System Level Design, a VHDL Based Approach.

Joris van den Hurk and Edwin Dilling
Product Concept and Application Laboratory Eindhoven (PCALE)
Philips Semiconductors, The Netherlands

Abstract

A hierarchical system design flow was developed to facilitate concurrent development and Time-to-Market reductions. The system design flow provides for codesign of (embedded) driver software, digital hardware, and analogue hardware. The flow starts from a prosaic functional target specification, which is formally recorded in a system algorithm, in VHDL. Through functional decomposition and partitioning, individual parts of the system algorithm are projected onto software, digital hardware, and analogue hardware design flows, all based on VHDL. The hierarchical flow was applied to the design of channel and source decoding systems for Digital Video Broadcast applications.

1. Introduction

The increasingly high requirements on the performance and integration levels of integrated circuits (ICs), in conjunction with a necessity to reduce Time-to-Market require revision of existing design methods. For the past 25 years, a common design method has been “capture-and-simulate” [1]. New, structured design flows allow handling of large device complexities, device performance optimization, and concurrent development [1]-[2]-[3]. Such design methods are based on hierarchical specifications, and verification, and the associated design activities: synthesis and analysis respectively [4]. Hierarchical design methods and “describe-and-synthesize” are therefore gaining popularity in digital IC design [1]-[2]-[3].

This paper introduces the concept of a hierarchical design flow and studies its merits for digital IC design (section 2). To extend these benefits to both embedded software development (section 3) and mixed analogue / digital design (section 4), expansion of the flow was examined in an industrial environment. The results of the explorations are summarized in section 5.

2. Hierarchical digital design flow

2.1. Design flow characteristics

A top-down hierarchical design flow was successfully used for the development of several digital ICs at PCALE [2]-[3]. Figure 1 depicts the (formal) description levels and verification steps in the flow. The description levels

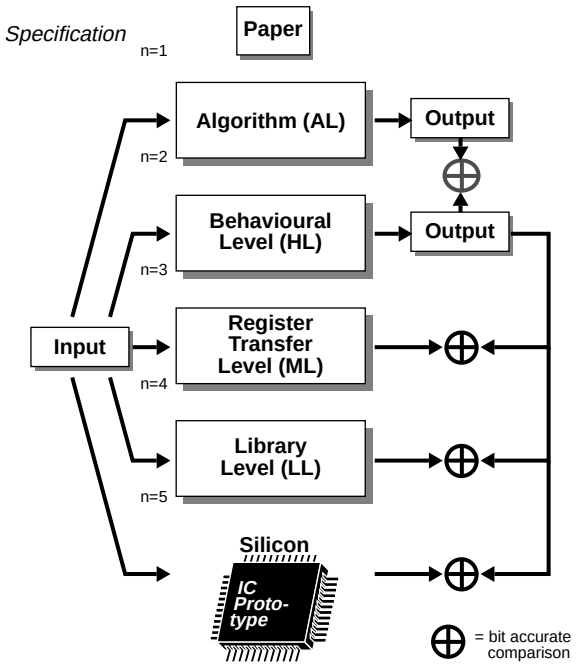

Figure 1. Hierarchical design flow.

are similar to the levels proposed in [1]-[2]-[3], where the feasibility and usefulness of the approach are shown. Each description level in the flow is a partial implementation of the device under development, but also an increasingly more detailed formal specification.

The principal characteristics of each of the description levels are listed in table 1. At later stages of a design, implementation restrictions may require modifications to the device behaviour. When discovered, these modifications are back-annotated to higher description levels. Simulation and verification are repeated. However, iterations rarely span more than two description levels. The extra effort is compensated for by the inherent reduction in the number of design errors.

2.2. Verification

Because for example word length effects result in small functional differences between AL and HL, design verification at this level cannot be bit-accurate (figure 1). Algorithm and formal IC specification (HL) are therefore subjectively compared on the basis of simulations. From

Table 1: Description level characteristics for digital IC design.

Description	Characteristics	Requirements	Applications
Algorithm (AL)	• executable functional description • causal timing, unclocked • abstract data types (e.g. integers, reals)	• implementation / partitioning independent • high execution speeds	Functional system development, verification, and specification
Behavioural (or High Level, HL)	• executable description (function and architecture) • clock related timing • composite bit data types (e.g. integers, bit vectors)	• pin compatible with device • abstract description in VHDL • exact representation of IC functionality • high simulation speeds	*Executable* functional and architectural reference (formal specification) for IC design, and system verification
Register Transfer (RT- or Medium Level, ML)	• executable description (function, architecture and implementation) • clock related timing • composite bit data types	• detailed internal device hierarchy (blocks) • preferably synthesizable VHDL	Input to block based design trajectory
Gate (or Library Level, LL)	• structural description • propagation delay based timing • bit value data types	• VHDL netlist of entire device	Input to layout synthesis, performance and timing analysis, and netlist verification

the HL description onwards verification is bit-accurate. So far, verification was performed through simulations, in future formal verification may be an alternative [5].

Functional verification of hardware prototypes is performed by using the HL description (VHDL) as an exact (pin compatible) reference for an IC (figure 1). Stimuli are applied to the device at full frequency, and its response is recorded and compared (off-line) to HL output (bit-accurately). In addition to functional verification, hardware is tested under various electrical conditions.

2.3. Experimental results

Table 2 summarizes the consistent experimental results of applying the hierarchical flow to several digital IC designs. The gate counts approximate the number of standard cell 2 input NAND-equivalent gates. The development effort (p-mth = person months, mth = months) was measured from the start of design work (setting up an HL description) until the release of first silicon engineering samples (no redesigns included). The redesign modifications represent the design changes required to make the first silicon specification compliant.

Table 2: Hierarchical Design Flow, Experimental Results.

		I	II	III	IV	V	
gate count		25,000 + 12 kbit RAM	35,000 + 23 kbit RAM	60,000 + 43 kbit RAM	35,000 + 46 kbit RAM	40,000 + 1 kbit RAM + 3 DACs	(appr.)
clock freq.		27	27	27	27	27	MHz
design flow		HL → layout	HL → layout	HL → layout	HL → layout	ML → layout	
development effort		28		30		28	p-mth
total elapsed time		14		13		7	mth
redesigns	functional	0	0	0	0	1	modifications
	electrical	2	0	0	0	0	

The HL step was omitted for design V, resulting in an increase in verification complexity, and a decrease in simulation performance. The total design effort was consequently not reduced substantially. Moreover, a functional redesign was required, (not accounted for in the effort in table 2) due to insufficient simulation coverage. Design V therefore demonstrates that starting design at higher levels of abstraction is sensible.

2.4. Advantages

A hierarchical design flow, such as the one previously described, has the following advantages:

- the same formalism (i.e. VHDL) is used for all description levels throughout the design
- concurrent development (engineering) is facilitated
- the risk of functional errors is reduced, early design flaw detection is facilitated, and iteration loops are kept short
- models are easily maintainable for (future) reuse, redesigns are consequently simplified
- Time-to-Market is reduced effectively (by a factor of about 2, according to [1])

3. Hardware / software codesign

3.1. Driver software

Embedded software can be represented in the layered model depicted in figure 2. The bottom layer of the model consists of self-contained components combining hard-

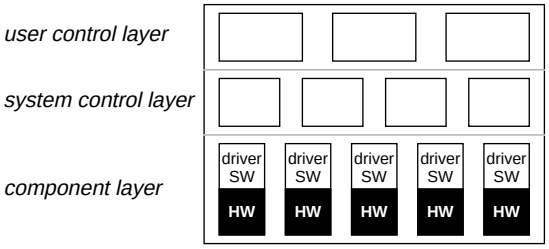

Figure 2. Basic architectural model for embedded software development.

ware, and hardware specific driver software. The component layer moulds the bare functionality of the hardware (HW) into the logical functionality required for software (SW) design.

The two higher layers in the model consist of software only. The system control layer combines the functions of several component layer modules. Such subsystems are the basic abstraction of hardware functionality, arranged according to the structure of the software. The user control layer combines all software functionality and provides a user access to, and feedback from the system.

3.2. Hardware / Software Codesign Characteristics

Hardware / software codesign is a method for concurrent development of individual component layer modules (figure 2), consisting of digital hardware and associated driver software. Concurrent development provides for a reduced Time-to-Market, as hardware and software design are performed concurrently instead of consecutively. Furthermore, the risk of functional errors is reduced, as both hardware and software design benefit from extensive verification, and from close cooperation with the other.

The advantages of hierarchical flows (listed in section 2) inspired the development of a hierarchical hardware / software codesign flow (figure 3). In this flow the hardware column is identical to figure 1. The software description levels however have different characteristics (table 3).

The hardware / software algorithm (AL) is basically not different from the algorithm for digital hardware design. No distinction is made between hardware and software. An instruction-execution based specification style [1] in VHDL is well suited for hardware / software algorithms. Commands issued by higher layer software modules (figure 2) are executed, under the assumption that processing power is sufficiently available [6].

The behavioural description (HL) in the software column consists of algorithmic implementations of driver tasks in VHDL. Software processing power is assumed to be adequately available, dedicated hardware resources are limited. All of the concurrent driver tasks are served instantaneously, upon request [6]. However, communications channels to the hardware, provided by an elementary

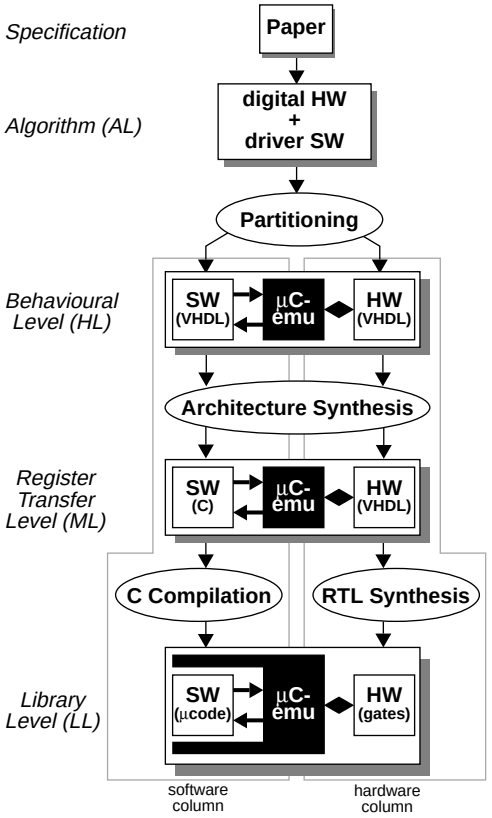

Figure 3. Digital hardware / software codesign.

micro controller emulator (µC-emu), have to be shared between driver tasks.

The ML description contains the same software tasks as the HL, but statements are executed strictly sequentially. A real time kernel (operating system) is implemented to allocate priority to competing driver tasks [6].

The library level description (LL) represents the microcode of the driver software. It is strictly sequential, and ready-to-run on a system micro controller or microprocessor. To allow for performance and timing analysis through VHDL simulations, a vehicle which emulates hardware execution (µC-emu, figure 3) may be needed. The software may subsequently be released for incorporation in a complex suite of system software (as in figure 2).

In literature, several possible hardware / software codesign problem areas are identified. The most relevant issues are tackled in the following paragraphs.

3.3. Cospecification

If an algorithm (AL) is set up as an instruction-execution specification in VHDL, at the start of system development, it can serve as a hardware / software cospecification, provided the description does not prescribe an implementation nor a partitioning (table 3).

At the behavioural (HL) level, hardware and software are specified in separate descriptions, but in the same for-

Table 3: Description Level Characteristics for Software Design.

Description	Characteristics	Requirements	Applications
Algorithm (AL)	• executable functional description • causal timing, unclocked • abstract data types (e.g. integers, reals)	• implementation / partitioning independent • high execution speeds	Functional system development, verification, and specification.
Behavioural (or High Level, HL)	• executable description (function and architecture) • causal timing, clock synchronous I/O to hardware • composite bit data types (e.g. integers, bit vectors) • individual software tasks processed concurrently	• abstract description in VHDL • detailed timing only for communication (I/O) to hardware • high simulation speeds	*Executable* functional reference for software design, and system verification.
Register Transfer (RT- or Medium Level, ML)	• executable, scheduled description (function and architecture) • causal timing, clock synchronous I/O to hardware • composite bit data types • individual software tasks processed sequentially	• high level programming language (HLL, e.g. C), or VHDL description • scheduling based processing time allocation	Input to high level programming language (HLL) compilation
Gate (or Library Level, LL)	• structural description (microcode) • processor clock cycle based timing • bit value data types	• executable on appropriate micro-controller / microprocessor	Input to performance and timing analysis, and design verification

malism (VHDL). The software HL specifies driver behaviour, in a timing causal manner. If hardware and software HL are combined in one VHDL environment (as in figure 3), hardware / software cospecification results at this level also (as in [6]).

3.4. Partitioning and Integration

Architectural synthesis tools [7] may prove useful to support hardware / software partitioning, which has so far been performed manually. As VHDL is the formalism used at the behavioural level, modifications of the partitioning are straightforward (figure 3). Parts of the software description can be cut and pasted into the hardware description, and / or vice versa.

3.5. Cosimulation and Verification

To enable cosimulation and verification, a micro controller emulator (µC-emu, figure 3) is needed to virtually run the driver software on. For behavioural and register transfer level descriptions the emulator is a rudimentary (VHDL) model of the micro controller, implementing clock cycle based I/O protocols only. The VHDL emulator therefore consists of a small set of read and write instructions. For the library level (microcode), a more sophisticated model, performing accurate microcode execution, is required. This model enables cycle based timing analysis.

Hardware / software verification is performed on the basis of VHDL simulations. In our experiments we found that hardware / software verification benefits from concurrent development. The number of hardware and software design errors is effectively reduced. Furthermore, stimuli generation is simplified, as the driver software translates logically abstract, but intelligible instructions into appropriate, but hardly comprehensible communication actions.

After hardware-only prototype evaluation (section 2), co-evaluation is performed. A real time micro controller emulator is connected to the hardware evaluation environment. The emulator runs the driver software, and allows real time interactive software debugging.

3.6. Experimental Results

Hardware / software codesign experiments have so far been restricted to control oriented implementations in which hardware handles real time tasks under software control, with the results listed in table 4. The efforts for the two equally complex devices are different, because several design VI blocks were reused in design VII.

Table 4: Hardware / Software Codesign, Experimental Results.

		VI	VII	
gate count		25,000 +46 kbit RAM	30,000 + 12 kbit RAM	(appr.)
clock freq.		9	9	MHz
design flow		HW / SW codesign	HW / SW codesign	
programmability	protocol	memory I/O, parallel	memory I/O, parallel	
	control registers	900	1852	
development effort		32	24	p-mth
total elapsed time		9	6	mth
redesigns		0	0	modifications

The effort for both devices in table 4 is comparable to

the data in table 2. However, hardware / software codesign in combination with rapid prototyping [8] results in not only engineering samples, but also driver software and CPLD (complex programmable logic device) prototype boards. The early availability of software and CPLD prototypes led to a functional prototype system, even before the IC designs (VI and VII) were transferred to a foundry.

3.7. Conclusions

Present day formalisms such as VHDL and design tools allow effective hardware / software codesign, with the following advantages:

- models are easily maintainable for (future) reuse
- the risk of functional errors in both hardware and software is reduced, and iteration loops are kept short
- concurrent development is supported, and even recommended to increase development speed
- the Time-to-Market for hardware and associated driver software is effectively reduced
- in combination with rapid prototyping, hardware / software codesign allows for early system integration

Most of the hardware / software codesign obstacles mentioned in literature can be overcome fairly easily.

4. Mixed analogue / digital design

4.1. Analogue / Digital Design Flow Characteristics

Traditional analogue design methods focus on the structure (transistor) and geometry (layout) domains. However, mixed analogue / digital coverification (similar to section 3) in these domains is practically infeasible, because circuit simulations are slow. A hierarchical mixed signal design flow (figure 4), which starts in the function domain, for software (SW), digital hardware (DHW), and analogue hardware (AHW), is therefore proposed [11]. The digital hardware and software columns are identical to figure 4, the analogue hardware description levels are defined in table 5.

The algorithm (AL) does not distinguish between analogue hardware, digital hardware, nor software. The analogue behavioural description (HL) is an oversampling based VHDL description of desired hardware behaviour [9]-[10], because current VHDL simulators only support event driven, discrete time simulations.

The analogue medium level (ML) description is a macro based description of an analogue implementation. It may contain macro cells such as operational amplifiers, and filters. Modern circuit simulators (such as MILES, [12]) allow for ML cosimulation, through VHDL simulator interfaces.

The library level (LL) description is the transistor implementation of an analogue device. Simulation of analogue behaviour at this level is accurate, but extremely slow, and therefore used to verify specific implementation aspects such as timing and signal integrity only.

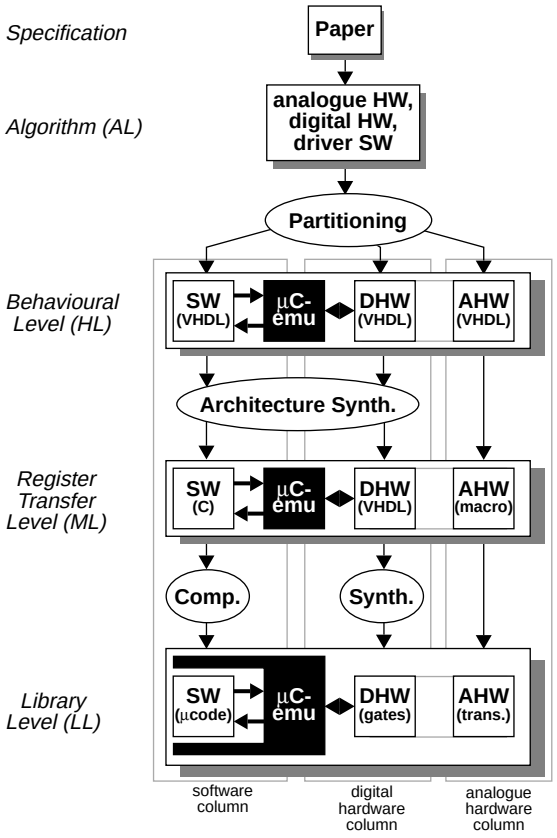

Figure 4. Mixed analogue / digital design flow.

4.2. Cosimulation and verification

Designers working in either the digital hardware, or the software column, are uninterested in detailed analogue behaviour. They use the analogue HL description as a functional reference. Analogue designers in turn use the digital hardware and software HLs.

4.3. Experimental Results

Mixed analogue / digital design flow experiments have yielded the results in table 6.

4.4. Conclusions

The proposed analogue modelling is feasible, and the benefits of a hierarchical design flow apply to analogue design as well [11]. However with the future extension of VHDL towards the analogue domain, VHDL-A, more detailed descriptions of the analogue behaviour can be made. This may facilitate the incorporation of parasitic analogue behaviour at the behavioural level, providing more accurate system models.

5. Discussion

Hierarchical design flows for digital IC design exhibit

Table 5: Description Level Characteristics for Analogue Design.

Description	Characteristics	Requirements	Applications
Algorithm (AL)	• executable functional description • causal timing, no clocks • abstract data types (e.g. enumerated)	• implementation / partitioning independent • high execution speeds	Functional system development, verification, and specification
Behavioural (or High Level, HL)	• executable description (function and architecture) • timing related to oversampling clock • real data types	• pin compatible with device • abstract description in VHDL • representation of IC functionality • high simulation speeds	*Executable* functional reference (formal specification) for IC design, and system verification.
Register Transfer (RT- or Medium Level, ML)	• executable description (function, architecture, and implementation) • continuous time • real, or physical data types	• detailed internal device hierarchy (macroblocks) • includes parasitic characteristics	Input to block based design trajectory
Gate (or Library Level, LL)	• structural description • propagation delay based timing • physical data types (e.g. voltage)	• transistor netlist of analogue part of the device	Input to layout synthesis, performance analysis, and design verification

distinct advantages. If hardware / software codesign, and

Table 6: Mixed Analog / Digital Design, Experimental Results.

		VIII	IX	X	
gate count		1,000 + 2 ADCs, 5 DACs, 4 Opamps	22,000	45,000 + 2ADCs, 3 DACs, 3 Opamps	(appr.)
clock freq.		60	14	28	MHz
design flow		Mixed Analog / Digital	Mixed Analog / Digital	Mixed Analog / Digital	
program-mability	protocol	-	-	I^2C	
	control registers	-	-	250 (est.)	
development effort		24	18	30	p-mth
total elapsed time					mth
redesigns		0	0	0	modifications

mixed analogue / digital design flows are set up hierarchically, they benefit from, at the very least, the same advantages:

- a small but comprehensive set of descriptions
- a single formalisms (VHDL) for higher levels of design abstraction
- concurrent development
- design error risk reduction, resulting in a significant decrease in the number of redesigns
- Time-to-Market reduction

References

[1] H.S. Juan, N. Holmes, S. Bakshi, D. Gajski, "Top-Down Modelling of RISC Processors in VHDL", *Proc. European Design Automation Conf.*, pp. 454-459, September 1993.

[2] J.A.A.M. van den Hurk, "Evaluation of the PCALE VLSI Design Flow for HDTV ICs", Instituut Vervolgopleidingen, *Eindhoven University of Technology,* The Netherlands, ISBN 90-5282-189-5, April 1992.

[3] W.P.G. Crooijmans, "From Algorithm to VLSI for HDTV", *IEEE Trans. on Cons. Electr.*, vol. 37, no. 4, pp. 933-936, November 1991.

[4] B.J. Hosticka, W. Brockherde, R. Klinke, R. Kokozinski, "Design Methodology for Analog Monolithic Circuits", *IEEE Trans. on Circ. and Syst.*, vol. 41, no. 5, pp. 387-394, May 1994.

[5] S. Malik, A. Wang, R. Brayton, A. Sangiovanni-Vincentelli, "Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment", *Proc. IEEE ICCAD*, pp. 6-9, November 1988.

[6] W. Ecker, "Using VHDL for HW / SW Co-Specification", *Proc. European Design Automation Conf.*, pp. 500-505, September 1993.

[7] L. Stok, "Architectural Synthesis and Optimization of Digital Systems", Ph. D. Thesis, *Eindhoven University of Technology*, The Netherlands, ISBN 90-9003966-X, July 1991.

[8] J.A.A.M. van den Hurk, R. Schutte, "Prototyping for MPEG2 with Programmable Logic Devices", *Proc. ProRISC & IEEE Workshop on Circuits, Systems, and Signal Processing*, pp. 145-152, March 1995.

[9] B.R. Stanisic, M. W. Brown, "Behaviour Modeling of Mixed Analog Digital Circuits", chapter 3 in *Applications of VHDL to Circuit Design*, R.E. Harr and A.G. Stanculescu, eds., Kluwer Academic Press, 1991.

[10] B.R. Stanisic, "Modeling of Analog Digital Loops in VHDL", chapter 4 in *Applications of VHDL to Circuit Design*, R.E. Harr and A.G. Stanculescu, eds., Kluwer Academic Press, 1991.

[11] E.R. Dilling. "Design and evaluation of a mixed signal design flow for digital video applications", Instituut Vervolgopleidingen, *Eindhoven University of Technology,* The Netherlands, ISBN 90-5282-438-X, January 1995.

[12] MILES user manual, *Philips Electronic Design & Tools*, 6th edition, June 1993.