Designing High Performance CMOS Microprocessors Using Full Custom Techniques

William J. Grundmann, Dan Dobberpuhl*, Randy L. Allmon, Nicholas L. Rethman

Digital Semiconductor, Digital Equipment Corporation, Hudson, MA.
*Digital Semiconductor, Digital Equipment Corporation, Palo Alto, CA.

Abstract **-- In this paper, we describe a full custom CMOS design methodology and supporting CAD technologies used to develop ALPHA and StrongARM microprocessors at Digital Semiconductor. The paper is subdivided into four parts, starting with a description of the design methodology and general CAD flows. Additional sections focus on two particular areas of interest: high performance low-power and full custom design benefits and verification issues.**

1 Introduction

Microprocessor designers continue using higher speed clocks combined with advanced microarchitectures to create the highest performance [2,3,4] and highest performance per Watt [1] CPUs. Microprocessor chips like these are difficult to design and verify, while meeting all performance and functional goals on first pass silicon.

High clock speed chips are considerably more difficult to design because they use complex circuit styles. These circuit implementations require extensive electrical verification in addition to conventional logical verification. This emphasis on electrical issues causes problems when trying to force-fit traditional logic oriented design methods and CAD tools into the design flow.

Using conventionally sanctioned design methodologies and CAD techniques that were primarily developed for much simpler and slower design architectures (i.e. the more common ASIC designs) handicaps a microprocessor designer. Many microprocessor designers are using or reverting to full custom design methodologies to stay competitive. Since conventional EDA vendors do not support this type of niche-market design style, these designers have to have significant local CAD resources to satisfy their needs.

At Digital Semiconductor, we use a stylized full custom design methodology supported by both design and CAD resources, which has evolved over many generations of microprocessor designs. We concluded many years ago that full custom design methods allow the designer freedom to solve their electrical problem, and to achieve their circuit's performance goal.

Successfully applying a full custom methodology can be very difficult. This approach can also be particularly difficult for many outside of the design team to understand.

Design Automation Conference ®

0-89791-847-9/97/0006/$3.50 DAC 97 - 06/97 Anaheim, CA, USA

2 Full custom design methodology

Everyone seems to have a different definition for the meaning of full custom design methodology. We provide our own definition. Characteristics of our full custom design methodology are simply:

- Transistors are the building elements. Other building elements (cells) are nice but not required.
- Every transistor in the design can be (and often is) individually sized, regardless of its functional context.

Transistors are combined together to form a broad range of logic families with full and reduced output voltage swings. The logic families include dynamic, single or dual-rail circuits, differential cascode voltage swing logic (DCVSL), pass transistor logic, and of course, complementary logic gates.

Designers have the freedom to use these transistors in any creative fashion, anywhere in the design, to achieve their performance goals while meeting functionality requirements and their schedules. This means that functional units and state-elements can be invented "on-the-fly", and application of both dynamic and static circuits are possible.

A successful full custom design methodology depends upon many essential prerequisites and requirements, each having incredible interdependencies. Essential prerequisites are experienced circuit/logic designers and team accepted design standards. Important requirements include customized CAD tools and support oriented to full custom design and verification.

Digital Semiconductor's design methodology follows a "Correct by verification" (CBV) instead of the more popular "Correct by construction" (CBC) methods. CBV

better addresses the key electrical issues involved with high-performance designs, while CBC may still be adequate for non-critical designs.

In the following sub-sections, we talk about four of our design methodology aspects: use of hierarchy across different representations of the design, creation of circuits and layout, verification of intent, and design flow.

2.1 Digital Semiconductor's use of design hierarchy

Our use of design hierarchy is different from the industry norm. Champions of the status quo advocate strict use of matching design hierarchy across all chip design steps as the only possible way to be successful.

While use of hierarchy helps the logical understanding of the design, exhaustive use of logic hierarchy often hinders circuit understanding. Design hierarchy is used when it makes appropriate electrical sense. Electrical hierarchy helps control the physical attributes of the chips' layout, which is ultimately more difficult than the logical aspects.

Our hierarchy may be significantly different between different views of the design (RTL, schematic, and layout). The designer is free to move logic/circuit functions physically to achieve their performance goals without having to maintain strict correspondence to the RTL description. This causes irregular overlapping of schematic and RTL boundaries as shown in Figure 1.

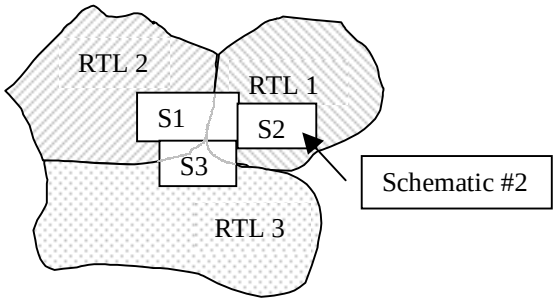

Figure 1: RTL vs. Schematic hierarchy

The implementation of any logic function can also be different between design views, particularly between RTL and schematic as discussed in section 2.2.

2.2 Creation of logic, circuits, and layout

Most transistors on our microprocessors are constructed in arrayed or datapath structures. Traditional logic and layout synthesis techniques have not been very successful in creating these high performance circuits. Therefore, most logic in the design is still manually created. Automatic logic synthesis, when used, is oriented towards creation of raw unsized gates, allowing designer manipulation to the final form. Transistors are sized either by the designer or by using automatic path sizing techniques.

Implementation in different views of the design may significantly deviate from other views as long as overall they still achieve the same logical intended behavior (at some design boundary). For example, a functional description in the RTL may be modeled as a single output, which changes value at most once per cycle. The same function in transistor circuits may be implemented as a dual-rail, precharge-discharge circuit, which has a complementary value on the outputs in only one phase.

Schematic cell libraries are not required. However, we have found that circuit topology templates are very useful in full custom. For instance, a NAND gate function can have a NAND gate appearance, but have individual control of device sizes per instance. Arbitrary complex complementary gates can be created and sized "on-the-fly". Additional use of local schematic hierarchy (macro-box) is extremely useful to define templates that are used frequently on a particular schematic, but not needed anywhere else.

While automatic logic synthesis has not historically been a major factor in our designs, CAD layout synthesis and assistance tools have had a greater impact in our layout creation. The emphasis of these layout generation tools is to assist in the creation of macrocells, at the level of transistor place and route.

2.3 Verification

Digital Semiconductor's design methodology is supported by many methods to assist in logical, electrical, physical, and reliability verification. A large challenge caused by our methodology is the automatic recognition of groups of full custom transistors in their logical and electrical meanings. The logical behavior or intent of a collection of transistors has no inherent pre-defined meaning as normally provided by traditional cell library approaches. Subsequently, all logic and timing constraints along with electrical requirements have to be automatically and conservatively deduced from the topology and context of the actual transistors.

For many verification questions, we do not have an absolute answer. Instead, we use CAD tools to filter the amount of design the designer has to inspect. These CAD tools use the circuit recognition information along with other information (e.g., capacitance and timing) to provide filtering of circuits that do not have a problem, and reporting those circuits that might have a problem. This allows the designer to work with the CAD tool to identify and isolate real problems in the design.

Overall, this methodology maximizes the freedom for the creative designer. It also requires a significant increase in trusting a designer's ability to make a working design and delivery of a supporting verification methodology to conservatively analyze correctness of the design.

This design methodology is constantly changing. We have to adapt to different circuit implementation ideas, new electrical concerns, and new manufacturing processes, while still addressing and improving designer productivity.

Additional topics on logic, circuit, and timing verification are covered in section 4.

2.4 ALPHA microprocessor design flow

The design flow used for ALPHA CPU designs is similar in appearance to many other design flows (Figure 2). A significant difference to other design flows is the amount of automatic synthesis of schematic and layout. Since there is a reduced amount of automatic synthesis, there has been much more emphasis on the verification of all implementation representations.

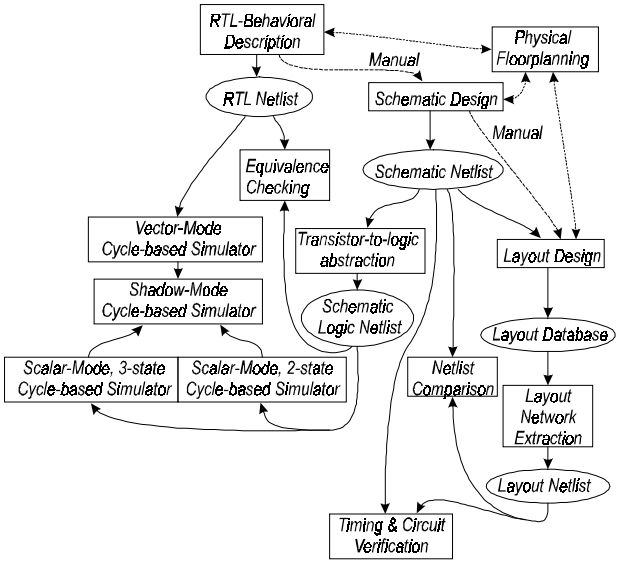

Figure 2: ALPHA design flow

Although this appears as a top-to-bottom flow, there are actually many bottom-to-top interactions. For instance, there are many feasibility studies on different circuit implementations during the development of the RTL. These studies analyze timing, layout area, power, and electrical concerns. Physical floorplanning also occurs during all design phases and helps control eventual circuit performance and area results.

3 Low power design

Power dissipation is a concern for any chip, but it is especially important for applications intended for portable, "Anywhere-Anytime" computing [1]. In 1992, the first ALPHA chip delivered the raw performance of a Cray-1 in a single device dissipating about 25W [2]. The next generation of ALPHA chips delivered more than four times that performance level at about the same power [3]. The latest ALPHA CPU delivers more than 8X the performance level at about twice the power [4]. The design and fabrication technology which has made this possible, when applied within the constraints of the portable computing market, can deliver Cray-1 class performance to battery-powered and low cost tethered applications. The StrongARM 110™ is such a device, designed by full custom designers using techniques developed for ALPHA CPU's.

To achieve the substantial power reduction from ALPHA to StrongARM, several well known methods were applied: reduced VDD, conditional clocking, efficient state elements, and micro-architecture carefully balanced between clock rates (pipelining) and clock efficiency (CPI). It is interesting to quantitatively compare the power dissipation of StrongARM to an ALPHA CPU to see how this is achieved. As shown in Table 1, the ALPHA 21064 is compared against StrongARM for first order differences.

Starting with ALPHA 21064: 200MHz @ 3.45v, Power = 26W	
VDD reduction:	power reduction = 5.3x -> 4.9W
Reduce functions:	power reduction = 3x -> 1.6W
Scale process:	power reduction = 2x -> 0.8W
Clock load:	power reduction = 1.3x -> 0.6W
Clock rate:	power reduction = 1.25x-> 0.5W

Table 1: ALPHA -> StrongARM Power Dissipation

Starting with a 200MHz 21064 in 0.75• technology, factoring in VDD, functionality differences, process scaling, clock loading and frequency, we end up with a power dissipation close to the realized value of 450mW.

A process for low-power using a low-supply voltage and low-threshold device is essential to the design of a microprocessor that will run at 160MHz while burning only 500mW. However, the low device thresholds, which allow the reduction of VDD, also result in significant device leakage. While this leakage is not large enough to cause a problem for normal operation, it does pose problems for standby current. To reduce this leakage, devices in the cache arrays, the pad drivers, and certain other areas were lengthened by 0.045µm or 0.09µm as part of the design process. This brought the leakage power to below the 20mW specification in the fastest process corner.

4 Full custom benefits and headaches for verification

Designing full custom must result in manufacturing technology and designer creativity limiting circuit performance, not the CAD methods. This freedom also means the extreme need for project control (adherence to design methodology) and verification CAD-tool support.

There are really only three silicon product goals: cost, performance, and functionality or capability. Everything a

designer does relates to one or more of these fundamental goals. "Cost" is managed via the architecture definition (function, packaging, etc.), time-to-product (schedules), and quality (Defects, Lifetime, etc.) Our verification methodology breaks most of these product goals into three categories: Logic verification, circuit verification, and timing verification.

4.1 Logic verification

We perform logic verification at four levels: Behavioral/RTL simulation, standalone schematic simulation, shadowed schematics under RTL simulation, and RTL to schematic equivalence checking. Since most of our logic verification is simulation based, the speed of simulation is very important. Phase accurate simulation of Behavioral/RTL can be performed, achieving >200 cycles per second per simulation CPU. To execute our typical logic design verification goals of two billion aggregated simulated cycles per day requires dedication of about 100 CPUs.

Our high-level logical model of a full-custom design includes both behavioral and RTL constructs. The level of detail for any part of the description depends upon many issues of uncertainly in implementation. For example, a common problem is to determine if a particular circuit's implementation functional works at all. The Behavioral/RTL description is the first representation of the design, and is continuously updated to better reflect differences in actual circuit implementation.

Standard hardware description languages have proven to be inadequate for us when describing highly variable (function changing daily) parts of the design. In addition, these standard languages tend to require more hierarchical levels than desired. Some of our functional units are just difficult to code in standard languages and result in highly inefficient run-times, e.g. a 2000 "port" CAM structure. We have developed a hardware language driven by our style of designing microprocessors, with programming constructs that make sense for the design itself, and which compiles into very efficient code.

Our full custom designs have a significant difference between the Behavioral/RTL model and the circuit description. This is caused by two factors: circuit designers adding creative difference by liberally interpreting the Behavioral/RTL model, and circuit detail which is created to achieve other goals but maintaining overall logical intent.

Since the circuit implementation is loosely equivalent to the high-level model, two methods are used to check functional correctness. The first is logic simulation, executed in a stand-alone type of simulation or more popular at Digital Semiconductor is the "shadow-mode" simulation. This latter simulator is a mixed mode simulation of full design Behavioral/RTL with a part of the circuit logic shadowing (not replacing) the corresponding RTL description. Simulation requires stimulus patterns, which are either manually generated or pseudo-random sequences.

The second method for functional correctness of circuits is logical equivalence checking. This does not require input stimulus, however a common difficulty is the amount of logical difference that an equivalence-checking tool can accommodate. This can be complicated since the designer has the freedom to create a circuit that behaves the same with different state declarations and state transitions. For instance, a counter coded in the Behavioral/RTL model with an output every five events may be implemented in the circuit as a shift register with a cyclic value of five. In this example, both achieve the same behavior, but are significantly different in internal implementations.

Digital Semiconductor's logic verification strategies include both simulation and equivalence checking, thoroughly providing coverage of logic intent.

4.2 Circuit verification

Circuit verification covers any circuit implementation issues not directly related to logic or timing behavior [5]. Most of these verification steps address circuit functional hazards, adherence to chip specific design methodologies, and circuit reliability. The many logic families and heavy use of dynamic logic has given the designer the ability to meet their circuit's performance and functionality goals. Each of these logic families has electrical considerations that are taken into account during the design phase, however a post-layout verification step ensures that all goals were met.

The circuit verification at Digital Semiconductor depends upon heavy use of CAD verification for those issues which rules can be clearly specified. Additional CAD tools perform probability "filtering" on any remaining complex, hard to clearly specify design rules. This approach eliminates those situations that have a high degree of confidence of being correct while reporting the situations that may have violations and require closer inspection by the designer.

Figure 3 illustrates the type of verification checks that are performed looking for circuit noise in dynamic circuits. The primary sources of noise are interconnect capacitance coupling that could corrupt the dynamic node, charge sharing between the dynamic output node and the internal transistor stack nodes, and power supply voltage differences between the driver and receiver circuits. Other sources of noise include Alpha particle and noise induced minority carrier charge collection from the substrate and wells, and sub-threshold leakage through the N-device network. In-house CAD tools are used to analyze these concerns. The tools use extracted interconnect parasitic capacitance and resistance data, signal timing information,

transistor capacitance, drive strength and fanout to identify potential circuit failures.

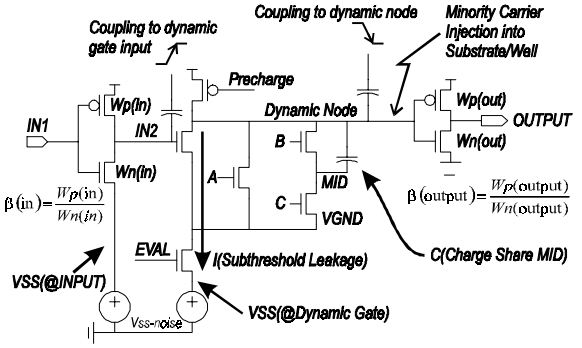

Figure 3: Noise sources in dynamic structures

The automated CAD circuit verification checks performed at Digital Semiconductor include:

- Transistor configuration analysis
- Beta ratio and device size checks of all complementary and ratioed structures.
- Clock distribution RC analysis.
  - Node-by-node clock RC analysis
  - Correlated minimum/maximum RC analysis
- Edge rate and delay analysis for clocks and signals
- Latch checks
- Coupling analysis of static and dynamic nodes
- Dynamic charge share analysis
- Dynamic node leakage checks
- State-element writability and noise margin analysis
- Electromigration, statistical and absolute failures
- Antenna checks
- Hot Carrier and Time Dependant Dielectric Breakdown checks

4.3 Timing verification

Timing verification is used to identify all critical and race paths. Critical paths (slow paths) will limit the clock frequency of the chip while race paths (fast paths) will prevent the chip from working at any frequency. Figure 4 illustrates our definition of both the critical path and races. Traditionally timing verification has focused more on critical paths than on race paths. However, when exotic clocking methodologies are used, the importance of verifying race paths greatly increases.

The biggest worry a designer has about timing verification is that a timing induced functional violation may go undetected. If a violation is missed by the timing verifier and the violation makes it to silicon then a costly debug along with a schedule slip will probably result. The designers also want to minimize the number of false violations they have to examine. As the number of false violations goes up, the productivity of the designer goes down and the greater the risk that real violations will be lost in a sea of output.

Two main areas can cause violations to be missed. The first occurs when the timing verifier miss-recognizes some circuit constraints (setup, hold, and glitch). The second occurs when min/max delay times are not calculated accurately.

$$\max(t_1) + t_{setup} - \min(t_2) \le T_{cycle}\left[1\ Cycle\right]$$

$$\max(t_1) + t_{setup} - \min(t_2) \le T_{phase}\left[\frac{1}{2}\ Cycle\right]$$

Critical Path Definition

$$\min(t_1) \ge k\left(\max(t_2) + t_{hold}\right), \text{ where } k \ge 1$$

Race Definition

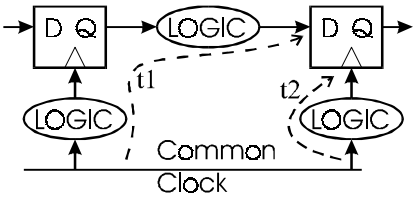

Figure 4: Clocking and Timing Methodology

The reliability of recognizing circuit constraints is a big problem due to the freedom the designers have in creating state-elements "on-the-fly". The automatic recognition of state-elements, clocking nodes, glitch sensitive nodes, and data nodes is essential. In addition, algorithms are needed, which when given this information, will automatically identify the constraint and calculate the correct constraint time (setup time and hold time) for any full custom circuit. The constraint generation algorithms must be accurate but error on the side of being pessimistic in order to insure no violations are missed.

There are three main areas concerning the accuracy of the delay calculation:

- Accuracy of minimum and maximum capacitance calculation (fixed, coupling, and transistor input)
- Accuracy of RC interconnect models
- Accuracy of computing transition offsets and transition edge

A traditional extraction and calculation of max capacitance partially addresses the goal of identifying critical paths. However, max-capacitance alone does not address the issue of correctly identifying circuit races that prevent the part from working under any operating conditions. Internodal capacitance values (coupling capacitance) have significant variation from both manufacturing tolerances and miller coupling capacitance multiplicative effects. Bounding the min/max coupling along with manufacturing tolerances is essential in accurately computing nodal capacitance.

Transistor gate input capacitance can also have a wide range of values, depending upon its logical context. That context includes the state and transitions of other inputs to the logic function, topological position relative to power or ground, and the state and transition of associated source and drain transistor connections.

The traditional "gate" modeled with a single output "port" no longer works in high-performance designs, especially in the presence of significant RC interconnect. For instance, a large inverter is commonly implemented with many smaller transistor fingers distributed across a large area along the output node. This results in the output of inverter tied into multiple positions along the RC grid as in Figure 5. This is additionally complicated by the fact that the inputs of the individual inverter transistors are also themselves, outputs of another RC grid. The inverter's "turn-on" characteristics are highly dependent upon the inverter's RC input and output grid characteristics.

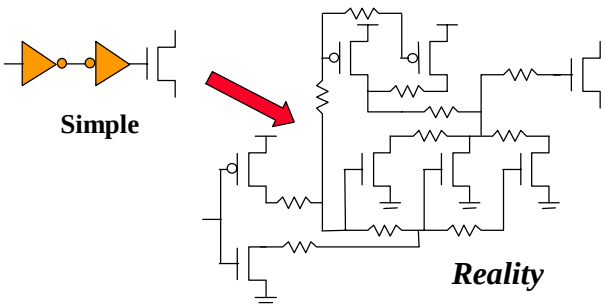

Figure 5: Real gates have multiple inputs/outputs

The timing model itself must be accurate and, if necessary, error on the side of being pessimistic. Typically, the designer uses SPICE to obtain the delay times and edge rates. However, using SPICE on large structures is not feasible due to the size and turnaround time of the timing simulation. Therefore timing models for individual transistors and "clumps" of transistors are derived that sacrifice accuracy for simulation efficiency. Also, SPICE will only give you the delay times for the particular input condition which the user supplied and not the worst case min and max times. Therefore timing models must also be smart enough to setup the delay calculation for the worst case min (fastest delay time) and max (slowest delay time).

Three main areas can cause false violations. Two of these areas are the constraint identification and the delay calculation mentioned above. If either of these two areas is too pessimistic then false violations will be created. Static timing verification always has two conflicting goals: enough pessimism to insure identification of all violations, while not so much pessimism to cause false violations. A third culprit of false violations is a logically or architecturally false path. Automatic elimination of these false paths is difficult due to insufficient information of designer's intent.

By allowing the designer to have the freedom to use any arbitrary configuration of transistors, we enabled them to develop high performance microprocessors. However, allowing this freedom poses significant challenges to any circuit timing verification tools.

5 Conclusion

We have provided a short description of the design and CAD methodology used to develop the high performance ALPHA and StrongARM microprocessors at Digital Semiconductor. This methodology was shown similar yet in many ways significantly different to traditional methods. These differences were focused on the correct-by-verification strategy as applied to high-speed clocked microprocessor designs.

Acknowledgement

The authors thank Bill Bowhill for his help with this paper.

References

[1] J. Montanaro, R. Witek, K. Anne, A. Black, E. Cooper, D. Dobberpuhl, P. Donahue, J. Eno, G. Hoeppner, D. Kruckemyer, T. Lee, P. Lin, L. Madden, D. Murray, M. Pearce, S. Santhanam, K. Snyder, R. Stephany, S. Thierauf, "A 160MHz 32b 0.5W CMOS RISC Microprocessor," ISSCC Digest of Technical Papers, pp. 214-215, Feb., 1996.

[2] D. Dobberpuhl, et. al., "A 200MHz 64b Dual-Issue CMOS Microprocessor," IEEE Journal of Solid State Circuits, vol. 27, no. 11, Nov., 1992.

[3] P. Gronowski, et. al., "A 433Mhz 64b Quad-Issue CMOS RISC Microprocessor," ISSCC Digest of Technical Papers, pp. 222-223, Feb., 1996.

[4] B. Gieseke, et. al., "A 600mhz Superscalar RISC Microprocessor With Out-of-Order Execution," ISSCC Digest of Technical Papers, pp. 176-177, Feb., 1997.

[5] P. Gronowski, et. al., "A 433 MHz 64b Quad-Issue RISC Microprocessor", IEEE Journal of Solid State Circuits, vol. 31, no 11, page 1687-1696, Nov., 1996.