A Methodology for Hardware Architecture Trade-off at Different Levels of Abstraction

Claus Schneider

Siemens AG, Corporate Research and Development, ZT ME 5
D-81730 Munich
E-Mail: Claus.Schneider@mchp.siemens.de

Abstract

In this paper a method of architecture exploration and selection is presented. Compared with other approaches, no special tools or modeling languages are needed - instead the models and tools of the ASIC design flow are used. The architecture trade-off process is performed iteratively, and considers information from different levels of abstraction in parallel. At system level, software and behavioral models, which are part of executable specifications are examined to get the necessary top-down information (performance). Bottom-up information (hardware costs) for irregular hardware structures is obtained by generating, analyzing and synthesizing VHDL code at RT-Level. For regular structures, formulas or tables can be used to estimate area and timing. The proposed approach was successfully performed for parts of a multimedia design, where an executable specification (in 'C') was available together with the standard.

1 Introduction

To an increasing degree, ASICs are specified by so-called executable specifications. These specifications consist of pure functional software models as well as of behavioral models comprising clock-related timing. The advantage of this approach is the ability to perform validation (are we building the right product?) and verification (are we building the product right?) in early design stages to avoid long iteration loops in the design process. After the specification, the whole design is modeled again at a lower level of abstraction - the register transfer level. In some cases behavioral synthesis tools can be used to perform this transformation. Often, the models (software, behavior, RTL) are only used for the verification (simulation) or design (synthesis) process, while for architecture exploration, special tools for performance modeling are used. The intention of this paper is to show a way to exploit the models of the design process for architecture trade-offs as well.

1.1 Related Work

Architecture trade-offs can be performed at all levels of abstraction. A classification of design levels related to abstraction can be found in [EcHo92]. For architecture exploration at system level, various tools were developed. A toolset for computer architecture design and simulation, in which the models are described in an object oriented language, is presented in [IHH95]. Another specification-driven design environment [TAB95] for design trade-offs at the architectural abstraction level uses a special Design Specification Language (DSL) as input. The exploration is performed through simulation and behavioral synthesis. Special abstract modeling approaches, such as the use of Petri Nets [AbCo90, Ram93], are also used for early system evaluation. All of the above referenced approaches have in common that a special tool or description language is needed in parallel to the HDL-based design flow to perform architectural explorations. Other approaches only cover a special class of architecture trade-off. In [HoGa95] a method of architectural exploration for data paths with memory hierarchy is presented which allows for performance/cost trade-offs in memory-intensive applications. A memory estimation technique for DSP applications can be found in [VSR94]. Another approach [VaGa95] is based on a control-unit/datapath model to perform incremental hardware estimation during hardware/software partitioning. A generator-based method for the design space exploration in time and area for regular structured logic like mathematical operators is proposed in [JhDu92, GWG93].

1.2 Paper Structure

The next section gives an overview of the iterative architecture trade-off process and outlines the interaction of top-down and bottom-up information. Afterwards the relation of reuse and architecture trade-off and their impact on the modeling style are discussed. The main part of the paper focuses on the different abstraction levels and shows methods of architecture trade-off at system (functional), behavior (synchronization), and register transfer level. Application examples are given for each abstraction level.

2 Overview

Hardware architecture trade-off consists of architecture exploration and selection. During the exploration process, top-down information is collected at system level. For example, software models can be used to perform functional explorations like algorithm selection and to generate data statistics. Average data rates can help to determine the required performance for each subdesign. Behavior models are a good means of examining the queueing behavior of several buffers and the synchronization between functional units. The data distribution functions, calculated by the software model, can be used to stimulate a behavioral simulation to consider the data-dependent processing time.

For architecture selection, bottom-up information is needed to estimate the hardware costs and to evaluate each

ED&TC '97 on CD-ROM

architectural alternative. This information can be obtained by analyzing or synthesizing models at register transfer level, which are parameterizable regarding performance and cost. For regular hardware structures (e.g. adder, multiplier) hardware costs can be estimated by tables or formulas. The exploration and selection is an iterative process, for which top-down and bottom up information is needed in each step (see Figure 1).

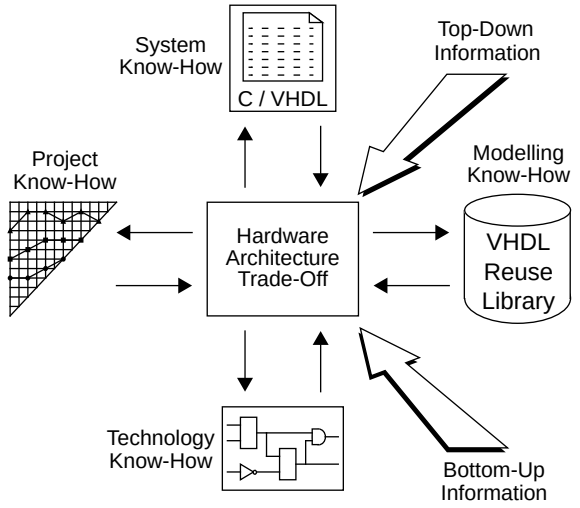

Figure 1: Top-down and bottom-up information

Architecture trade-off is not only a technical, but also a managerial problem. A trade-off between design complexity and development time and risk has to be made. Not in every case is the most sophisticated architecture that meets the technical constraints at the lowest hardware cost the best to meet the time-to-market constraint as well.

Similar to the design process, modeling know-how and reuse are essential parts for architecture trade-off. At RT-Level, VHDL can be used for modeling as well as Verilog. Because of the powerful modeling capabilities at behavior level, VHDL was chosen for the whole design flow.

3 Reuse Modeling for Architecture Trade-off

For reuse as well as for architecture trade-off, parameterizable models are needed. However, the requirements concerning the parameters are different.

The goal of reuse is the integration of frequently used components into many different applications. Therefore the components must have flexible interfaces (e.g. variable port widths, parameterizable control signals, optional input/output registers). In addition, a test bench for the integration test must be available as support for the model. A reuse scenario for the VHDL-based hardware design flow can be found in [PHSR95].

For architecture trade-off, many architectural alternatives must be examined in order to select one for the final implementation. This trade-off has to be performed for each application that reuses the component. In contrast to reuse, the model must be parameterizable for architectural explorations with regard to performance and cost (e.g. parallel/serial, internal processing data width). Another important feature of models for architecture trade-off is the ability to rapidly estimate the hardware costs for different performance values. For regular structures (e.g. adder, multiplier, barrel-shifter) a formula or a table can be supplied with the model [JhDu92]. This is not possible for complex or irregular hardware structures. Here rough estimations can be done by analyzing the VHDL code presynthesis and by counting, for example, the number of states, inputs and literals [BRSW87]. Better cost figures (e.g. area, timing) can be obtained by performing a more or less precise (time consuming) synthesis process.

3.1 Self-generating VHDL Models

During the elaboration of self-generating VHDL models, for analysis and statistics generation, the VHDL synthesis subset is limited too much. For example, text I/O and access types (pointer) are totally unsupported by current synthesis tools. These language features should not be synthesized, but they are very useful in supporting the model and report generation, such as for the following tasks:

- Read a complex look-up table in text format as an input parameter of the model.
- Perform transformations by using dynamically linked lists with access types.
- Write the actual parameter settings to a report file to document the design process.
- Write model-based analysis and hardware cost estimation results.

The long elaboration time and large memory requirements for parameterizable, self-generating VHDL models are another serious problem for architecture trade-off. (Nested) Loops are often used for the calculation of intermediate results during the model generation process. The synthesis tools unfortunately interpret these kind of calculations as hardware and perform loop unrolling.

3.2 External Model Generators

For architecture trade-off, the usage of external VHDL code generators is a good method to work around the limitations of current synthesis tools. With scripting languages like PERL [WaSc] for example, which are used in the design process as well, text processing is very easy because powerful functions (e.g. sort, grep, regular expression matching) are available. Another advantage is that the generated VHDL code is independent of the synthesis tool, because only the synthesis subset is used. While self-generating models are parameterized by generics, parameterization with packages is a better alternative for externally generated models.

4 Abstraction Levels

4.1 System (Functional) Level

In the field of multimedia (e.g. MPEG), 'C'-executable specifications ore often available with the standard. These reference models cover the whole functionality in a sequential description style. They are mostly used to figure out the whole system functionality and the operation of its parts. Another useful application is the generation of test data for lower levels of abstraction (e.g. Behavior or RTL). Due to the high execution speed of software models, a huge amount of data can be processed. By modifying or

extending these models, high-level architecture trade-offs can be performed.

One example of architecture trade-off with software models is algorithm selection with a subsequent compliance test. For the matrix multiplication of the Inverse Discrete Cosine Transformation, a floating point implementation was given by the MPEG executable specification. Several algorithms are known from literature to calculate this transformation. They range from highly regular approaches, with a large number of operations, to irregular ones, with a minimum number of operations. For a hardware implementation not only the number of operations (e.g. ADD, MULT), but also the number of registers and data path multiplexors is important. When implementing different algorithms in software, the required resources (e.g. operations and memories) can be counted and the hardware size can be estimated roughly. But the most important modeling step for this example is the conversion from floating to fixed point arithmetic and the compliance check with the reference model. For the compliance with the standard, a huge number of data blocks must be processed, which can be best done with a software model.

Another example of architecture trade-off at system level is the generation of data statistics. The MPEG video data stream, for example, consists of blocks of up to 64 coefficients of the 2-dimensional Discrete Cosine Transformation (DCT). In the decoder, each coefficient has to be multiplied by two factors to perform the Inverse Quantization (IQ) operation. Due to encoding by the DCT, only a few coefficients of each block are non-zero. Therefore, only a few multiplications of the non-zero coefficients are necessary on average to perform the IQ operation for each block. Figure 2 shows the distribution of non-zero coefficients per block at the input of the IQ operation. A huge number of blocks are not coded (zero coefficients) or contain only a few non-zero coefficients.

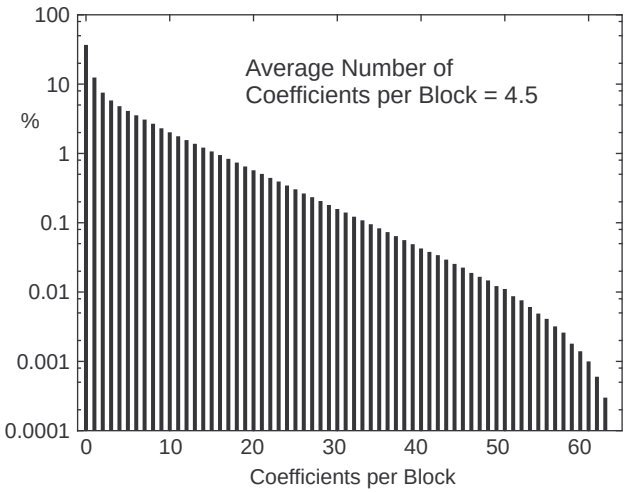

Figure 2: Distribution of coefficients

If the IQ operation should only meet the average performance constraints, an architecture that can perform 9 (2 * 4.5) multiplications in a certain period of time would be sufficient. Depending on the buffer depth before and after the operation, a higher performance of the IQ operation may be necessary. For the operation-schedule/buffer-size trade-off the distribution of the coefficients can be used to stimulate the behavior level simulation model as shown in the next section.

4.2 Behavior (Synchronization) Level

Behavioral models are becoming more and more a part of executable specifications. In [ScEc96] a behavioral modeling approach was presented that consists of separation of synchronization and functionality, which is an ideal structure for architecture trade-offs. An example of a structure for this behavioral modeling style is shown in Figure 3.

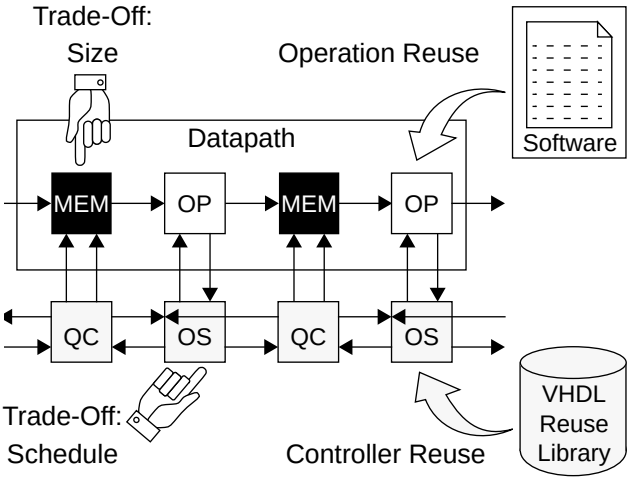

Figure 3: Buffer-size/operation-schedule trade-off

The operations (OP) are decoupled by buffer memories (MEM) to allow parallel processing. Due to the separation of controller and data path, the operations can be modeled sequentially without considering timing. The controllers need only to model the synchronization without any functionality. Therefore, on the one hand, reuse of operations from an existing software model is made easy. On the other hand, standard controllers can be reused from a VHDL library.

An operation is controlled by an operation scheduler (OS) that generates the synchronization signals for the data path as well as for the connected queue controllers (QC). For architecture trade-offs, the model can be parameterized by the operation schedules and the buffer sizes. Additional parameters for the synchronization between operation and buffer are the threshold values for the buffer data request lines.

Another feature of the separation of synchronization and functionality is that in many cases no data path is required for performance analysis and therefore can be omitted in early design stages. The architecture trade-off process as well as the modeling and verification can be done in three steps - see Figure 4.

In the first step only the controllers (e.g. operation scheduler, queue controller) are modelled. The test bench for the verification and analysis of the synchronization consists only of controllers for the handshake with the behavioral model. The operation schedulers (OS) are parameterized by dynamic schedules, generated by subprogram calls. If no data statistics are available, a ramp is used for stimulation.

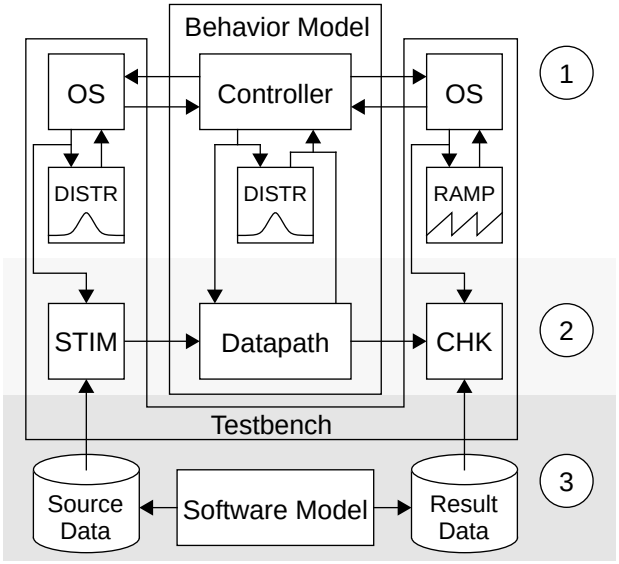

Figure 4: Test bench and behavioral model

Otherwise the data statistics can be used to generate random numbers of this kind of distribution. An example of a tabular distribution procedure is shown in Figure 5. The distribution table is derived from the data statistics for the coefficients from Figure 3.

```
type distr_rec is record
  v : integer;
  p : real;
end record;

type distr_vec is array(natural range <>) of distr_rec;

constant distr_iq_coeff : distr_vec :=
( ( 0, 0.368102), ( 1, 0.492626), ( 2, 0.567988),
  ( 3, 0.626089), ( 4, 0.674154), ( 5, 0.715147),
  ( 6, 0.750619), ( 7, 0.781390), ( 8, 0.808165),
.....
  (61, 0.999996), (62, 0.999999), (63, 1.000000) );


procedure distribute( constant distr: distr_vec;
                      variable seed: inout real;
                      variable x: out integer) is
  variable r: real;
begin
  random(seed, r);
  for i in 0 to distr'high loop
    if r <= distr(i).p then
      X := distr(i).v;
      exit;
    end if;
  end loop;
end distribute;
```

Figure 5: VHDL listing for distribution procedure

In the second step (Figure 4) a dummy data path can be added to the controller. The dummy data path has the same structure as the final one, but the operations simply pass input data to the output without processing. The test bench is extended with stimuli generators that generate special test data (e.g. serial numbers) to analyze the data flow.

In the final step the dummy operations in the data path are replaced by the full functional ones and the test bench is extended to read source and result data for stimulation and result checking. In the behavior model the distribution function for the data path schedule can be replaced by the data-dependent schedule from the full functional data path operation.

4.3 Register Transfer Level

The previous chapters presented methods for architectural exploration at higher levels of abstraction to get the necessary top-down information. To evaluate a selected architecture, bottom-up information like circuit area and propagation delay is required as well for decision making.

In semi-custom design, frequently used blocks - for example, an adder or a barrel-shifter, can be instantiated from a reuse library. These blocks are often characterized by a regular structure. Therefore the bottom-up information of area and timing can easily be estimated by using a formula or a table. On the other hand, it is difficult to make estimations about irregular hardware structures, like look-up tables, which consist of random logic.

The following application example illustrates an approach to quickly estimate the area for a look-up table for different degrees of parallelism. This kind of parallel/serial trade-off is not only interesting for architecture selection on the hardware side, but is also an estimation method for hardware/software co-design. The flow of the estimation process is depicted in Figure 6.

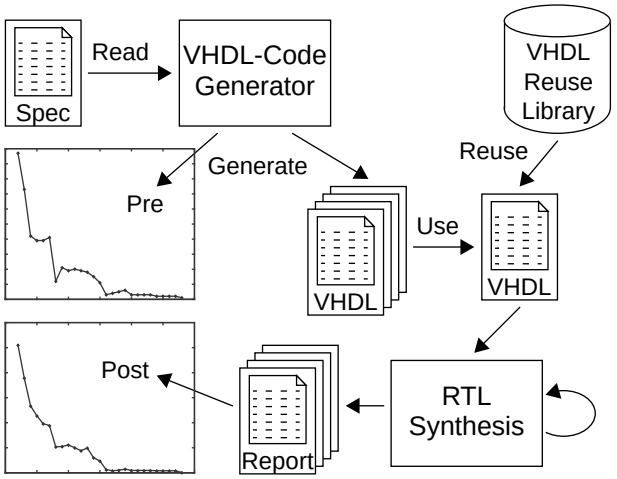

Figure 6: Parallel/serial trade-off for look-up tables

To start out from a specification of the look-up table in textual form, the VHDL code for each processing width is generated. After reading the specification, the table is serialized by splitting and introducing a state vector. The resulting table logic is then minimized and don't care values are mapped. The optimization leads to better estimations, area results and halves the time for logic synthesis. Together with the VHDL code, a statistic is written for each architecture variant (degree of parallelism). Afterwards the statistic values can be displayed graphically (Pre) to assist the architecture trade-off and to help select architectures for the time-consuming logic synthesis process.

In this case the look-up table was in the critical timing path together with a barrel-shifter and therefore they were synthesized together. The look-up table is used from the generated VHDL packages and the barrel-shifter reused from a library. To show that the proposed approach works, all architecture variants from 2-bit to 28-bit were synthesized and the area and timing statistics after the synthesis (Post) were compared with the pre-synthesis results.

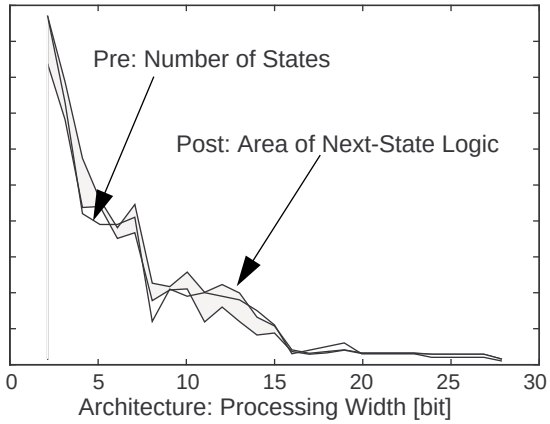

Figure 7: Pre- & post-statistics of next-state logic

Figure 7 shows that there is a close correlation between the number of states of the generated finite state machines and the area of the synthesized next-state logic. The thick line shows the number of states and the grey shaded range the circuit area of the next-state logic for different state encodings (binary and onehot) and different synthesis strategies.

There is also a relationship between the width of the input vector and the area of the output logic. Again the gray shaded range shows the area results of the synthesis runs with different optimization strategies - see Figure 8.

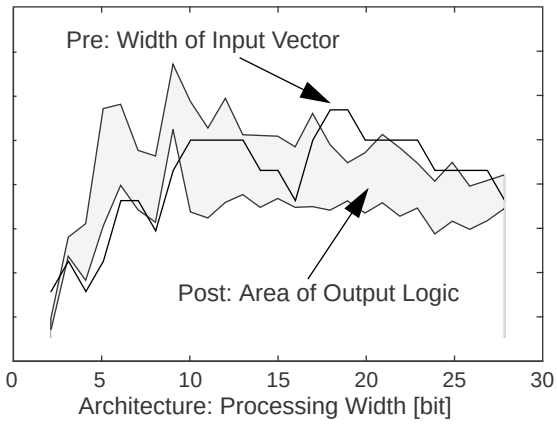

Figure 8: Pre- & post-statistics of output logic

What is remarkable about the results is not only the good match between pre- and post synthesis statistics but also the time to generate both statistics. The pre-synthesis statistic can be generated in less than five minutes, whereas the post-synthesis statistic required more than three days to calculate.

5 Conclusion

An architecture trade-off methodology for different abstraction levels of the ASIC design process was presented. In contrast to tools for architecture exploration that use special modeling languages, the proposed approach is based on the models of the design process only. Beginning from an executable specification, software models are used to generate data statistics to determine the average performance required for each functional unit. The data distribution function, generated by the software model, is then used to stimulate the behavior model for the analysis of the queueing behavior. System level models are used to get the top-down information for the architecture exploration. For the selection of an architecture, bottom-up information is needed to estimate the hardware costs. To obtain this bottom-up information, the usage of RTL models, which are parameterizable with regard to performance and costs, was proposed. In addition to this, modeling styles for reuse and architecture trade-off were discussed. The proposed approach was successfully performed for parts of a multimedia design. For each abstraction level (system, behavior, RTL) the architecture trade-off methodology was illustrated by an industrial application example.

Acknowledgments

I would like to thank Wolfgang Ecker for his suggestions and helpful discussions.

References

[AbCo90] Aboulhamid, M.; Cordeau, M.: System Level Modeling in VHDL using Timed Petri Nets. Proceedings of the EURO-VHDL'90.

[BRSW87] Brayton, R.K.; Rudell, R.; Sangiovanni-Vincentelli A.; Wang, A.R.: MIS: A Multiple-Level Logic Optimization System, IEEE Trans. on CAD, Nov. 1987.

[EcHo92] Ecker, W.; Hofmeister, M.: *The Design Cube - A Model for VHDL Designflow Representation*. Proceedings of the EURO-VHDL'92.

[GWG93] Gasteier, M.; Wehn, N.; Glesner, M.: Synthesis of Complex VHDL operators, Proceedings of the EURO-VHDL'93.

[HoGa95] Holmes, N.D.; Gajski, D.D.: Architectural Exploration for Datapaths with Memory Hierarchy, Proceedings of the ED&TC 1995.

[IHH95] Ibbett, R.N.; Heywood, P.E.; Howell, F.W.: HASE: A Flexible Toolset for Computer Architects, The Computer Journal, Vol. 38, No. 10, 1995.

[JhDu92] Jha, P.K.; Dutt, N.D.: Rapid Estimation for Parameterized Components in High-Level Synthesis, Sixth International Workshop on High Level Synth., 1992.

[PHSR95] Preis, V.; Henftling, R.; Schütz, M.; März-Rössel, S.: A Reuse Scenario for the VHDL-based Hardware Design Flow, Proceedings of the EURO-VHDL'95.

[Ram93] Rammig, F.: Modeling Aspects of System Level Design. Proceedings of the EURO-VHDL'93.

[ScEc96] Schneider, C.; Ecker, W.: Stepwise Refinement of Behavioral VHDL Specifications by Separation of Synchronization and Functionality, to appear in the Proceedings of the EURO-VHDL'96.

[TAB95] Tanir, O; Agarwal, V.K.; Bhatt, P.C.P.: A Specification-Driven Architectural Design Environment, Computer Vol. 28, No. 6, 1995.

[VaGa95] Vahid, F.; Gajski, D.D.: Incremental Hardware Estimation During Hardware/Software Functional Partitioning, IEEE Transactions on VLSI Systems, Vol. 3, No. 3, 1995.

[VSR94] Verbauwhede, I.M.; Scheers, C.J.; Rabaey, J.M.: Memory Estimation for High Level Synthesis, Proceedings of the 31st DAC 1994.

[WaSc] Wall, R., Schwartz R.L.: Programming PERL, O'Reilly & Associates, 1992.