The performance prediction model

a methodology for estimating the performance of an FPGA implementation of an algorithm

The Performance Prediction Model – A Methodology for Estimating the Performance of an FPGA Implementation of an Algorithm

Tobias Jeger, Rolf Enzler, Didier Cottet, Gerhard Tröster
Swiss Federal Institute of Technology
Electronics Laboratory
CH-8092 Zürich
E-mail: tjeger@ee.ethz.ch or enzler@ife.ee.ethz.ch

Technical Report, January 2000

Abstract

Employing FPGAs for computing purposes requires a mapping process. Although this procedure saves substantial development time compared to the physical design of an ASIC, several mappings are almost always needed until the constraints can be met. The Performance Prediction Model (PPM) is a means to help reducing the development time while providing an estimation of the performance to be achieved. The estimation is based on a preliminary implementation[1] only and separates different influences, which improves its portability. The PPM yields an accuracy of about 10% in average and is thus suitable for finding a good implementation for starting the actual mapping process.

1 Introduction

Many a designer of electronic systems nowadays faces the question how the necessary performance for a given computational task can be obtained. Beyond the two common solutions, the design of an ASIC and the programming of a processor, there exists a third approach: employing reconfigurable hardware. With field-programmable gate arrays (FPGAs) time to market can be shortened while preserving the high computational abilities of application specific circuits.

FPGAs consist of a regular array of logic blocks which can be configured individually to provide the desired functionality. They provide relatively many storage elements compared to their logic abilities, and a mesh of routing channels allows generous inter-block routing.

Employing an FPGA for computational purposes requires some HDL code (similar to the code written for hardware synthesis) to be mapped to the FPGA's resources. Since there are many possible solutions for mapping an algorithm to reconfigurable hardware, the resulting performance figures may be spread over a wide range.

While designers are forced to step through the entire design flow to obtain performance figures and to compare them with the constraints, the Performance Prediction Model (PPM) [3] allows an estimation of these numbers while skipping most of the designing steps. It provides an estimation of the performance to be obtained with a particular implementation, and hints on how to improve the circuit.

This paper gives an overview over the PPM first and explains the particular steps then. After demonstrating its usage with two examples, the PPM is evaluated for the XC4000E FPGA family of Xilinx [5]. Finally the results are discussed, and the advantages of PPM are outlined.

2 The Performance Prediction Model

The major difficulty when mapping an algorithm to FPGA is finding the optimal implementation. The difficulty arises from the fact that it depends on the algorithm, the constraints and the chosen architecture simultaneously. What is more, if performance con-

[1] With *implementation* we mean a way to map a given functionality, the *algorithm*, to the *architecture* of the FPGA.

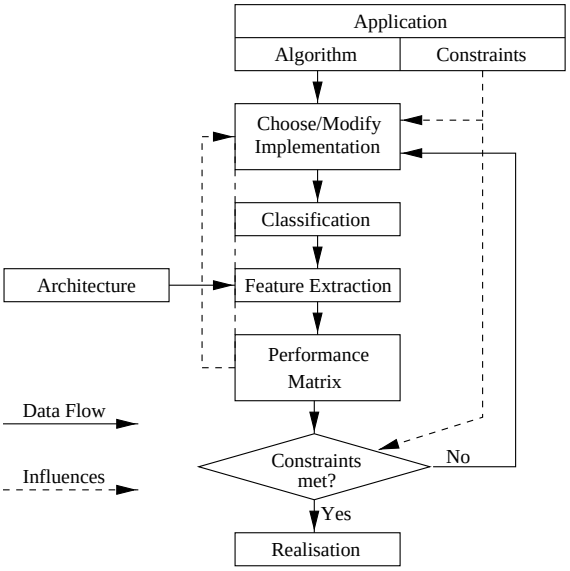

Figure 1. The Performance Prediction Model

straints are not met, the implementation must be modified, and thus there is a feedback loop between implementation and performance figures.

The PPM consists of the four steps (figure 1): choice and modification of the implementation, classification, feature extraction and performance matrix. The result is a set of performance figures for the chosen implementation. The results are compared with the constraints to see if the chosen implementation is suitable for entering the design process. If not, the implementation must be modified.

In the next sections, the PPM is deduced step by step.

2.1 Algorithm Classification

The first step of the PPM is to choose a preliminary implementation of the algorithm in the form of a schematic drawing. This sketch is then simplified by substituting subtractors and comparators with adders. Shifters are replaced by multiplexors (except barrel-shifters which are hardwired and need not show up in the implementation), and multipliers are replaced by one of the three following circuits:

- 1 array multiplier
- 1 adder, 1 multiplier (fan-in 4), 1 bit-operation (fan-in 2)
- 1 LUT[2] (fan-in ρ), ($2^{log_2(\rho)-2}-1$) adders (precision 2ρ), where ρ is the precision of the multiplier

[2] A LUT is a look-up table.

element	meaning
C_0	number of inputs
C_1	average input precision
C_2	number of outputs
C_3	average output precision
C_4	number of adders
C_5	average adder precision
C_6	number of multipliers
C_7	average multiplier input precision
C_8	number of multiplexers
C_9	average multiplexer fan-in
C_{10}	average multiplexer width
C_{11}	number of bit-operations
C_{12}	average bit-operation fan-in
C_{13}	average bit-operation width
C_{14}	number of LUTs
C_{15}	average LUT fan-in
C_{16}	average LUT width
C_{17}	number of iterations
C_{18}	internal parallelism

Table 1. Classification Vector

The obtained implementation can then be classified in a *Classification-Vector* of length 19 as listed in table 1.

A bit-operation is a boolean function which cannot be expressed as a mathematical formula. The iterations of C_{17} refer to the number of times a data set has to run through the *entire* circuit before appearing at the outputs. The internal parallelism reflects the number of non trivial data-paths through the circuit.

2.2 Feature Extraction

The next step of the PPM extracts *features* from the classification vector depending on the architecture of the FPGA device and its characteristics only. The eight features model the circuit as a *data river*. Figure 2 tries to visualise this idea. W is the **w**idth or the number of arms of the river, L is the number of **l**oops. The number of **In**puts In and **IO** pins Io belong to these features as well as the estimation for the number of pipeline **s**tages S, when pipelining after one or two logic stages. P denotes the average internal **p**recision, T the propagation **t**ime through the circuit and A the **a**rea requirements. The formulas for the particular features are listed below.

$$W = C_{18} \quad (1)$$
$$L = C_{17} \quad (2)$$
$$In = C_0 C_1 \quad (3)$$
$$Io = C_0 C_1 + C_2 C_3 \quad (4)$$

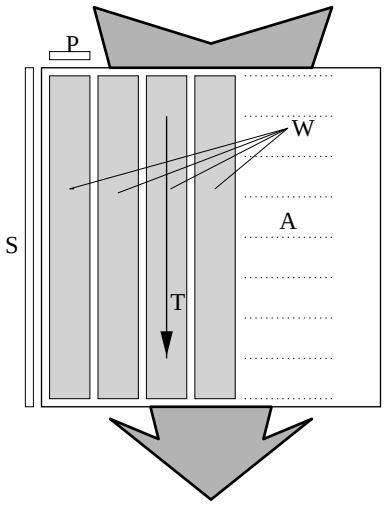

Figure 2. Principle of the Data River

$$S = \frac{1}{W}\left[C_4 + 8 \cdot C_6 + \frac{C_8 C_9}{4} + \frac{C_{11} C_{12}}{8} + C_{14}\right] \quad (5)$$

$$P = \frac{C_4 C_5 + C_6 C_7 + C_8 C_{10} + C_{11} C_{13} + C_{14} C_{16}}{C_4 + C_6 + C_8 + C_{11} + C_{14}} \quad (6)$$

$$T = \frac{1.4}{1000 \cdot W}\left[(rd \cdot C_5 + cd) C_4 + 2cd \cdot C_6 C_7 + cd\frac{C_8 C_9}{2} + cd\frac{C_{11} C_{12}}{4} + cd \cdot C_{15}\right] \quad (7)$$

$$A = \frac{C_4 C_5}{2} + 2 \cdot C_6 C_7^2 + \frac{C_8 C_9 C_{10}}{4} + \frac{C_{11} C_{12} C_{13}}{8} + \frac{C_{14} C_{15} C_{16}}{8} \quad (8)$$

The constants cd and rd in equation 7 refer to the combinational and the ripple line delay respectively. A speed grade of 3 for the XC4000E FPGA family implies 3 ns and 0.4 ns[3].

2.3 Performance Matrix

As the final step of the PPM, the *performance matrix* is computed. It provides performance estimations for frequency, latency, throughput, area requirements and IO pin requirements. The matrix consists of five rows and four columns, each of which representing a characteristic implementation. As shown in figure 3, these implementations differ in the degree of pipelining and parallelism. The rules how to calculate the estimations based on the eight extracted features are listed in table 2.

The units of frequency, latency and throughput are MHz, μs and Mbit/s respectively. The area is expressed in terms of configurable logic blocks (CLBs). The factors introduced in the first three rows originate in the additional delay caused by pipelining and control circuitry for serial implementations, the area overhead of which is estimated to be 30%. The additional two IO pins are the clock and reset pin of the circuit.

[3]These values are derived from the data sheet.

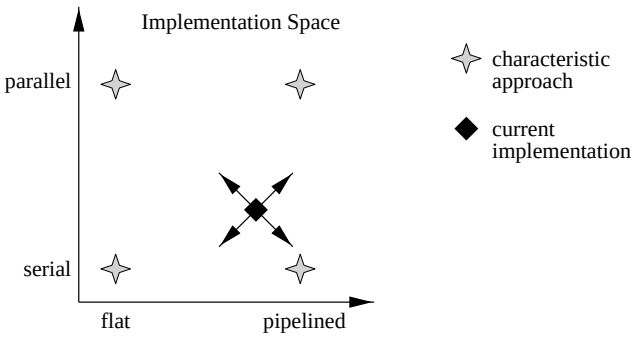

Figure 3. The Implementation Space

parallelism	parallel		serial	
frequency	$\frac{1}{T}$	$\frac{S}{1.3 \cdot T}$	$\frac{S}{2 \cdot T}$	$\frac{1}{1.5 \cdot T}$
latency	T	$1.3 \cdot TS$	$2 \cdot TLS$	$1.5 \cdot TL$
throughput	$\frac{In}{T}$	$\frac{InS}{1.3 \cdot T}$	$\frac{InS}{2 \cdot TL}$	$\frac{In}{1.5 \cdot TL}$
area [CLBs]	AL	AL	$1.3 \cdot A$	$1.3 \cdot A$
IO [pins]	$Io + 2$	$Io + 2$	$Io + 2$	$Io + 2$
pipelining	flat	pipelined		flat

Table 2. The Performance Matrix

The column which matches the architecture of the current implementation (parallel-flat, serial-pipelined, ...) best is taken as the desired performance prediction. Values from the other three columns can yield hints on how to modify the implementation to meet the constraints.

3 Usage of PPM

The usage of the PPM is demonstrated with two example algorithms below, the finite impulse response filter (FIR) and the motion estimation algorithm from MPEG2 encoding [4].

3.1 FIR

The implementation of the FIR algorithm is a multiple instantiation of the basic *tap* block shown in figure 4. It consists of a constant multiplier (vertical

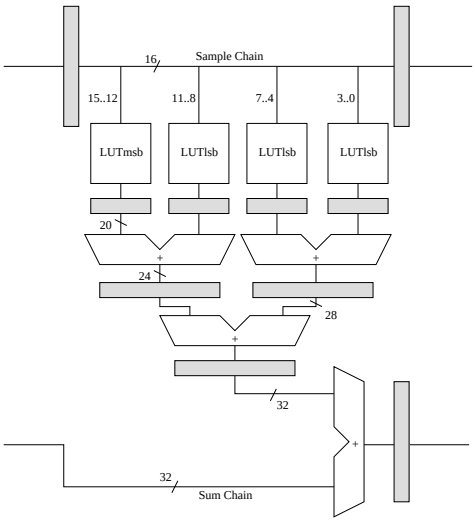

Figure 4. Implementation of the FIR Filter

measure	estimation	realisation
frequency	41.5 MHz	40 MHz
latency	151 ns	125 ns
throughput	664 Mbit/s	640 Mbit/s
CLBs	104	83
pins	50	50

Table 3. Performance for the FIR Tap

data-path part) and an accumulator. The classification vector, from which the features are extracted and the performance matrix is built, is shown next. There is only one LUT accounted since they all work in parallel.

$$\{1, 16, 1, 32, 4, 32, 0, 0, 0, 0, 0, 0, 0, 0, 1, 16, 20, 1, 2\} \quad (9)$$

Since the implementation resembles the parallel pipelined approach, the second column of the performance matrix is compared to the data obtained by running through the design flow with the above implementation. The results shown in table 3 provide a deviation of less than 8.6% between the estimation and the actual implementation on a XC4000E device of Xilinx.

3.2 Motion Estimation

The motion estimation algorithm accumulates the absolute differences between all the bytes of two image frames to be correlated. The implementation shown schematically in figure 5 processes 16 pixel pairs per cycle. The classification vector counts 48 adders of

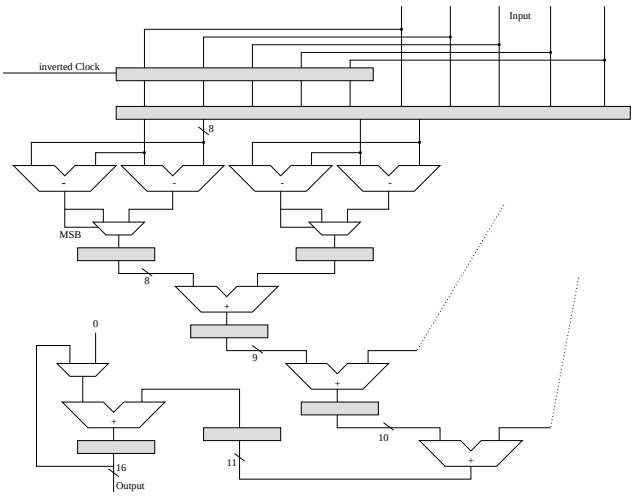

Figure 5. Motion Estimation Implementation

measure	estimation	realisation
frequency	38.4 MHz	40 MHz
latency	666 ns	525 ns
throughput	9.8 GBit/s	10.2 GBit/s
CLBs	297	260
pins	274	146

Table 4. Motion Estimation Performance

various precision and 17 2-to-1 multiplexors. Unfortunately, the implementation is not similar to one of the characteristic approaches, and the results of PPM can not be compared with the realised performance figures. As a work-around, the implementation must be subdivided into the tree-like adder structure and the accumulator block. The former is a parallel-pipelined design (11), the latter is serial-flat (12).

$$\{32, 8, 1, 16, 48, 9, 0, 0, 17, 2, 9, 0, 0, 0, 0, 0, 0, 16, 8\} \quad (10)$$

$$\{32, 8, 1, 16, 47, 9, 0, 0, 16, 2, 8, 0, 0, 0, 0, 0, 0, 1, 12\} \quad (11)$$

$$\{1, 16, 1, 16, 1, 16, 0, 0, 1, 2, 16, 0, 0, 0, 0, 0, 0, 16, 1\} \quad (12)$$

The results of PPM must then be combined in an intuitive way. The lower frequency must be chosen, the latencies are added up. Throughput is computed as data *entering* the circuit per second, so the throughput of the parallel part must be scaled to the lower frequency. Area is added up, and the pin count estimation of the parallel part reflects the total pin count. The results are compared in the table 4.The difference in pin count arises from the fact that PPM cannot model the demultiplexed input pins implemented to save IO pins.

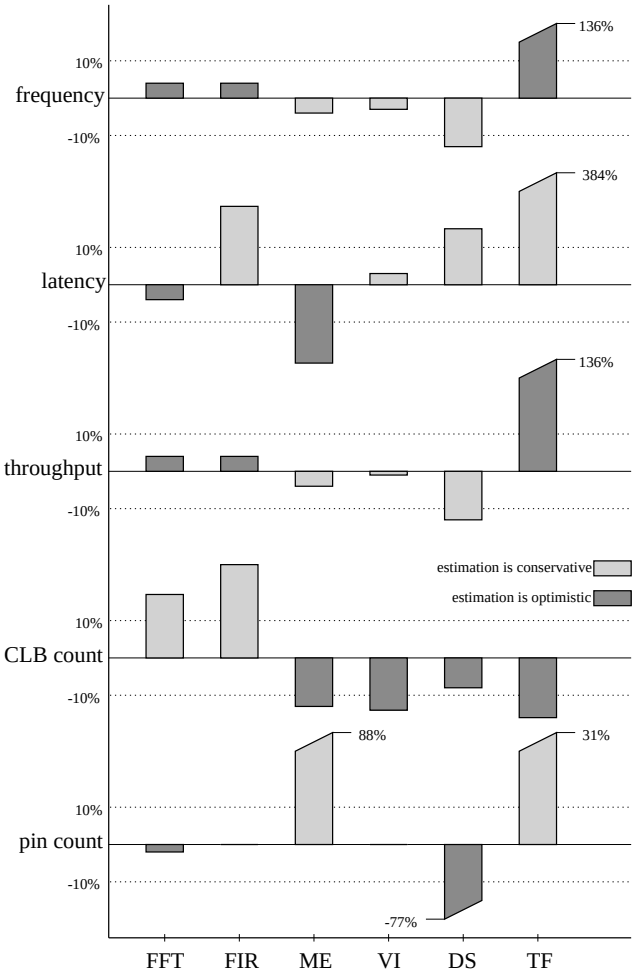

Figure 6. Evaluation of PPM

Neglecting this discrepancy, the maximum difference is 21.2%.

4 Evaluation

As shown in the previous two sections, four additional algorithms are used to evaluate the PPM. These are the fast Fourier transform (FFT), the Viterbi algorithm, CDMA de-spreading and Twofish [2, 1], a block cipher. The full set of six algorithms covers every characteristic implementation, where Twofish comes with the most displaced approach with respect to the implementation space. Figure 6 shows the evaluation of the PPM.

The average error is in the range of 10% or lower, when disregarding the results of Twofish. The estimations for frequency, latency and throughput tend to be more precise than the hardware figures. One of the reasons for the poor quality regarding FPGA area is the fact that CLBs may or may not hide storage elements in functionally used blocks. PPM assumes that all these flipflops are cached in this way.

5 Conclusions

The PPM presented so far is a means for estimating the performance of an algorithm mapped to configurable hardware without actually running through the design process. The precision of the model depends on the accuracy of the several stages involved. The classification step is considered to be the most critical step since the meaning of its single elements must be fully understood. The evaluation of the PPM with six algorithms from different areas shows deviations of less than 20% from the values obtained when realising the implementation. Frequency, latency and throughput estimations are likely to yield better accuracy than the hardware figures. Regular algorithms matching the *data river* structure yield better estimations than fairly irregular and inhomogeneous ones.

The PPM comes with a few assumptions and restrictions the user has to know to be able to apply the model correctly.

- As a consequence of the storage redundancy of FPGAs, the PPM assumes that all registers can be cached in the functionally used CLBs. This may lead to inaccurate results for very storage-intense algorithms or implementations.

- The routing facilities of the hardware are not considered to constrain the performance in any way. For a hardware utilisation of up to 90 % this is considered to hold true for a not too complex circuit.

- The PPM is designed for devices allowing different degrees of parallelism and depth of pipelining. If this is not the case for hardware to which the model is ported, it will not necessarily be applicable anymore, or with some restrictions only.

Due to the limited accuracy, the PPM is not amenable for precise predictions. It can be employed for finding an appropriate implementation to enter the usual design flow. The separation of influences from implementation, algorithm and hardware yields easy portability of the model, so that it can address a variety of architectures once their feature extraction equations are investigated. As a consequence, the PPM may reduce the development time of configurable systems.

References

[1] Counterpane. Twofish, a new block cipher. http://www.counterpane.com/twofish.html.

[2] Bruce Schneier et alt. Twofish: A 128-bit block cipher. Technical report, Counterpane Systems, Minneapolis, USA, 1998.

[3] Tobias Jeger. DSP algorithms on FPGA. Technical report, Federal Institute of Technology Zürich (ETHZ) Switzerland, 2000.

[4] MPEG. MPEG software simulation group. http://www.mpeg.org/MPEG/MSSG.

[5] Xilinx. homepage. http://www.xilinx.com.