N(F_{\text{ship}})} \omega_{\text{time}} + \max (t_{\text{lead,all Proc}}).
\] (6.8)

Once the total production time has been computed, it could be included into the existing cost objective. Fig. 6.2 gives an example for such a strategy where time issues can be translated to total unit cost using a cost correction factor. On the other hand, this “cost penalty” approach requires the construction of a market model including prediction of the market behavior over the entire product life, which we believe is difficult to achieve accurately. Moreover, the “penalty” approach belongs to the class of quasi-single optimization problems such as introduced in section 4.1 and suffers from the same shortcomings, such as unwilling compensation of under-performing objectives.

In order to avoid these shortcomings, we include the production time as a third optimization objective next to cost and quality.

6.2.3. Trend Analysis

Once the MOE calculus is amended with the production time calculation, a full comparison of manufacturing models can be conducted. When no accurate data is available, at least a trend analysis can be performed.

In a trend analysis, for the input parameters cost, yield/fault coverage, and time, only ranges are specified. Either the parameters are linked together like in Table 4.5, where cost parameters are dependent to the yield/fault coverage parameters, or all parameters can vary independently. For every scenario under investigation, two “optimization” runs have to be performed to obtain the wrapping Pareto front: one run minimizing cost, defect level, and time, and another run maximizing them. Then, similar to Fig. 4.15, an enveloping body, in the three-dimensional case a box, can be derived: Any solution based on the parameter ranges specified is located inside this enveloping box.

For this trend analysis, we refrained from performing a full numerical study as in the prior examples, since the method has been verified in principle already in the previous chapters. An example of how such a comparison result would look like is given in Fig. 6.3. The graph compares two different manufacturing scenarios, the first using SOP, the second using SOC, for two different production volumes (low and high). When sampling the volumes, a window of opportunity can be identified when both solutions start to overlap. For our chosen example, we see that under high-volume conditions the SOP and the SOC solution intersect. With a graph like Fig. 6.3, a comparison is still very tedious as it is hard to identify when the Pareto solution
6.3. Examples

This section provides short examples illustrating the SOC vs. SOP and ECM vs. IHM trade-off areas. Since no accurate data was available to perform elaborate case studies, the examples sketch only the various scenarios, comparing rather “working points” than doing a full trend analysis as in Fig. 6.3.

Example 6.1 (cf. Example 3.1)

For the GPSMSI® the assembly shall be done internally to reduce the cost, thus increasing the profit margin. The following implications have to be considered:

- For low annual quantities of about 10'000 units, the component cost will roughly increase by 5%, although the ECM's additional fee of 8% to the component cost for procurement, storage, and logistics can now be removed. The reason is the significantly lower purchase cost the ECM can achieve due to its overall purchase quantity and negotiation power. Moreover, for IHM now own logistics costs (=NRE) have to be taken into account.
- The direct cost for the assembly decreases by roughly 80%, but in turn equipment costs of 3Mio a.u. for a full SMD line plus test equipment have to be depreciated over 5 years, plus 100'000 a.u. annual running costs.
- While the selection of a qualified ECM and time-to-volume took 6 months with costs of about 100'000 a.u. (to be depreciated over 2 years), equipment installation, and yield ramp up demanded 1 year with additional expenses of 1Mio a.u. (to be depreciated over 5 years). Since only centered processes are compared, all the yield data is to be the same as in Example 3.1.

3The data is based on assumptions.
Figure 6.3: Example for comparing an SOP and SOC solution at different production volumes after the MOE/EA enhancement with the production time: a) for the lower volume, SOP offers shorter production time and lower cost; b) when the volume is increased, especially the cost improves for SOC, since the large up-front investment can be spread among a higher number of units produced. The boxes mark the max-min range enveloping all results of a specific model (cf. the rectangular envelopes from Fig. 4.15), the supporting points indicate the best solution achievable with this model.
In order to fully benefit from the setting up of an own in-house assembly facility, higher annual production quantities of 1'000'000 units also have to be studied.

- For both scenarios, IHM and ECM, the component cost will equally decrease by 5% from today’s cost.
- The ECM assembly cost reduces by 10%, and also the in-house cost goes down to lower material costs and machine utilization.

The cost/yield implications of all scenarios can be found in Table 6.4. Fig. 6.4 shows the MOE model used for both IHM and ECM manufacturing of the GSP module.

Example 6.1 (continued) Fig. 6.5 presents the cost result details for the four different scenarios from Table 6.4. Setting the (existing) low-volume scenario “ECM,10’000 units” to 100%, the respective low-volume IHM setup results in a 60% cost overrun, which is driven by the high NRE part, while having lower direct cost and yield loss than ECM. Switching to high volumes of 1Mio units a year, the IHM scenario now winds up first due to its lower direct cost. For high volumes the NRE cost is almost negligible for both options.

Combining all three objectives, cost (“total cost”), quality (“DL”), and time (“PT”), in a radar plot as shown in Fig. 6.6, we find that for low volumes the ECM approach equals or supersedes IHM for all objectives. The assumption of similar yield and fault coverage data for centered processes in-house and externally gives the same defect level; the lead time to achieve such a centered process favors the ECM solution. With a production line already available in-house, both options would offer the same lead times, without loss of generality. Moving to higher volumes, the situation does not change for DL and PT, while IHM is now the most cost-effective choice.

Summarizing the findings, we can say that under the condition of a market volume of 5Mio units over 5 years, which allow to fully depreciate the equipment investment, the move towards IHM would be justified.
### Table 6.4: Partial cost for the ECM vs. IHM scenarios; data is confidential and therefore given only in arbitrary cost units a.u.; all data imply a wire bond MCM-L build-up with 80/80/250 design rules.

<table>
<thead>
<tr>
<th>(all cost a.u.)</th>
<th>ECM, 10k (existing solution)</th>
<th>IHM, 10k</th>
<th>ECM, 1Mio</th>
<th>IHM, 1Mio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DC</td>
<td>NRE</td>
<td>DC</td>
<td>NRE</td>
</tr>
<tr>
<td>DSP</td>
<td>8.50</td>
<td>8.92</td>
<td>all</td>
<td>8.07</td>
</tr>
<tr>
<td>μC</td>
<td>5.75</td>
<td>6.03</td>
<td>comps;</td>
<td>5.46</td>
</tr>
<tr>
<td>SRAM/Flash</td>
<td>6.50</td>
<td>6.82</td>
<td>10k</td>
<td>6.14</td>
</tr>
<tr>
<td>Substrate</td>
<td>4.25</td>
<td>4.46</td>
<td>logistics</td>
<td>4.03</td>
</tr>
<tr>
<td>Assembly back side</td>
<td>18.20</td>
<td>3.20</td>
<td>3Mio</td>
<td>16.34</td>
</tr>
<tr>
<td>Solder front</td>
<td>11.38</td>
<td>2.30</td>
<td>100k (5yrs)</td>
<td>10.24</td>
</tr>
<tr>
<td>Setup efforts</td>
<td>(2yrs)</td>
<td>1Mio</td>
<td>(5yrs)</td>
<td>100k (2yrs)</td>
</tr>
</tbody>
</table>

**Figure 6.5:** Cost results for the four scenarios: with the initial low production volume, the ECM solution is more cost effective, while IHM pays off when moving to higher quantities. The lower yield loss values can be explained with lower direct cost per module sorted out.

**Figure 6.6:** Comparison of all three objectives cost, quality, and product development time for the ECM vs. IHM (the smaller the value the better): while for the low volume ECM is equal/superior in all objectives (left), the IHM scenario surpasses the ECM cost-wise when the production quantity increases.
The second example discusses the move from a system-on-a-package to a system-on-a-chip for the GPS module from Example 4.3.

**Example 6.2 (cf. Example 4.3)**

In this example we will investigate the benefits for the two-chip GPS SOP of Example 4.3 when moving towards a single-chip solution.

In order to integrate the RF front end chip (BiCMOS) and the correlator (CMOS), the two technologies have to be merged. According to Table 6.2, this merger results in an RF CMOS technology, requiring up to 50% more masks compared to the correlator. In the next step, we have to identify the scaling factors $\gamma_{\text{cost}}$ and $\gamma_{\text{yield}}$.

$\gamma_{\text{yield}}$: Since the area of the RF chip is roughly 25% of the correlator one’s, and hardly any connections, area, or I/O pads can be spared, the area scaling factor $\gamma_{a,\text{yield}}$ is assumed to be 1.2. The risk of integrating analog noise-sensitive structures into a CMOS device will be paid with a complexity scaling factor $\gamma_{c,\text{yield}}$ of 1.2 as well. The sensitivity factor $\gamma_{s,\text{yield}}$ remains at 1.

$\gamma_{\text{cost}}$: The area scaling factor $\gamma_{a,\text{cost}}$ is also set to 1.2, the design scaling factor is not required and set to 1.

Using Eqs. 6.1 to 6.4, the yield and cost for the RF-correlator SOC device can be extrapolated. Further implications for the SOC scenario are:

- Although external passive components are still required for the GPS single-chip package (SCP), the substrate type changes to a simple interposer, therefore reducing its cost and increasing the yield.
- Moreover, the assembly cost for this SCP decreases, the preliminary tests $f_{\text{c1}}$ and $f_{\text{c2}}$ have to be merged as well.
- While the design lead time for an SOC integration is about 4 months longer (compared to one year), bearing approx. 900'000 a.u. additional cost, the SCP manufacturing time is two instead of four weeks for every 100'000 units.

The cost/yield implications of the SOP/SOC scenarios can be found in Table 6.5. Fig. 6.7 shows the simple MOE model used for the SOP implementation, Fig. 6.8 the one for the SOC case. For this example, the MOE model had to be adjusted since the manufacturing flow is different.

**Example 6.2 (continued)**

Fig. 6.9 presents the cost result details for the three different implementations from Table 6.5. While for low volumes SOC is no match for SOP due to its severe NRE penalty, for higher volumes the cost comes in a comparable range, profiting from lower direct cost and thus lower yield loss.

![Figure 6.7: MOE SOP manufacturing model for the GPS RF front end (cf. Fig. 4.11).](image)
Table 6.5: Partial data for the SOP vs. SOC scenarios; cost data is confidential and therefore given only in arbitrary cost units a.u.

<table>
<thead>
<tr>
<th>(all cost in a.u., yield in %)</th>
<th>SOP, 10k (existing solution)</th>
<th>SOC, 10k</th>
<th>SOC, 1Mio</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="image" /></td>
<td><img src="image.png" alt="image" /></td>
<td><img src="image.png" alt="image" /></td>
<td><img src="image.png" alt="image" /></td>
</tr>
</tbody>
</table>

Moving to the radar plot combining the three objectives (Fig. 6.10), we see that for low volumes the SOC approach is only favorable in terms of defect level. Moving to higher volumes and extrapolating the total product design and manufacturing time for SOP, we find that with SOC 1Mio units can be produced in the same time range as with SOP, even minimally shorter (1 month). Also, in terms of cost now minimally lower amounts are reached.

Drawing an overall conclusion on the SOP vs. SOC findings, we see that in this case the SOC margin is very close. Taking into account

- that the $\gamma$-scaling factors only have been estimated
- and that RF CMOS integration has a known risk of excess noise endangering performance,

the use of SOC is at least questionable unless the performance concerns are overcome. Again, as in the last example, a market volume of 1Mio units over a year is mandatory to ensure a low NRE share.

Figure 6.8: MOE SOC manufacturing model for the GPS RF front end.
6.3. Examples

Figure 6.9: Cost results for the three scenarios: again setting the existing SOP low-volume solution to 100%, the low-volume SOC approach fails to show any cost benefits. Even when moving to higher quantities, the profit is below 10% (compared to the low-volume SOP cost, which is also certain to decrease with the quantity).

Figure 6.10: Comparison of all three objectives cost, quality, and product development time for SOP vs. SOC (the smaller the value the better): Offering only superior defect level for the low volume, for high volumes SOC is at least capable of closing the gap cost- and time-wise. Still, the SOC advantage is not encouraging enough to clearly favor this solution.
6.4. Summary

In this chapter we showed that the “sea of production choices” consists not only of the number of various component and manufacturing technologies, but also, on a higher level, of different fabrication business models and alternative partitioning concepts. These two aspects can also be investigated using an amended MOE model incorporating production time in a similar manner as the cost issue.

Since for fabrication and partitioning issues even less precise data information is available, a trend analysis is more appropriate. The representation of the SOC processing on the basis of the number of mask levels (see [129]) may be seen as a hint for this notion. We have presented some models to extrapolate SOC cost and yield, based on scaling factors and the number of mask levels, and have illustrated how a comparison for ECM vs. IHM and SOP vs. SOC could look. Given value ranges for cost, yield/fault coverage, and time parameters, a box enveloping all possible solutions can be derived to compare different scenarios.

The following chapter summarizes the entire work and sheds some light on possible future work.
Summary and Outlook

In Section 7.1 we review the research objective claimed in chapter 1, summarize our achievements, and tackle limitations envisaged throughout the work. Section 7.2 outlines some pieces of future research work to be undertaken.

7.1. Achievements and Fundamental Novel Results

The two primary research objectives were:

**Objective I**: to develop a model that incorporates all relevant aspects of manufacturing an existing HDP product in order to produce the metrics cost and quality.

**Objective II**: to make this model scalable to all types of HDP technologies.

In order to accomplish these two objectives, we analyzed different types of cost modeling approaches (section 2.2). The most promising approach found was process-oriented cost modeling, due to its applicability to all existing and future HDP technologies. We further examined different HDP processes in order to identify the main “building blocks” for such processes (section 2.3). Since in most cases HDP process interactions are described as flow graphs, we used a Petri net approach, being a specific type of graphs especially suited for production modeling (section 2.5). By employing colored tokens for this Petri net, we were able to map the calculation of cost and quality equations directly to the evaluation of the Petri net (section 2.6).

We implemented the five basic transition types for this Petri net, representing the “building blocks” for HDP processes. Because all other transitions can be constructed using these basic transitions, the scalability to all types of HDP technologies is ensured, overcoming the main drawback of other modeling approaches. For the colored HDP Petri net model, a graphical description language MOE together with a user interface has been set up (section 3.1).

Examples have shown how to analyze an existing process, set up a model, and obtain input data (sections 3.2 and 3.3).

**Objective III**: to automate the cost/quality trade-off and to develop a method that gives decision support.

To compare different cost/quality results, a ranking scheme had to be found. Selecting the Pareto approach, we pleaded for “true” multi-objective optimization to avoid possible compensation of underperforming objectives and to enable search space exploration (section 4.1). Because the search space could embrace too many solutions to be exploited manually, an automated search was required. We adopted an evolutionary algorithm by creating own encoding and recombination schemes and by coupling the EA to MOE for fitness extraction (section 4.2). Using an example case study for a GPS production, this approach
7.2. Future Work

has been verified (section 4.3). Combining for the first time the Pareto scheme with a cost/quality trade-off, a method has been created to identify turning points, where further quality improvements do not make sense anymore from an economical point of view.

The post-optimization procedure comprised the analysis of the parameters constituting the Pareto front solution and a stability analysis of a possible trade-off point. The reason for this stability investigation is to ensure that a drift of these “working point” parameters after adoption in the real process results only in a tolerable shift of the metrics cost and quality. Moreover, general considerations on Pareto front types have been discussed (section 4.4).

The application of the mechanism to an early-design trade-off for a smart card production pointed out the difficulty of extracting reliable data.

Objective IV: to show how the trade-off concept can be used to cope with fabrication (ECM) and partitioning issues (e.g. SOC/SOP).

The stability investigation also laid the fundamentals for considerations on more general fabrication and partitioning concepts. An analysis of the concepts showed that for proper evaluation production time and production volume also have to be included as metrics (section 6.1). Whereas volume investigations can be implemented as a parametric set of solutions, the production time should be included as a third objective. A time calculation model similar to the cost has been presented. Using the stability concept, trend analyses, where no definite data is available, can also be conducted (section 6.2). After implementation of this concept, it will be possible to determine when to change partitioning and fabrication concepts.

Limitations

The quality of the results relies heavily on the quality of the input data, and therefore the majority of the modeling and analysis work should be dedicated to data mining activities described in section 3.2. As the author had to experience, this is not always a simple job due to the fact that yield, test, and cost data is highly confidential intellectual property. Most of the data for the case studies has been obtained under a non-disclosure agreement or a research contract in the product prototyping phase. No further data has been shared during fabrication, and thus it was not possible to compare the findings with statistical production data.

7.2. Future Work

Future activities should head in two different directions: enhancement of the model itself and enhancement of the design space exploration.

Enhancement of the Model: Switching between Submodels

After implementation of the third optimization objective, as outlined in section 6.2, the encoding scheme could be extended to incorporate more than a single manufacturing model. So far, the search algorithm supports only parameter variation within a single model, but using a gene switching between several hidden alternative manufacturing options (see Fig. 7.1) would open the route towards a simpler comparison of different scenarios.

Unlike in the trend analysis from Fig. 6.3, where intersecting envelopes for the two distinct SOP and SOC models had to be analyzed to decide when to change from SOP to SOP, a single MOE model incorporating model switching would result in a single Pareto front.
Figure 7.1: Part of a one-chromosome individual similar to Fig. 4.7: depending on the actual switch value, a specific submodel would be selected using either an SOP or an SOC setup (selected branch marked gray).

Figure 7.2: Design space exploration model: parallel exploitation to search the design space faster, branching when appropriate (First-level interconnect flip chip can use MCM-D or MCM-L substrates, ... ) and abandon solutions whenever specifications are not met at performance barriers (..., but with MCM-L the number of interconnect layers required violate the cost barriers for the substrate.).
7.2. Future Work

Enhancement of the Design Space Exploration: Including more Issues

A next step would be to combine the cost/quality optimization with other performance assessments, such as routing density, signal integrity, etc. Part of this work has been accomplished by linking MOE to JavaCAD, a tool for placement/routing analysis [133] and to the commercial software EDANavigator from Xynetix, now Avant! [108, 134].

For a fully automated design space exploration there is still a long way to go. Although a tool chain for analyzing the different implementations seems to be available, the pure formal number of possibilities exceeds the analysis capacity. For an experienced design engineer this number reduces by orders of magnitude within the first few moments working on the design, intuitively ruling out infeasible solutions. A promising procedure recently proposed in [133] exploits several possible first and second interconnect implementations in parallel and only sorts out solutions violating performance boundaries (see Fig. 7.2). This procedure is an adoption of the graph theory’s “branch-and-bound” [66, 69], and could also be automated using the fitness concept of evolutionary algorithms.
Glossary

Scalars

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C(P_x)$</td>
<td>Capacity or maximum number of tokens in place $P_x$</td>
</tr>
<tr>
<td>$DC$</td>
<td>Direct cost</td>
</tr>
<tr>
<td>$DL$</td>
<td>Defect level</td>
</tr>
<tr>
<td>$E$</td>
<td>Edge of a graph</td>
</tr>
<tr>
<td>$F$</td>
<td>Fitness of an individual</td>
</tr>
<tr>
<td>$G(V, E)$</td>
<td>Graph</td>
</tr>
<tr>
<td>$G_d(V, E)$</td>
<td>Directed graph</td>
</tr>
<tr>
<td>$I(T_x)$</td>
<td>Input function of $T_x$ in a Petri net</td>
</tr>
<tr>
<td>$N(P_x)$</td>
<td>Number of tokens in place $P_x$</td>
</tr>
<tr>
<td>$NRE$</td>
<td>Non-recurring expenditure</td>
</tr>
<tr>
<td>$N_{\text{Pareto}}$</td>
<td>Size of the Pareto set</td>
</tr>
<tr>
<td>$N_{\text{pop}}$</td>
<td>Size of population</td>
</tr>
<tr>
<td>$O(T_x)$</td>
<td>Output function of $T_x$ in a Petri net</td>
</tr>
<tr>
<td>$P$</td>
<td>Place in a Petri net</td>
</tr>
<tr>
<td>$PT$</td>
<td>Product development and manufacturing time</td>
</tr>
<tr>
<td>$QL$</td>
<td>Quality level</td>
</tr>
<tr>
<td>$S_{\text{Pareto}}$</td>
<td>Pareto set; external elitist population</td>
</tr>
<tr>
<td>$S_{\text{pop}}$</td>
<td>Population</td>
</tr>
<tr>
<td>$T$</td>
<td>Transition in a Petri net</td>
</tr>
<tr>
<td>$TT$</td>
<td>Test transparency</td>
</tr>
<tr>
<td>$V$</td>
<td>Node of a graph</td>
</tr>
<tr>
<td>$YL$</td>
<td>Yield loss</td>
</tr>
<tr>
<td>$\alpha, \beta$</td>
<td>Inclination of the population shape</td>
</tr>
<tr>
<td>$\epsilon_r$</td>
<td>Dielectric permittivity</td>
</tr>
<tr>
<td>$\gamma_{\text{cost}}, \gamma_{\text{yield}}$</td>
<td>Technology scaling factor for cost and yield</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Colored token in a Petri net</td>
</tr>
<tr>
<td>$c$</td>
<td>Cost</td>
</tr>
<tr>
<td>$d^-(i), d^+(i)$</td>
<td>Incoming and outcoming degree of node $i$</td>
</tr>
<tr>
<td>$f_c$</td>
<td>Fault coverage</td>
</tr>
<tr>
<td>$n_0$</td>
<td>Fault clustering factor</td>
</tr>
<tr>
<td>$n_{\text{max rework}}$</td>
<td>Maximum number of rework repetitions for a repair step</td>
</tr>
<tr>
<td>$n_{\text{max generation}}$</td>
<td>Maximum number of generations</td>
</tr>
<tr>
<td>$n_{\text{param}}$</td>
<td>Number of parameters to be coded in a chromosome</td>
</tr>
<tr>
<td>$n_{\text{rep}}$</td>
<td>Number of repetitions in a process step</td>
</tr>
<tr>
<td>$n_{\text{res}}$</td>
<td>Bit resolution of a gene</td>
</tr>
<tr>
<td>$p$</td>
<td>Probability</td>
</tr>
<tr>
<td>$p_{\text{cross}}$</td>
<td>Crossover probability</td>
</tr>
<tr>
<td>$p_{\text{mut}}$</td>
<td>Mutation probability</td>
</tr>
<tr>
<td>$r(t_1, t_2)$</td>
<td>Route through a graph</td>
</tr>
<tr>
<td>$t_{\text{lead, proc}}$</td>
<td>Lead time of a process step</td>
</tr>
<tr>
<td>$t_{\text{manufacturing}}$</td>
<td>Manufacturing time of a process step</td>
</tr>
<tr>
<td>$y$</td>
<td>Yield of a component or process</td>
</tr>
</tbody>
</table>
Glossary

Vectors

X  Decision space
Y  Objective space
f  Target function vector
i  Individual
x  Decision vector
y  Objective vector

Abbreviations

µC  Microcontroller
A/D  Analog/Digital
AFIS  Automatic Fingerprint Identification System
AOI  Automated Optical Inspection
ASCII  American Standard Code for Information Interchange
ASIC  Application Specific IC
ATPG  Automated Test Pattern Generator
BCB  Benzo Cyclo Butene
BIST  Built-In Self Test
BP  Band Pass
CA  Conductive Adhesives
COO  Cost of Ownership
CSP  Chip Size or Scale Package
CostAS  Cost Assessment System
DC  Direct Current
DRAM  Dynamic RAM
DSP  Digital Signal Processor
EA  Evolutionary Algorithm
ECM  External Contract Manufacturing
EDA  Early Design Analysis
EEPROM  Electrically Erasable Programmable Read-Only Memory
EP  Evolution Programming
ES  Evolution Strategies
ESD  Electro-static Discharge
FC  Flip Chip
FCT  Functional Test
FEM  Finite Element Modeling
FPGA  Field Programmable Gate Array
FR4  Fire Retardant, Class 4
FRAM  Ferroelectric RAM
GA  Genetic Algorithm
GP  Genetic Programming
GPS  Global Positioning System
GSPN  Generalized Stochastic Petri Net
HC  Hill Climbing
HDP  High Density Packaging
HTCC  High Temperature Co-fired Ceramics
Bibliography


Acknowledgments

Thanks to

Gerhard Tröster for supervising this thesis, providing a generous research atmosphere, and asking the right questions,
Paul Franzon for co-supervision,
Etienne Hirt for many fruitful discussions, cross-checking the chapters, and pushing me forward,
Geert Bernaerts for the music and the late-evening dinners,
Rolf Schmid for sharing way too many coffee shots,
Didier Cottet for offering always a calmer view to things,
Rolf Enzler for uncountable proofreadings and his “hawk eye”,
Alex Rhomberg for programming support and joining the spinning, rowing, and muscle pump lessons,
the Wearables for their fresh and motivating spirit,
the Berliners for their cheering comments and visits,

my parents and also to my grand-parents who would have loved to join the party,
all the other people from Hanover, Berlin, and Zurich who supported me throughout my work,

and Claudi for just being what she is!
Curriculum Vitae

Personal Information

Michael Scheffler
Born June 14, 1969
Hanover, Germany

Education

1982–1988: Ratsgymnasium Hanover
1990–1996: M. Sc. (Dipl. Ing.) in Electrical Engineering, Technical University Berlin, with honors
1996–2001: Ph. D. student, Swiss Federal Institute of Technology (ETH) Zurich

Work

1996–2001: Teaching and research assistant, Electronics Laboratory, ETH Zurich

Additional Activities

1988–1990: Military service (Lt. dR)
1995–1996: ERASMUS exchange, Università degli Studi di Bologna, Italy for Master thesis