Practical Concurrent ASIC and System Design and Verification

Ian Gibson and Chris Amies

Canon Information Systems Research Australia
PO Box 313, North Ryde, NSW 2113, AUSTRALIA
gibbo@research.canon.com.au and chrisa@research.canon.com.au

Abstract

This paper describes the evolution of a design and verification methodology successfully used to develop advanced ASICs as components of multiple new commercial products. The ASICs are typically large, high speed, algorithmically complex and implement novel functionality. The ASIC development process is driven by the commercial pressures of low cost and short schedules of multiple projects. It is carried out using a team of designers of varying experience including new staff. The dual emphasis of our methodology is maintaining fine control over the design and verification process, together with full independent cross verification as an integral part of the entire ASIC and system development process.

1 Introduction

A successful design methodology can be defined as one that best delivers the final product with least cost. Typically, in our environment a dominating constraint is meeting the delivery date. This implies two key issues that become paramount in formulating a system design methodology: design correctness and control over the design process.

We are generally operating under the premise that there is no time for ASIC re-fabrications, or any significant design iteration at any phase of a project. In the event that an ASIC or other key component has a fatal flaw or functional shortcoming we have in effect completely failed at our task. Unexpected delays can also prove fatal to the goal of on-time delivery. Any tool, technique or process can only be incorporated into our design flow if sufficient control over it can be maintained.

This paper describes the methods and procedures that we have used to rapidly develop novel complex systems, taking designs from conceptualisation through to prototyping and pre-production in a relatively short time. The emphasis is on how a formalised ASIC design and verification flow forms the basis for a complete concurrent system design and verification process.

2 The Design Environment

Canon Information Systems Research Australia (CISRA) develops a range of digital image processing and management products. Systems range from pure software packages such as image libraries, through to embedded systems such as high speed colour printers and video animation systems. Typically, all constituent sub-systems, from user interfaces, application and system level software, through to digital imaging output devices and their controllers, are implemented for the first time concurrently.

The electronic sub-systems of such systems tend to have tight real time constraints, implement complex and novel algorithms and are heavily reliant on semi-custom components - typically ASICs. In turn these ASICs tend to be complex, utilise the latest commercially available semi-custom semi-conductor technology, and are almost always implementing algorithms in hardware for the first time. The ASIC design group, currently consisting of approximately 20 people, has designed about a dozen such ASICs over the last five years. These ASICs form the basis of six different product families.

The latest ASIC developed using this methodology is X-17. X-17 is an embedded graphics co-processor for a range of high resolution colour applications. X-17 is an 80Mhz device implemented using a 0.5µ gate length, standard cell technology of approximately 1 million transistors.

X-17 is a central component of multiple new products, some of which were developed in parallel with the ASIC development. These products ranged from a low cost PC plug-in board to provide publication quality colour document composition capability based on an advanced page description language, through to a multi-board system driving a high quality scanning and printing system. Other digital image systems will be developed based on

ED&TC '97 on CD-ROM

X-17. These systems are characterised as having high resolution, high output quality, high speed, and are cost sensitive.

X-17 is designed to fit a wide range of price-performance targets, and to form the core of multiple systems. It typically operates as a loosely coupled graphical co-processor interfacing to a host, primarily via a PCILocal Bus, although it can stand alone independent of any host. It directly interfaces to a range of colour I/O devices such as scanners and printers via a configurable on-chip peripheral interface. X-17 implements a novel co-processor architecture allowing very efficient interfacing to a wide range of hosts. It is a SIMD (Single Instruction Multiple Data) processor, optimised for a core set of graphical operations that form the bulk of the computational load for high resolution digital colour systems. X-17 is also a real-time device, providing guaranteed worst case performance for its operations, thereby facilitating its use in a range of embedded systems.

3 Evolution of a Design and Verification Methodology

CISRA's design methodology has evolved over the years as tools and technologies have evolved [4].

There has been much recent literature describing successful and efficient verification strategies for large complex circuits [3], [5], [8], [7] and [2]. The focus has been on applications where the functional space is well understood and the specification well defined (such as is the case for microprocessors) and the problem is one of how to cover the space more effectively.

Our problem is slightly different. Because we are developing entire systems in parallel, we are characterising the problem space at the same time as formulating methods for adequately covering that space. Specifications drift during the course of development due to various factors. For example, while the ASIC design team is verifying that the ASIC gives the correct answer as per specification, the system designers and architects are trying to verify that the specification is what they wanted after all. The algorithms implemented in the ASIC are verified concurrently and incrementally with the ASIC development.

Many design methodologies targeted at verification of semi-custom based systems tend to use rapid prototyping in other programmable devices such as FPGAs [6]. Our experience has shown that the time and effort involved prototyping in this manner, coupled with the inability to achieve complete functional and timing coverage renders this style of verification of minimal use.

We have found that unless a given technique allows us to maintain tight control over the process, and the process is reliable, re-producible and independently cross-verifiable then it doesn't actually improve our ability to carry out the design and verification process on time.

Possibly the single most important trend in the development of our design flow is the increased adoption of standards. In particular, the standardisation of HDLs (Hardware Description Languages) has been the basis for many of the improvements in the level of automation possible in the design flow. This standardisation has been a pre-condition for bringing many of the design automation advances into the commercial world - from logic synthesis, simulation and specification through to automating such mundane but critical tasks such as carrying out a design sign-off to a vendor.

4 A Practical Concurrent Design and Verification Flow

Typical systems being developed consist of in-house authored art, user interfaces, application software, driver software, hardware subsystems (circuit boards) semi-custom components and input/output devices (printers and scanners). Usually, all subsystems are developed in parallel. We rarely have the luxury of sufficient time for a sequential design process. Our focus here is on the verification of the co-designed hardware and software subsystems at the core of our target system. We follow a general principle in verifying that entire systems will work correctly together, and that principle is independent cross verification.

The development of each of the major adjacent components in the system is performed by two separate means, either by separate groups of designers, or designers and an automated process. By the semi-associative nature of cross verification, our design flow provides a means for ensuring that the production hardware and software will operate correctly, and will operate correctly together.

The *Experimental Software Model* is used to develop and refine algorithms and techniques. By its nature, it can be developed and adapted quickly and provides a solid and tangible link from the researcher into the design flow.

The *Hardware Emulation Software* is the key element that binds together the cross verification chain. It is typically written in C or C++ by a small group of programmers that do not participate in other aspects of the design. This software emulates each of the necessary hardware sub-components, providing a bitwise accurate (but not timing accurate) model at various levels of hardware - typically it provides a bit accurate model of any ASICs in the system, as well as at the pins of any significant internal ASIC submodule boundaries. The hardware emulation is the primary basis for verifying the work of the

ASIC developers and provides input functional test vectors and reference output vectors.

After any specification work, developing the *Abstract Hardware Model* is the first step in the design process for the ASIC designers, and is implemented in the HDL being used on the project. This model bootstraps the design process by hiding complexity; provides a means of insulating the design process from the final implementation technology issues and is written in a simulation efficient manner and as such can be effectively used the basis for simulation based hardware verification. The Abstract Hardware Model is also bitwise accurate, and is timing accurate on a cycle by cycle basis. This model is maintained throughout the design process, and the bulk of functional verification through simulation is performed on this model.

Our experience has shown that the most time consuming part of verification through functional simulation is in generating the input vectors and checking the output vectors. Simulator plus platform technology has improved at a rate comparable to that of ASIC technology so that although our systems become more complex they simulate just as quickly.

The use of a universal testbench environment allows tests to be written at the top level, executed on the hardware emulation software and extracted for any module or sub-module. This makes functional test vector generation easier.

There are often several *Physical Hardware Models*:

The "Logical" model is the source for logic synthesis and is written in the same HDL as the Abstract Hardware Model - these two can be interchangeable in system level simulations. One of the major challenges that was initially faced with the introduction of logic synthesis was in maintaining sufficient control over the process. Several factors, including tool immaturity, quality control of technology libraries and specific platform dependence all conspired to make the logic synthesis process experimental yet not reproducible. The logic synthesis process is almost entirely opaque to the user - they have very little direct influence over the various optimization algorithms that are applied. To solve this problem we use only a very small subset of the synthesizable HDL language - the Logical model is written from a small set of templates. This set of templates includes various state machine varieties, code to produce synchronous and non synchronous circuits etc. The output from these templates can be characterised against a new technology very quickly. The Logical Model is verified against the Abstract model by using the same test suite as was used on the Abstract Model. This task typically only requires CPU time and not significant design time.

"Gate level" models: the X-17 ASIC actually required *four* different but equivalent gate level models to perform the necessary design and verification tasks of floor-planning; static timing analysis; scan-insertion and test pattern generation; functional verification through simulation and vendor sign off. Functional simulation of these models is very CPU intensive but fortunately very little functional verification is necessary. The bulk of the ASIC design functional verification is performed on the Abstract Hardware Model and on the Logical Model. Each of these models is produced via an automated means and so sufficient functional simulation is only necessary to verify that the translation process is correct.

4.1 Production Hardware and Software

The Production Software is verified using conventional Software Engineering methods, and is also cross verified by running it with the Hardware Emulation Software.

Once the production hardware (ASICs and circuit boards) is constructed, it is cross verified with the Hardware Emulation Software. This cross verification is carried out at all levels down to ASIC sub-module boundaries. All ASICs are built with internal bypass paths or other test structures to enable full access to the pins of each sub-module at full speed. All functional tests (and more) that were carried out on each model of each ASIC submodule are carried out on the actual submodule in a separately constructed test board. In parallel with this module level verification, the ASIC itself is verified as one component of each concurrently developed Production Hardware System.

4.2 Independent Cross Verification

By far the most effective means for verifying a particular step in the design process, is to verify it using some independent means. Cross verification has several key characteristics that are critically important to our design flow. It not only provides a mechanism for verifying a particular design process but it also verifies any assumptions that may have been made prior to the design flow. This is an important issue as these initial assumptions are often the most difficult to verify. Independent cross verification is also a visible and understandable process.

Within the context of the ASIC development, cross verification is carried out both "vertically" and "horizontally". Vertical cross verification involves cross checking the ASIC submodule functionality with the Hardware Emulation Software. Horizontal cross verification is ensuring that each of the submodules within the ASIC operate correctly together - without the necessity of simulating them together.

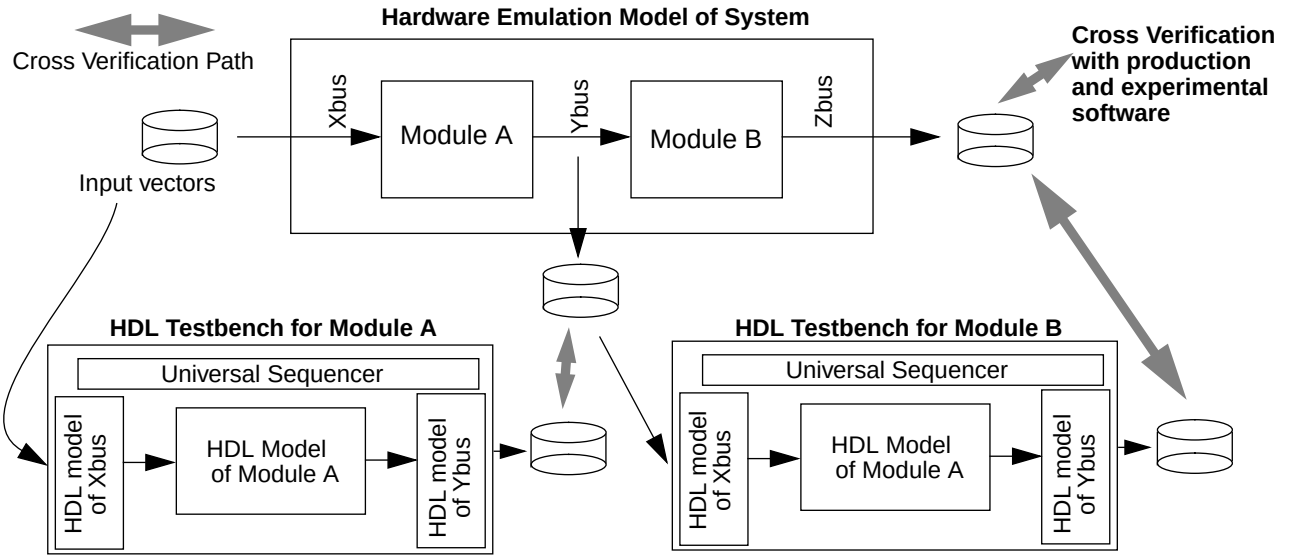

Figure 1 Cross verification environment

Horizontal Verification is carried out at two levels: timing and functionality. Consistent timing is guaranteed through the simple method of maintaining a central repository of all module boundary-level timing and load information, and this repository is parsed to automatically set up any synthesis constraints. By leaving suffi.cient margin on inter-module signals we can guarantee that timing is correct by construction. Horizontal verification of functionality involves cross checking that interconnect protocols are met. This is achieved by restricting inter-module connectivity to using a small number of standard bus architectures, and then writing simple bus models for these. Each module can then be validated in a standard testbench environment, with functional verification vectors and bus protocol verification models provided by third parties

4.3 Common, Understandable and Reproducible Methods

All design and verification flows are automated. Several formalisms for specifying design flows have been published recently [11], [12]. Most major EDA companies offer some sort of workflow management tool. We find that any reliable mature dependency tracking utility is adequate - we use the old Unix favourite *make*. Every aspect of the design can be (re)produced via a make target. As a side effect of this, we tend not to make heavy use of the Graphical User Interfaces (GUIs) provided with most tools. GUIs are used as a last resort when attempting to diagnose some problem (as in functional debugging) or to intervene in some automated process (such as logic synthesis). Another side effect of this is that each invocation of a design tool is for a shorter period leading to a reduced need for multiple licences. Considering the substantial cost of EDA tool licences, this leads to significant cost savings.

All aspects of the design flow from documentation through to HDL coding, constraint specification and test generation are template driven. This makes it easier for inexperienced designers to come up to speed and for designers to help out in other parts of the design as becomes necessary.

Design flow automation is essential for many reasons. Specification drift often means that incremental design changes occur late in the day. We can accommodate these changes relatively painlessly. Tools and technologies drift over the course of a design. Over the course of the X-17 project, the technology libraries changed versions twice, each time requiring reiteration of part of the design process. These changes had minimal impact on the design time. Most EDA tools (logic synthesis, timing analysis, ATPG tools etc.) can be very complex to use and take a substantial time to master. It is a better use of resources to have one or two team members thoroughly knowledgeable on a particular method or tool and to encapsulate this knowledge for the use of others. New and inexperienced team members can be bootstrapped up to delivering productive contributions very quickly. One of the factors in the design process that we cannot gain control over is the quality control of suppliers - tool and technology vendors. A common, automated design flow provides the best infrastructure to build in checks into the design flow as soon as any external issues such as library faults or tool bug workarounds are uncovered.

5 Formal Verification Methods versus Simulation Based Verification

Formal verification methods have been used to verify complex systems at least experimentally [1], and are also starting to be applied to real commercial developments [10]. While formal methods have made it into the commercial world via tools, they have tended to be used for simple constrained tasks such as comparing two different circuit representations. We have not adopted any traditional formal methods in our methodology for several reasons. We are rarely in a position to formally describe our design space - we are typically verifying the underlying design assumptions in parallel with the design process. The development of a distinct abstract hardware model gives us the ability to carry out full simulations. A full set of functional tests of X-17 can be carried out over a period of about two weeks using our network of Sparc-10 and Sparc-20 workstations.

6 Evaluating the design and verification flow

Although the principles of the design and verification flow are stable, actual details are dominated by what tools are available and the implementation technology of the day. We also find that the project design cycle is of the same order as the semiconductor and EDA tool technology cycle - each new project must adapt the methodologies quite radically to encompass these new technologies. It thus becomes very difficult to experiment, and to learn by mistakes. How then do we characterise how well we have done? One measure of course is how many of our ASICs have met the primary objective: being correct first time and on time. Of the eleven ASICs developed over the last five years, one required a re-fabrication and three overran their delivery date.

7 Summary and Conclusion

The key to the successful commercial development of complex and sophisticated ASIC based products is in delivering product on time and within budget. We have developed an evolving design and verification methodology to enable us to adopt new techniques as they mature and enter the commercial arena, and to deliver new complex systems efficiently using a relatively inexperienced team.

Our methodology, which enables us to design and verify most system sub-components concurrently is based on few key principles: as tight a control as possible is maintained over the development process; each step is made as simple and transparent as possible; common, automated methods are imposed throughout the design team; each step must be independently cross validated; successful management of complexity and quality forms the basis of carrying out the design efficiently. These principles must be satisfied before any new design automation tool, technique, method or process can be adopted into our development flow.

8 References

[1] Derek, L Beatty and Ranald E Bryant, “Formally Verifying a Microprocessor using a Simulation Methodology”, *Proceedings of the 31st Design Automation Conference*, San Diego, June 1994.

[2] Francoise Casaubieilh, et al, “Functional Verification Methodology of Chameleon Processor”, *Proceedings of the 33rd Design Automation Conference*, Las Vegas, June 1996.

[3] J Freeman et al, “The 68060 Microprocessor Functional Design and Verification Methodology”, *Proceedings of the Design SuperCon ‘95 On-Chip Design Conference*, Santa Clara, March 1995.

[4] Ian Gibson et al, “Practical Industrial Use of HDLs for the Development of Large High Speed ASICs” *Proceedings of the Asia Pacific Conference on Hardware Description Languages, Standards and Applications*, Brisbane, December 1993.

[5] Michael Kantrowitz and Lisa M. Noack, “I’m Done Simulating; Now What? Verification Coverage Analysis and Correctness Checking of the Decchip 21164 Alpha Microprocessor”, *Proceedings of the 33rd Design Automation Conference*, Las Vegas, June 1996.

[6] P.H. Kelly et al, “Rapid prototyping of ASIC Based System”, *Proceedings of the 31st Design Automation Conference*, San Diego, June 1994.

[7] Val Popescu and Bill McNamara, “Innovative Verification Strategy reduces design cycle time for high end sparc processor”, *Proceedings of the 33rd Design Automation Conference*, Las Vegas, June 1996.

[8] C. Roth et al, “The PowerPC 604 Microprocessor Design Methodology”, *International Conference on Computer Design*, 1994.

[9] Alberto L. Sangiovanni-Vincentelli et al, “Verification of Electronic Systems”, *Proceedings of the 33rd Design Automation Conference*, Las Vegas, June 1996.

[10] Mandayam Srivas and Steven P. Millar, “Applying Formal Verification to a Commercial Microprocessor”, *Proceedings of the 1995 IFIP International Conference on Computer Hardware Description Languages*, Japan, August 1995

[11] Peter R Sutton and Stephen W Director, “A description language for Design Process Management”, *Proceedings of the 33rd Design Automation Conference*, Las Vegas, June 1996.

[12] Marina Zanella, “Principles of Design Methodology Management for Electronic CAD Frameworks”, *Proceedings of The European Conference on Design Automation*”, Brussels March 1992.