Accelerating Concurrent Hardware Design with Behavioural Modelling and System Simulation

Allan Silburt, Ian Perryman, Janick Bergeron†, Stacy Nichols, Mario Dufresne, Greg Ward

Bell Northern Research
P.O. Box 3511, Station C
Ottawa, Ontario, Canada K1Y 4H7

†AnalySYS Inc.
25 Loiselle St. Suite 201
Embrun, Ontario, Canada K0A 1W1

Abstract—This paper describes a functional hardware verification methodology for ASIC intensive products. It spans the ASIC, board, and system level, enabling simulation of the design concurrent with ASIC and board development. The simulation strategy relies on rapid development of behavioural models of ASICs to enable work to proceed in parallel and to achieve the necessary simulation efficiency. The results from a project on which the methodology was used are presented. The process provided early visibility of over 200 issues in the system of which 32 were critical to the successful conformance and timely completion of the project.

I. GOALS - DESIGN/VERIFICATION PROCESS

This paper describes a functional design and verification process for new products whose H/W complexity is largely embodied in a set of newly developed ASICs. The goal was to create a process that supports concurrent engineering of ASICs, software and circuit boards together to achieve a "right first time" product. Since the ASIC design interval was the largest component of the project schedule's critical path, design recycles had to be avoided at all costs.

The product on which the process was proven in required the development of 8 ASICs ranging in size from 20 to 70K gates (plus embedded RAM). A minimal system is comprised of 2 Line Interface Modules (LIM's) and one Switch Module (SM). The LIM consisted of 6 boards and a backplane. The most complex LIM circuit board contained 8 instances of 4 new ASICs. The SM contains 4 boards, and a combination backplane-switch matrix. The most complex board on the SM contained 20 instances of 2 new ASICs.

It has been common lore throughout the industry that ASICs in most new products have experienced at least 1 iteration. There are numerous reasons for reported chip iterations that can be classified into 3 main categories:

- specifications not conforming to system intent
- errors in interpreting or implementing the specification
- changes to the product specification

Our methodology addresses the first point by providing a simulation environment for a given ASIC that includes its system environment while its peer ASICs are still in development. This raises the confidence in the validity of a specification since it verifies complex interaction between components. The method addresses the second cause by providing an independent implementation of each ASIC specification as an executable model. The ambiguity of a natural language is removed when a working system must be created. Behavioural model generation is rapid and thorough in its coverage of functional detail.

The third cause of chip iterations is much more difficult to address since it brings in the dynamic nature of the product marketplace, its perception by the marketing groups and the corporate organizational and funding structure in which the development group exists. Here, we can only claim that by assisting in shrinking the system design interval we contribute to more rapid realization of the initial product concept. In a dynamic market environment, the longer the development interval, the less likely a stable product specification will endure from concept to completion.

II. WHAT IS A BEHAVIOURAL MODEL?

The value of hardware emulation [1] and large scale simulation in detecting design errors during product development is well established. Recent work has shown how RTL (register transfer level) models of ASICs can be totalized for this purpose [2] [3]. The cornerstone of the hardware system modelling methodology we describe here is the development and exploitation of behavioural models for each ASIC, concurrently with its implementation. We define a *behavioural model* as one which is not synthesizable and distinct from an RTL model in the same hardware description language as summarized in Table I.

Table I: Behavioural vs RTL model

	Behavioural	RTL
View	Black-box functional	Implementation
Synthesis	Non-synthesizable	synthesizable
Timing	Can be added, not derived	Clock-true timing
Data	Can use complex types	Bits, bit vectors, integers
Language	Full language	Subset of language
Speed	10x RTL	10x gates
Effort	4-6 man-weeks	9 man-months
Structure	Arbitrary	Follows implementation
Accuracy	Passes >95% of ASIC conformance tests	Passes 100% ASIC conformance tests

Because behavioural models can be rapidly developed and simulate significantly faster than RTL models, they enable an independent and thorough audit of the ASIC specification. System simulations can then proceed concurrently with the ASIC development, thus highlighting problems early in the design process.

32nd ACM/IEEE Design Automation Conference ®

On this project, an ATM based product, architectural differences between the LIM and SM led to significant differences in the verification and modelling methodologies used. Because the LIM processes ATM cells as units, the behavioural models were accurate at the cell level (i.e. the input/output timing within a cell was accurate at the bit and clock level, but the cell latency through an ASIC model did not necessarily match its implementation). However, because the SM operated on fractions of cells and on time slices, the models had to be clock and bit level accurate to data latency on the chip.

To allow seamless replacement of a behavioural model with its corresponding RTL model, the former offered a pin-true top-level interface identical to the RTL. Timing annotations such as setup and hold checks or output settling times could be added if required.

Beyond the top-level interface, a behavioural model was not required to maintain any similarities with the RTL. They typically were composed of a single module or entity/architecture, using instantiations only to ease maintenance when functionality was replicated. The internal processing was performed on high-level data types, constructed from the bit-level input pins, processed in zero-time then formatted and sequenced for the bit-level output pins. Packages were developed for standard data structures and operations. Figure 1 shows an example of using a function of the *ATM_CELL* package in Verilog. A rich library of utilities such as these simplified the development and maintenance of the models.

```
ATM_CELL MY_CELL();//Instantiate cell struct
initial
begin
    MY_CELL.RANDOM;//random payload data
    MY_CELL.CORRUPT_HEC;//force HEC error
end
```

Figure 1: Using the Verilog ATM_CELL package

Behavioural models are inherently more efficient to simulate than RTL models because of the event-driven simulation technology used and the RTL modelling style required by the synthesis subset. RTL models are composed of many small concurrent processes of two types:

1. combinatorial processes evaluated whenever one of its input signals changes. These can be transient and hence re-computed several times within the same simulation time slice or clock cycle.
2. sequential processes evaluated whenever the clock changes. This occurs whether or not the inputs used to compute the next values of the registers have changed since the last clock cycle.

Except for the input and output data formatting sections, behavioural models tend to be asynchronous, evaluating processes only when required. They are composed of few (but large) processes which require little overhead for complex operations. This difference translates into better simulation performance as shown in Section V.

III. Verification process and team organization

An overview of the verification process showing the concurrent ASIC and board development along with simulation efforts is shown in Figure 2 and Figure 3. Behavioural models were developed for each ASIC following the release of its specification. These were carried out by specialists from a core group who were to a great extent independent of the ASIC design teams. They were able to rapidly implement the specification as a behavioural model and iron out the many details which were either ambiguous, incomplete, or incorrect in the paper specification. This modelling could be carried out by one person and required about 1/8 of the effort necessary to develop synthesizable RTL code.

In order to maximize code re-use and guarantee mutual conformance, the methodology required that test benches developed to test the behavioural model, would be entirely re-used by the ASIC teams and vice versa. The behavioural models were required to pass as close to 100% of the ASIC test plans as possible to enable them to stand in for the devices in system simulations while the RTL was under development. Non-conformance was only allowed on tests that required the exact internal structure such as scan and built-in test modes.

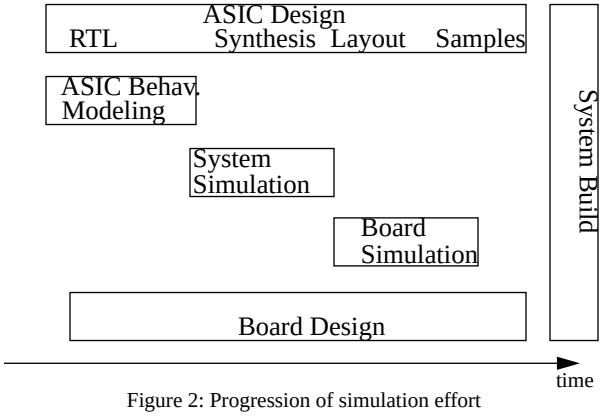

Figure 2: Progression of simulation effort

Once the behavioural models were complete, the focus shifted to System Simulation. This term is used widely in the industry to refer to many types of modelling work. In the context of this paper, system simulation refers to analyses which spanned multiple ASICs, crossing circuit board boundaries freely and stubbing out components whose functionality was not essential to the behaviour under test. Successful completion of these system tests gated the release of each ASIC to its layout phase. Note that since the ASIC schedules were not all exactly aligned, the behavioural models served an essential role in enabling these system simulations to take place in the absence of a complete RTL chip set.

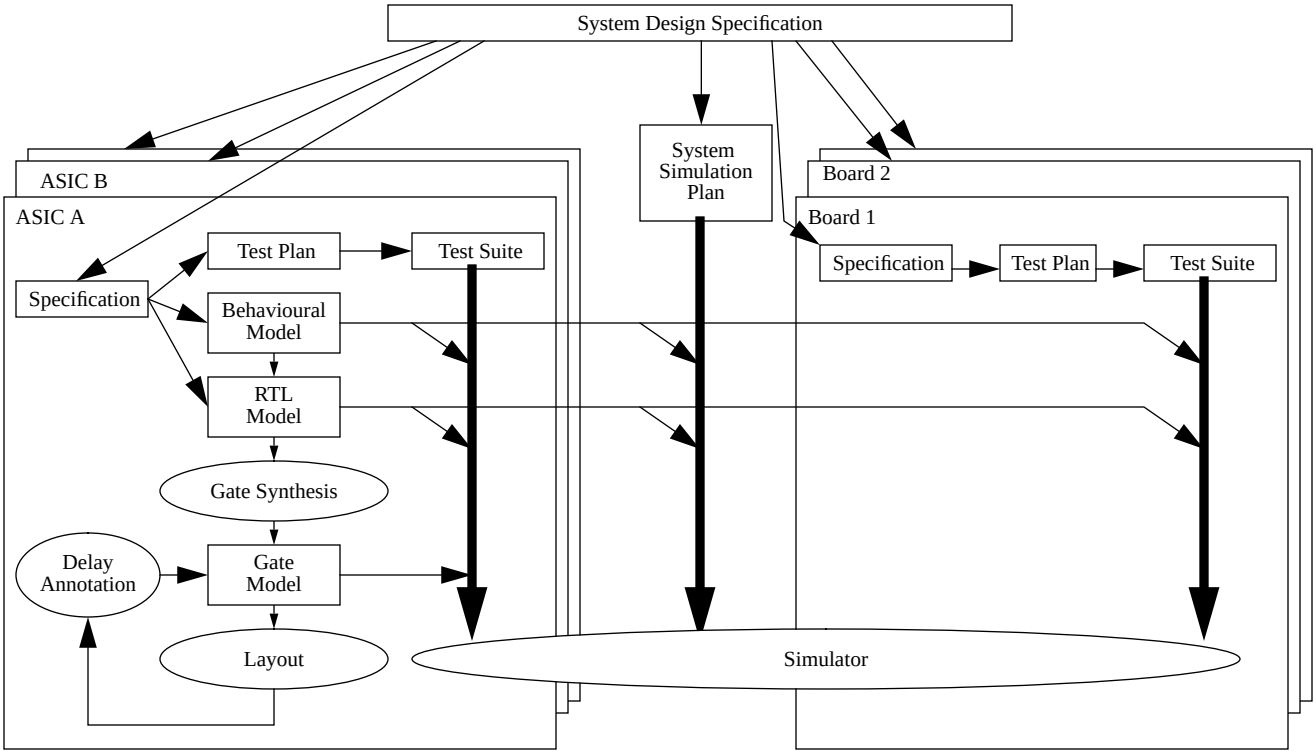

Figure 3: Verification Process

After system simulation work was complete, circuit pack simulation began. The emphasis shifted to debugging of the board netlist data and programmable logic components. Behavioural models for the ASICs were used primarily since simulation efficiency was essential to making progress in this effort. However, if CPU resources are available, RTL models could be used.

IV. Simulation Methodology

A *ASIC Verification*

From the specification of each ASIC, a conformance test plan was written to ensure that the implemented ASIC, regardless of its internal details, conformed to its specification. Each test case verified one or a few features of the ASIC from the top-level pins only, treating the device as a functional black box. A device or a model was declared to conform if, and only if, it passed all of the tests.

Testing conformance from the top-level pins only, and not at the block-level, enabled the use of a single set of test cases to test any model of the device which offered a pin-true top-level interface, regardless of its implementation. For the LIM ASICs, each test case was implemented as a single regressionable test bench whose output was a go/no-go flag. Scripts were used to run the entire regression test suite on either the behavioural, RTL, or gate level models and collect simulation results.

For the SM ASICs, the system-level tests were developed first and approximately 70% of the ASIC test cases were derived from them. ASIC-specific test benches were developed only where coverage was not easily accomplished at the system level. The exact timing-true nature of the SM models enabled vectors to be captured and exported at an ASIC boundary. These were re-run in isolated ASIC test benches for improved simulation efficiency. This method also enabled system simulation work on the SM to be carried out in VHDL while RTL was written in Verilog. RTL code was later instantiated in system tests using co-simulation to bridge the language gap.

B *Test Bench Tooling*

In order to faithfully reproduce the environment in which each ASIC or system would find itself, generators, monitors and bus-functional models were required to emulate the components to which the device under test would eventually be physically connected.

Up front specification of protocols for data and control signals at all ASIC boundaries (e.g. microprocessor interface, SONET processing, ATM cell encapsulation) enabled re-use of hardware functional blocks and test bench code. A common set of test bench utilities which could be configured to a particular chip interface was used throughout the project.

These utilities removed the test bench writers from the low-level details of the protocols by providing a procedural interface to all of their operations. For example, the processor utility provided *read*, *write*, and *interrupt handling* procedures. In the ATM cell generator and monitor, only the values of cell fields needed to be manipulated as the utilities themselves took care of formatting/extracting the cell into/from the low-level bit patterns and sequences required at each ASIC interface. They also allowed the test benches to

be tolerant to changes in the low-level protocols as only the utilities needed to be changed in the way they format the data while keeping the procedural interface unchanged. Additional controls were provided to introduce (or notify of) errors, such as parity, framing or cell length violations. Figure 4 shows the structure of a test bench using these utilities. This approach enabled approximately 10 000 lines of com-

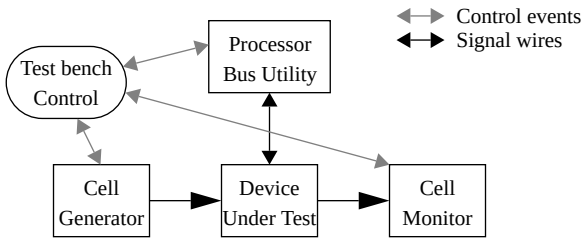

Figure 4: Test Bench Structure

mon code to be re-used in test benches throughout the simulation hierarchy. Many of the utilities are also being re-used in other projects within BNR.

C Test Bench Coding Techniques

The ASIC and system-level test benches for the LIM had to be self-adaptive to the cell latency through the models as they were different for the behavioural, RTL, and the gate levels (in transient cases).

Making a test bench self-adaptive requires a radical departure from conventional clock-by-clock test-vector and expected-response testing techniques. In the latter, the precise time when expected data is captured from the outputs becomes rigidly coded into test benches. With our method, expected responses are timed to a window relative to the initiating stimulus that is only constrained as tightly as the chip specification demands. Implicit timing derived from the ASIC implementation and not necessarily required for system conformance is not allowed to creep into the test bench code.

For example, a specification may not state an exact response time of the system to a protection switching event. This time will depend on many details of the system state as well as the precise implementation of each ASIC. An adaptive test bench simply ensures that the correct sequence of events takes place within some bounding (worst case) times that may be specified for the system.

Further simplification of test benches was accomplished by encoding expected response and destination information as data fill in regions of the cell not processed by the devices in a particular test. For example the CLP bit in the ATM header was at times used to flag a cell with a parity error inserted. Sequence numbers and destination were coded into payloads. Cell monitors could then easily ensure no cells were lost or scrambled by the system.

SM test benches did not need to be self-adapting since all models had accurate timing. However this required that test benches had to be modified whenever the latency changed as the design progressed. This was accomplished using a few generic parameters for tuning time intervals in the system test suites.

D System Simulation

The goal of the system simulation effort was to supplement the ASIC verification with additional functional coverage from a system feature viewpoint. This served to verify that the low level components implemented the high level features of the system as a whole. It required that simulations be carried out with the same RTL code from which each ASIC was to be synthesized. It was not practical to analyse performance characteristics such as cell throughput or loss ratios in this environment since simulation run times required to obtain meaningful data would be excessive. These parameters were simulated during initial system design using more abstract models in a dedicated simulator.

A system test plan was developed that focused on features and functions spanning multiple ASICs and the low level chip to chip interactions they relied upon to work together. To avoid unnecessary work, we attempted to minimize overlap of simulation feature coverage at the ASIC, board, and system level. The features targeted for system simulation on this product were:

- high level flow control
- fault detection and recovery mechanisms
- H/W S/W control sequences;
- full data path tests

These behaviours were not observable in isolated ASIC tests.

System simulation was focused on ASIC to ASIC behaviour, and traversed many physical partitions that existed in the product. The utilities developed to perform the system test benches modelled only the interfaces to commercial components on the circuit pack. This made the system simulations immune to many of the component changes at the early stages of the board design, allowing the two to proceed concurrently.

The link level flow control system test case is illustrated in Figure 5. In this test, multiple queues managed by the Q chip must be regulated to match the link bandwidth by a message passing scheme from each link. These messages are processed through intermediate chips such as the MUX chip. Many components, such as the microprocessor and control bus hierarchy are abstracted away as well as physical partitions such as backplane connectors. This simulation contained 4 ASICs (8 instances total).

E Board Simulation

The board simulations concentrate on the additional design data available with the circuit pack netlist. This consisted of board interconnect, back plane interconnect, FPGA and discrete component functionality. In contrast to system simulation, all of the components were now instantiated with either commercial and hand-crafted behavioural models or hardware models. The most complex board on the LIM for

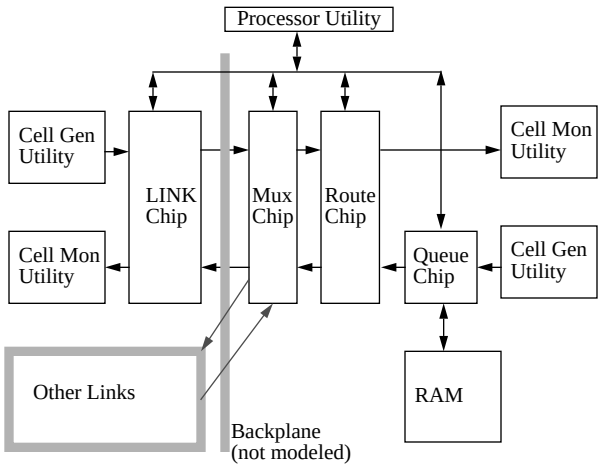

Figure 5: System Simulation Test Bench

example contained 4 ASICs (8 instances), 2 different Hardware models, and 10 commercial components (33 instances). Simulation characteristics for this board are given in Section V.

The methodology used is illustrated in Figure 6. Each circuit pack was tested as if it were installed in the backplane. Other backplane slots were populated with bus functional models of the other boards. These models were built largely from the test bench utilities. The bus functional models generated and monitored data to/from the card under test.

Typical board simulation test cases consisted of card reset and initialization and simple data path operations. Much attention was paid to the control circuitry from the on board processor to each ASIC. This path, which contained most of the discrete (non-ASIC) functionality was not visible in system simulations. However, the embedded S/W initialization sequences were developed previously in the system simulation environment and enabled embedded S/W development and board simulation efforts to be accelerated. Typical of tests performed at this stage were the verification of unique and correctly ordered buses as well as correctly connected and addressed interrupt lines.

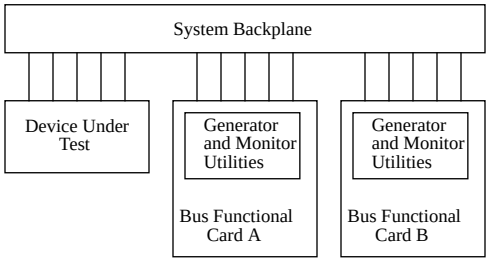

Figure 6: Circuit Pack Simulation Test Bench

V. Process and Simulation Metrics

During the course of the project, any time an issue was found with any of the specified design units (classified as ASIC, HW interface specification, circuit board or software) it was recorded in a data base. Issues were classified according to their severity by 3 levels. A Level 1 issue had to be addressed immediately. Its early detection was likely to have averted a slip in the schedule of a major deliverable or a significant specification non-compliance. Correction of the problem at the time it was detected was typically straightforward and caused minimal impact on schedule due to the early detection. At the other extreme, level 3 issues were ones that were unlikely to have found their way into the product or if so would have had minimal impact. Most commonly they were errors in the written specifications which were not in agreement with RTL code or other forms of design capture that were in progress at the time.

A summary of the issues found according to their severity and the simulation effort which uncovered them is shown in Table II.

Table II: Design Issues Found By Simulation

	Level 1	Level 2	Level 3	Total
ASIC Behavioural Modelling	18	51	101	170
System Simulation	8	2	20	30
Board simulation	6	5	5	16
Totals	32	58	126	216

Since simulation progressed from ASIC to system to board level, the majority of issues were found at the early stages of the concurrent efforts. The simulation burden (size and speed) followed an increasing progression as shown in Table III.

Since most of the H/W functional complexity of the product is embodied in ASICs and they were simulated at such and early stage, most of the issues were logged against these components as illustrated in Table IV. Most of these would likely have been caught later on during RTL coding had no behavioural models been developed. However, there were a handful of critical issues uncovered which may not have been found until very late in the design cycle or not at all. This was common in even the most heavily reviewed specs and indicates that the enormous detail required to specify a large ASIC can not be fully captured and verified by a natural language document and a manual review process.

Table III: Typical Test Bench Simulation Times And Process Sizes (seconds/MB)

Simulation Level	Gates.	RTL	Behaviour
ASIC test bench	1900/85	1350/23	400/6.5
System test bench		9872/42	1054/24
Board test bench		44574/102	16500/76

Table IV: Issues found by simulation

	I/F Specs	ASICs	Board	S/W	Totals
Behavioural modelling	13	157			170
System simulation	5	10	5	10	30
Board simulation		1	15		16
Totals	18	168	20	10	216

Board simulations primarily found issues on the functionality captured at the board level (connectivity and FPGA function) as they were targeted to do. System tests also uncovered problems with the early specifications for bootstrap sequences in the low level software. Common interface specifications which spanned multiple ASICs were also the target of a number of issues that were uncovered.

VI. CONCLUSIONS

We have described a methodology for functional hardware verification that spanned the ASIC, board and system level enabling simulation of the design concurrent with ASIC and board development. The simulation strategy relies on rapid development of behavioural models of ASICs to enable work to proceed in parallel and to achieve the necessary simulation efficiency. The effort provided the early visibility of over 200 issues in the system of which 32 were critical to the successful conformance and timely completion of the project.

REFERENCES

[1] A. Mendelsohn, "Now you're talking: verification strategy shapes telephony," *Computer Design*, vol. 33, no. 10, pp. 103-110, September, 1994.

[2] M. Hsu, "What you should expect the system simulation engineer you're going to hire to do", *Proc. VHDL International Users Forum*, pp.11.1 - 11.6, Nov. 1994

[3] E. Parrella and M. Tota, "Behavioral testbenches for telecommunications chipset development", *Proc. VHDL International Users Forum*, pp.12.27-12.33, Nov. 1994