

GC 2007-154: INTEGRATING ASYNCHRONOUS DIGITAL DESIGN AND TESTING INTO THE UNDERGRADUATE COMPUTER ENGINEERING CURRICULUM

Scott Smith, University of Arkansas

Scott C. Smith received BS degrees in Electrical Engineering and Computer Engineering from the University of Missouri - Columbia in May of 1996, an MS in Electrical Engineering from the University of Missouri - Columbia in May of 1998, and a PhD in Computer Engineering from the University of Central Florida, Orlando in May of 2001. He is an Associate Professor in the Department of Electrical Engineering at the University of Arkansas. He has authored 11 journal publications, 24 conference papers, 3 US/international patents, and 2 additional international patents, all of which can be viewed from his website: <http://comp.uark.edu/~smithsco/>. His research interests include computer architecture, asynchronous logic design, CAD tool development, embedded system design, VLSI, trustable hardware, and self-reconfigurable logic. Dr. Smith is a member of ASEE, Sigma Xi, Eta Kappa Nu, Tau Beta Pi, and a Senior Member of IEEE.

Waleed Al-Assadi, University of Missouri

Waleed K. Al-Assadi received a Ph.D. in Electrical Engineering with an emphasis in Computer Engineering from Colorado State University in 1996. He worked for AMD in Austin, TX and IBM in RTP, NC until summer 2003 as a senior engineer responsible for developing Design for Test Methodologies for Microprocessors. He has been with the Department of Electrical & Computer Engineering at the University of Missouri-Rolla as an Assistant Professor since August 2003. His research interests include VLSI systems design, embedded systems, computer architecture, and digital systems design. Dr. Al-Assadi is a member of ASEE, Eta Kappa Nu, Tau Beta Pi, and a Senior Member of IEEE.

Integrating Asynchronous Digital Design and Testing into the Undergraduate Computer Engineering Curriculum

Abstract

As demand rises for circuits with higher performance, higher complexity, and decreased feature size, asynchronous (clockless) paradigms will become more widely used in the semiconductor industry, as evidenced by the International Technology Roadmap for Semiconductors' (ITRS) prediction of a likely shift from synchronous to asynchronous design styles in order to increase circuit robustness, decrease power, and alleviate many clock-related issues¹. ITRS predicts that asynchronous circuits will account for 19% of chip area within the next 5 years, and 30% of chip area within the next 10 years². To meet this growing industry need, students in Computer Engineering should be introduced to asynchronous circuit design to make them more marketable and more prepared for the challenges faced by the digital design community for years to come.

1. Introduction

The development of synchronous circuits currently dominates the semiconductor design industry. However, there are major limiting factors to the synchronous, clocked approach, including the increasing difficulty of clock distribution, increasing clock rates, decreasing feature size, increasing power consumption, timing closure effort, and difficulty with design reuse. Asynchronous circuits require less power, generate less noise, produce less electro-magnetic interference (EMI), and allow for easier reuse of components, compared to their synchronous counterparts, without compromising performance.

In most Computer Engineering curricula students are only taught the synchronous, clocked paradigm, and never even touch on asynchronous digital design. Those curricula that do mention asynchronous design do so only in passing; the students are not taught how to design asynchronous circuits. The widespread introduction of asynchronous digital design in the classroom is largely constrained by the lack of introductory educational materials. This paper presents one approach for integrating asynchronous circuit design into the undergraduate Computer Engineering curriculum, focusing on inclusion in two courses, one on Hardware Design Languages (HDLs), such as VHDL, and the other on VLSI.

The paper is organized into 5 sections. Section 2 presents an overview of asynchronous logic; Section 3 describes the asynchronous materials developed for use in undergraduate Computer Engineering courses; Section 4 depicts the original VHDL and VLSI course outlines and shows how these courses have been augmented to include the asynchronous materials; and Section 5 presents the outcomes of the first offerings of the VHDL and VLSI courses with the asynchronous materials included, and provides conclusions and directions for future work.

2. Overview of Asynchronous Logic

Asynchronous circuits can be grouped into two main categories: *bounded-delay* and *delay-insensitive* models. Bounded-delay models such as *micropipelines*³ assume that delays in both gates and wires are bounded. Delays are added based on worse-case scenarios to avoid hazard conditions. This leads to extensive timing analysis of worse-case behavior to ensure correct circuit operation. On the other hand, ***delay-insensitive (DI)*** circuits assume delays in both logic elements and interconnects to be unbounded, although they assume that wire forks within basic components, such as a full adder, are isochronic^{4,5}, meaning that the wire delays within a component are much less than the logic element delays within the component, which is a valid assumption even in future nanometer technologies. Wires connecting components do not need to adhere to the isochronic fork assumption. This implies the ability to operate in the presence of indefinite arrival times for the reception of inputs. Completion detection of the output signals allows for handshaking to control input wavefronts. DI design styles therefore require very little, if any, timing analysis to ensure correct operation (i.e., they are correct-by-construction), and they also yield average-case performance rather than the worse-case performance of bounded-delay and synchronous paradigms⁶.

2.1 Delay-Insensitive Circuits

Most DI methods combine *C-elements*⁷ with Boolean gates for circuit construction. A C-element behaves as follows: when all inputs assume the same value then the output assumes this value, otherwise the output does not change. Seitz's⁸, DIMS⁹, Anantharaman's¹⁰, Singh's¹¹, and David's¹² methods are examples of DI paradigms that only use C-elements to achieve delay-insensitivity. On the other hand, both Phased Logic¹³ and NULL Convention Logic (NCL)¹⁴ target a library of multiple gates with *hysteresis* state-holding functionality. Phased Logic converts a traditional synchronous gate-level circuit into a delay-insensitive circuit by replacing each conventional synchronous gate with its corresponding Phased Logic gate, and then augmenting the new network with additional signals¹³. NCL functions are realized using 27 fundamental gates implementing the set of all functions of four or fewer variables, each with *hysteresis* state-holding functionality¹⁵.

Seitz's method, Anantharaman's approach, and DIMS require the generation of all minterms to implement a function, where a minterm is defined as the logical AND, or product, containing all input signals in either complemented or non-complemented form. While Singh's and David's methods do not require full minterm generation, they rely solely on C-elements for speed-independence. NCL also does not require full minterm generation and furthermore includes 27 fundamental state-holding gates for circuit design, rather than only C-elements, thus yielding a greater potential for optimization than other delay-insensitive paradigms¹⁶. Phased Logic does not require full minterm generation and does not rely solely on C-elements for speed-independence; however, Phased Logic circuitry is derived directly from its equivalent synchronous design, and not created independently; thus it does not have the same potential for optimization as does NCL. Furthermore, the Phased Logic paradigm has been developed mainly for easing the timing constraints of synchronous designs, not for obtaining speed and power benefits¹³, whereas these are main concerns of other asynchronous paradigms.

Self-timed circuits can also be designed at the transistor level as demonstrated by Martin ¹⁷. However, automation of this method would be vastly different than that of the standard synchronous approach, since it optimizes designs at the transistor level instead of targeting a predefined set of gates, as do the previously mentioned DI methods. Overall, **NULL Convention Logic (NCL)** offers the best opportunity for integrating asynchronous digital design into the predominantly synchronous semiconductor design industry for the following reasons:

- 1) The framework for NCL systems consist of DI combinational logic sandwiched between DI registers, as shown in Figure 1. This framework is very similar to synchronous systems, such that the automated design of NCL circuits can follow the same fundamental steps as synchronous circuit design automation. This will enable the developed DI design flow to be more easily incorporated into the chip design industry, since the tools and design process will already be familiar to designers, such that the learning curve is relatively flat;
- 2) NCL systems are delay-insensitive, making the design process much easier to automate than other non-DI asynchronous paradigms, since minimal delay analysis is necessary to ensure correct circuit operation; and
- 3) NCL systems have power, noise, and EMI advantages compared to synchronous circuits, performance and design reuse advantages compared to synchronous and non-DI asynchronous paradigms, area and performance advantages compared to other gate-level DI paradigms, and have a number of advantages for designing complex systems, like Systems-on-Chip (SoCs), including substantially reduced crosstalk between analog and digital circuits, ease of integrating multi-rate circuits, and facilitation of component reuse and technology migration.

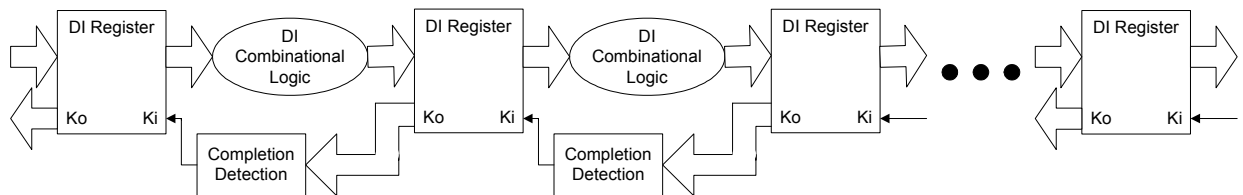


Figure 1. NCL system framework: input wavefronts are controlled by local handshaking signals and Completion Detection instead of by a global clock signal. Feedback requires at least three DI registers in the feedback loop to prevent deadlock ¹⁸.

2.2 NULL Convention Logic (NCL)

NCL is a delay-insensitive asynchronous paradigm, which means that NCL circuits will operate correctly regardless of when circuit inputs become available; therefore NCL circuits are said to be correct-by-construction (i.e., no timing analysis is necessary for correct operation). NCL circuits utilize dual-rail or quad-rail logic to achieve delay-insensitivity. A dual-rail signal, D , consists of two wires, D^0 and D^1 , which may assume any value from the set $\{\text{DATA0}, \text{DATA1}, \text{NULL}\}$. The DATA0 state ($D^0 = 1, D^1 = 0$) corresponds to a Boolean logic 0, the DATA1 state ($D^0 = 0, D^1 = 1$) corresponds to a Boolean logic 1, and the NULL state ($D^0 = 0, D^1 = 0$) corresponds to the empty set meaning that the value of D is not yet available. The two rails are mutually exclusive, such that both rails can never be asserted simultaneously; this state is defined as an illegal state. A quad-rail signal, Q , consists of four wires, $Q^0, Q^1, Q^2,$ and Q^3 , which may assume any value from the set $\{\text{DATA0}, \text{DATA1}, \text{DATA2}, \text{DATA3}, \text{NULL}\}$. The DATA0 state ($Q^0 = 1, Q^1 = 0, Q^2 = 0, Q^3 = 0$) corresponds to two Boolean logic signals, X and Y , where $X = 0$

and $Y = 0$. The DATA1 state ($Q^0 = 0, Q^1 = 1, Q^2 = 0, Q^3 = 0$) corresponds to $X = 0$ and $Y = 1$. The DATA2 state ($Q^0 = 0, Q^1 = 0, Q^2 = 1, Q^3 = 0$) corresponds to $X = 1$ and $Y = 0$. The DATA3 state ($Q^0 = 0, Q^1 = 0, Q^2 = 0, Q^3 = 1$) corresponds to $X = 1$ and $Y = 1$, and the NULL state ($Q^0 = 0, Q^1 = 0, Q^2 = 0, Q^3 = 0$) corresponds to the empty set meaning that the result is not yet available. The four rails of a quad-rail NCL signal are mutually exclusive, such that no two rails can ever be asserted simultaneously; these states are defined as illegal states. Both dual-rail and quad-rail signals are space optimal 1-hot delay-insensitive codes, requiring two wires per bit.

NCL circuits are comprised of 27 fundamental gates¹⁶, as shown in Table I, which constitute the set of all functions consisting of four or fewer variables. Since each rail of an NCL signal is considered a separate variable, a four variable function is not the same as a function of four literals, which would normally consist of eight variables. The primary type of threshold gate, shown in Figure 2, is the *THmn gate*, where $1 \leq m \leq n$. THmn gates have n inputs. At least m of the n inputs must be asserted before the output will become asserted. In a THmn gate, each of the n inputs is connected to the rounded portion of the gate; the output emanates from the pointed end of the gate; and the gate's threshold value, m , is written inside of the gate.

Table 1. 27 fundamental NCL gates.

NCL Gate	Boolean Function	Transistors (static)	Transistors (semi-static)
TH12	$A + B$	6	6
TH22	AB	12	8
TH13	$A + B + C$	8	8
TH23	$AB + AC + BC$	18	12
TH33	ABC	16	10
TH23w2	$A + BC$	14	10
TH33w2	$AB + AC$	14	10
TH14	$A + B + C + D$	10	10
TH24	$AB + AC + AD + BC + BD + CD$	26	16
TH34	$ABC + ABD + ACD + BCD$	24	16
TH44	$ABCD$	20	12
TH24w2	$A + BC + BD + CD$	20	14
TH34w2	$AB + AC + AD + BCD$	22	15
TH44w2	$ABC + ABD + ACD$	23	15
TH34w3	$A + BCD$	18	12
TH44w3	$AB + AC + AD$	16	12
TH24w22	$A + B + CD$	16	12
TH34w22	$AB + AC + AD + BC + BD$	22	14
TH44w22	$AB + ACD + BCD$	22	14
TH54w22	$ABC + ABD$	18	12
TH34w32	$A + BC + BD$	17	12
TH54w32	$AB + ACD$	20	12
TH44w322	$AB + AC + AD + BC$	20	14
TH54w322	$AB + AC + BCD$	21	14
THxor0	$AB + CD$	20	12
THand0	$AB + BC + AD$	19	13
TH24comp	$AC + BC + AD + BD$	18	12

Another type of threshold gate is referred to as a weighted threshold gate, denoted as THmnW_{w₁w₂...w_R}. Weighted threshold gates have an integer value, $m \geq w_R > 1$, applied to *inputR*. Here $1 \leq R < n$; where n is the number of inputs; m is the gate's threshold; and w_1, w_2, \dots, w_R , each > 1 , are the integer weights of *input1*, *input2*, ... *inputR*, respectively. For example, consider the TH34W2 gate, whose $n = 4$ inputs are labeled A, B, C , and D , shown in

Figure 3. The weight of input A , $W(A)$, is therefore 2. Since the gate's threshold, m , is 3, this implies that in order for the output to be asserted, either inputs B , C , and D must all be asserted, or input A must be asserted along with any other input, B , C , or D . NCL threshold gates are designed with *hysteresis* state-holding capability, such that all asserted inputs must be de-asserted before the output will be de-asserted. Hysteresis ensures a complete transition of inputs back to NULL before asserting the output associated with the next wavefront of input data. Therefore, a TH n n gate is equivalent to an n -input C-element and a TH1 n gate is equivalent to an n -input OR gate. NCL threshold gates may also include a *reset* input to initialize the output. Circuit diagrams designate resettable gates by either a d or an n appearing inside the gate, along with the gate's threshold. d denotes the gate as being reset to logic 1; n , to logic 0. These resettable gates are used in the design of DI registers¹⁸.

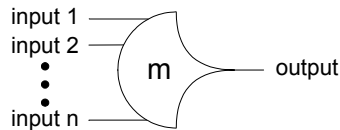


Figure 2. TH mn threshold gate.

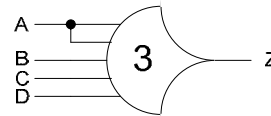


Figure 3. TH34w2 threshold gate:
 $Z = AB + AC + AD + BCD$

NCL systems contain at least two DI registers, one at both the input and at the output. Two adjacent register stages interact through their request and acknowledge signals, K_i and K_o , respectively, to prevent the current DATA wavefront from overwriting the previous DATA wavefront, by ensuring that the two DATA wavefronts are always separated by a NULL wavefront. The acknowledge signals are combined in the Completion Detection circuitry to produce the request signal(s) to the previous register stage, utilizing either the full-word or bit-wise completion strategy¹⁸. To ensure delay-insensitivity, NCL circuits must adhere to the following criteria: Input-Completeness¹⁶ and Observability¹⁶. Furthermore, when circuits utilize the bit-wise completion strategy with selective input-incomplete components, they must also adhere to the completion-completeness criterion¹⁹, which requires that completion signals only be generated such that no two adjacent DATA wavefronts can interact within any combinational component.

NCL systems consist of Registration, Combinational Logic, and Completion Detection, connected together as shown in Figure 1. NCL registration is realized through cascaded arrangements of single-bit dual-rail registers or single-signal quad-rail registers, depicted in Figures 4 and 5, respectively. These registers consist of TH22 gates that pass a DATA value at the input only when K_i is *request for data* (*rfd*) (i.e. logic 1) and likewise pass NULL only when K_i is *request for null* (*rfn*) (i.e. logic 0). They also contain a NOR gate to generate K_o , which is *rfn* when the register output is DATA and *rfd* when the register output is NULL. The registers shown below are reset to NULL, since all TH22 gates are reset to logic 0. However, either register could be instead reset to a DATA value by replacing exactly one of the TH22n gates with a TH22d gate.

An N -bit register stage, comprised of N single-bit dual-rail NCL registers, requires N completion signals, one for each bit. The NCL completion component, shown in Figure 6, uses these N K_o lines to detect complete DATA and NULL sets at the output of every register stage and request the next NULL and DATA set, respectively. In full-word completion, the single-bit output of the

completion component is connected to all K_i lines of the previous register stage. Since the maximum input threshold gate is the TH44 gate, the number of logic levels in the completion component for an N-bit register is given by $\lceil \log_4 N \rceil$. Likewise, the completion component for an N-bit quad-rail registration stage requires $\frac{N}{2}$ inputs, and can be realized in a similar fashion using TH44 gates.

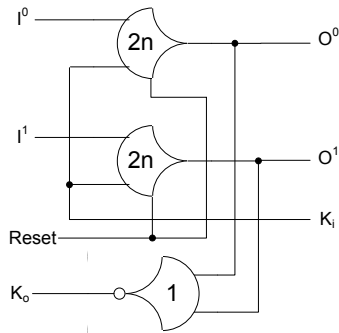


Figure 4. Single-bit dual-rail register.

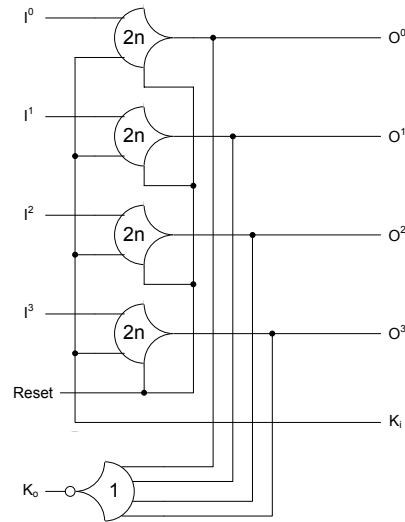


Figure 5. Single-signal quad-rail register.

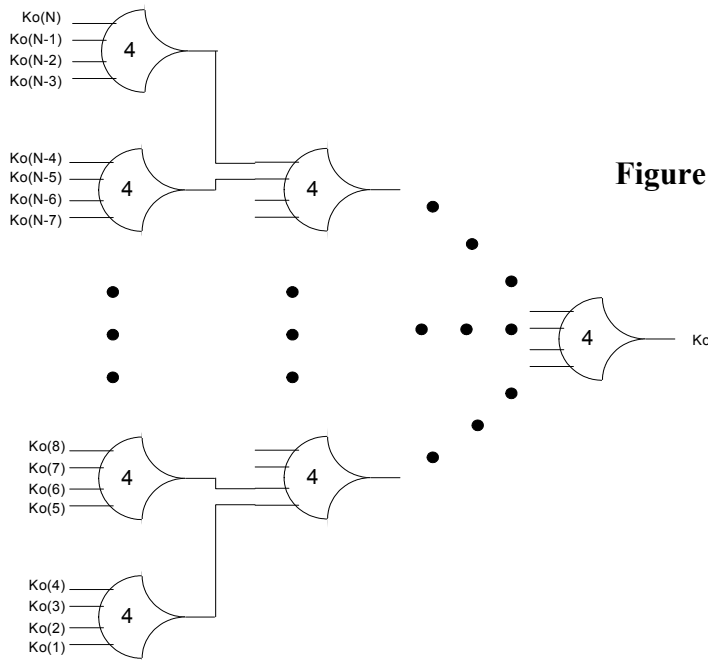


Figure 6. N-input completion component.

3. Developed Materials

To effectively introduce asynchronous digital design into the Computer Engineering curriculum, lecture notes, example problems, group projects, and libraries of fundamental asynchronous gates and components were developed. The educational materials were developed as *Modules*, such that portions of the materials could be easily integrated into a variety of courses, as appropriate, to meet the needs of a diverse set of courses with different learning objectives.

3.1 Educational Modules

The following is the list of specific educational modules that were developed:

- 1) **Introduction to Asynchronous Logic:** This includes a discussion of both bounded-delay and delay-insensitive asynchronous paradigms, highlighting the differences between the two and comparing each to the synchronous, clocked paradigm.
- 2) **Introduction to NULL Convention Logic (NCL):** This includes a description of dual-rail and quad-rail signaling, the 27 fundamental NCL gates, NCL Registration, Combinational Logic, and Completion Detection components, and NCL DATA/NULL wavefront flow.
- 3) **Transistor-level NCL Gate Design:** This details the process for designing both static and semi-static NCL gates.
- 4) **Input-Completeness and Observability:** This explains the two criteria that must be followed when designing NCL circuits to ensure delay-insensitivity.
- 5) **Dual-Rail NCL Design:** This details the process for designing dual-rail NCL combinational circuits.
- 6) **Quad-Rail NCL Design:** This details the process for designing quad-rail NCL combinational circuits.
- 7) **NCL Throughput Optimization:** This describes the NCL throughput calculation, NCL pipelining, and the NULL Cycle Reduction optimization.
- 8) **Group Projects:** This contains a number of comprehensive group projects consisting of the implementation and testing of various types of NCL arithmetic circuits, at various levels of abstraction.

All of these course modules can be downloaded from the authors' CCLI website:

http://web.umr.edu/~smithsco/CCLI_async.html. Module 1 is similar to Sections 2.0 and 2.1 in this paper, although more extensive; and Module 2 is similar to Section 2.2 in this paper. Modules 1 and 2 are introductory and therefore do not contain any specific example problems or exercises; they are also independent of each other, such that a broad discussion of asynchronous logic in general is not required before discussing NCL specifics. Modules 3-7 all contain an explanation of the specific topic along with a comprehensive example and exercise problems. Modules 2 and 4 are prerequisites for all subsequent modules, while Modules 3, 5, 6, and 7 are independent of each other. The comprehensive group projects in Module 8 require various other modules as prerequisites, depending on the specific project requirements and objectives.

3.2 Asynchronous Libraries

In order to assist students with designing and testing NCL circuits at various levels of abstraction, static NCL VHDL, transistor-level, and physical-level libraries were created. The transistor-level and physical-level libraries of the fundamental NCL gates were implemented with the Mentor Graphics CAD tools using the 0.18 μ m TSMC CMOS process. The VHDL library consists of a package that defines the fundamental NCL data types, a file containing the fundamental NCL gates, with delays based on the simulated physical-level static NCL gates, a file containing generic versions of standard NCL registration and completion components, and a package consisting of various functions to be used in testbenches. The VHDL, transistor-level, and physical-level NCL libraries can all be downloaded from the authors' CCLI website:

http://web.umr.edu/~smithsco/CCLI_async.html.

4. Course Integration

The asynchronous modules and libraries were successfully incorporated into two senior/graduate-level elective courses at University of Missouri – Rolla (UMR), *Digital System Modeling with VHDL* and *Introduction to VLSI*, in Spring Semester 2006 and Fall Semester 2006, respectively. The original schedule for the VHDL course is shown on the left-hand side of Figure 7. This provides the students with approximately 13 weeks of topic lectures, leaving around 3 weeks for discussion of homework and project assignments and their solutions, holidays, and the midterm exam. Note that the final exam is given the week after the 16-week semester concludes. This schedule has been vetted by the primary author over the past five years and has been shown to work well. It does require the students to do a sizable amount of work; however, after successful completion of the course, students are well versed in VHDL.

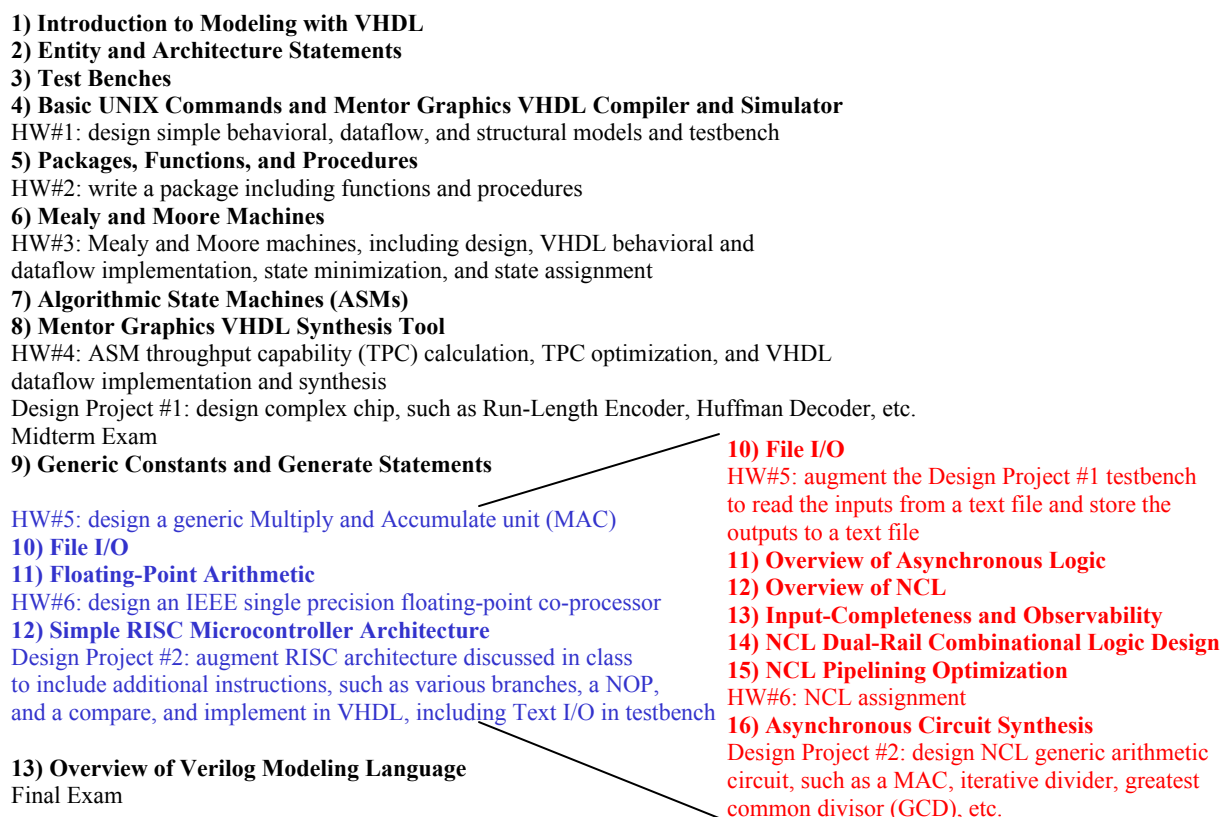


Figure 7. VHDL course schedule and changes.

To integrate the asynchronous logic material into the course, the last quarter of the original schedule was revised, as shown on the right-hand side of Figure 7. The floating-point arithmetic and microprocessor architecture topics were replaced with the asynchronous topics; HW#5 on generic constants and generate statements was changed to instead cover Text I/O; HW#6 on the design of an IEEE single precision floating-point co-processor was switched with an assignment on NCL; and Design Project #2 on implementing a microcontroller in VHDL and Text I/O was replaced with the design of a complex generic NCL arithmetic circuit. These changes replace 3 weeks of topics with 2 2/3 weeks of asynchronous logic topics, providing an extra 1/3 week for additional explanation of the NCL assignments and solutions. Furthermore, these changes do not

eliminate any key VHDL course modules; both floating-point arithmetic and RISC microcontroller architecture are covered in various other Computer Engineering courses, and were only discussed in the VHDL course so that they could be used as sample circuits to be designed in VHDL. Furthermore, since asynchronous circuits must be designed as structural models and cannot be described as behavioral or dataflow models and synthesized using industry-standard CAD tools, the topic fits seamlessly into the discussion of *generate statements*, which are mostly used for structural models.

The schedule for the revised VLSI course is shown in Figure 8. This provides the students with approximately 14 weeks of topic lectures, leaving around 2 weeks for discussion of laboratory assignments and their solutions, holidays, and occasional quizzes. Note that the final exam is scheduled the week after the 16-week semester concludes, and is utilized for each group to present their semester project design. The class requires a substantial amount of laboratory work; however, after successful completion of the course, students are well versed in VLSI design using the Mentor Graphics CAD tools.

1) Introduction to VLSI Systems

Lab#1: VHDL coding, synthesis, and simulation

2) CMOS Transistor Theory

3) Fabrication, Layout, and Design Rules

Lab#2: gate-level and transistor-level schematics and simulation

4) Analysis of Static Inverter

Lab#3: layout of static inverter and RC extraction

5) Design and Optimization of Static CMOS Gates

6) Introduction to NCL

7) Transistor-level design of NCL gates

8) Critical Path Delay Analysis and Transistor Sizing

9) Dynamic CMOS Circuit Design

10) Design of Flip-Flops, Latches, and Sequential Circuits

Lab#4: layout of basic static Boolean gates and static and semi-static NCL gates (NCL gates replaced flip-flops)

11) Static Timing Analysis for Sequential Circuits

12) Low Power Design

Lab#5: schematic driven layout

13) Datapath Design for Synchronous Circuits (e.g., comparators, adders, multipliers, registers, etc.)

14) Datapath Design for NCL Circuits (e.g., registration, completion, and DR and QR combinational circuits)

Lab#6: synchronous datapath design and simulation

15) Semiconductor Memories

16) Clock Distribution, PLL, Clock Skew, and Jitter

17) Floorplanning, Placement, and Routing

18) Control Unit Design

19) VLSI Testing and Design for Test

Design Project: design, layout, and simulate various NCL arithmetic circuits (e.g., quad-rail unsigned $24+8\times 8$ MAC, dual-rail 2^s complement 8×8 Booth2 multiplier, and dual-rail 2^s complement 8×8 Baugh-Wooley multiplier)

20) Future Trends in VLSI Design

Figure 8. VLSI course schedule and changes.

The asynchronous logic topics have been incorporated into the VLSI course by replacing previous miscellaneous lecture topics, by replacing Lab#4's layout of a flip-flop with the layout of a static and semi-static NCL gate, and by utilizing NCL circuits for the semester's comprehensive design project. The new semester design projects involve designing various NCL arithmetic circuits, using one of the industry-standard VLSI CAD tool suites, Mentor Graphics,

throughout all steps of the design flow (i.e., starting from the high level of abstraction, behavioral modeling, down to the low level of abstraction, physical layout), and proving the functional equivalence with simulations throughout all levels of abstraction.

5. Conclusions and Future Work

5.1 Evaluation of Developed Materials

Modules 1, 2, 4, 5, and 7 and the VHDL library were utilized in the VHDL class; and Modules 2-6 and the VHDL, transistor-level, and physical-level libraries were utilized in the VLSI course. Both courses also incorporated an NCL-based final project, Module 8. According to the feedback provided from UMR's end of semester student evaluation form for both courses, the students found the asynchronous logic topics very interesting, and would have liked to have been able to spend more time on NCL. Many students also stated that the libraries were easy to use and error-free. Overall, the students performed quite well on the NCL-related assignments. For the VHDL class, the average on the asynchronous logic homework assignment was the second highest of the six homeworks (i.e., 83% verses 86%, 76%, 73%, 64%, and 44%); and the asynchronous project's average was approximately the same as the first project (i.e., 85% verses 87%). However, this included one group of two students who decided not to complete the project because they were graduating, already had jobs, and already had enough points to pass the course, and therefore received a 31% on the partial submission and an overall grade of D in the class. Excluding this outlier boosts the asynchronous project's average to 91%. For the VLSI course, all students successfully completed the NCL laboratory assignment (i.e., Lab#4); and all three NCL-based semester projects worked correctly, all resulting in a conference publication with the students as first author^{20, 21, 22}.

The 8 educational modules were also evaluated externally by Dr. Jia Di from the University of Arkansas; and he rated them as excellent. In fact, he is currently working on the design of an NCL 8051 microcontroller for a NASA Phase II SBIR, and has required his graduate students working on the project to download Modules 2-5, study them, and complete the related exercise problems. Furthermore, he is utilizing the authors' VHDL library for the NCL 8051 functional design, although he is using Cadence for the transistor-level and physical-level design. Dr. Di's main suggestion for improvement was to implement the transistor-level and physical-level libraries in Cadence as well, such that the libraries are available for use with the three most prevalent digital design tool suites (i.e., Mentor Graphics, Synopsys, and Cadence), which are used in almost all U.S. universities. Note that the VHDL library is platform independent and is therefore already compatible with Synopsys.

5.2 Future Work

The authors are planning to expand upon this work through the following:

- 1) **Develop new educational modules focusing on additional asynchronous circuit topics**, such that asynchronous circuit concepts can be incorporated into a larger variety of Computer Engineering courses.
- 2) **Develop semi-static VHDL, transistor-level, and physical-level libraries of fundamental asynchronous components**, such that students can easily compare asynchronous circuits designed using static vs. semi-static gates, in terms of speed, area, and energy usage.

- 3) **Complete the development of NCL design and optimization CAD tools**, which work with the Mentor Graphics design tool suite, such that students can design and test large NCL circuits and can study the operation of the asynchronous CAD tools in the context of their synchronous counterparts.
- 4) **Port the static and semi-static libraries to Cadence, and the NCL CAD tools to Synopsys**, such that the libraries and CAD tools are available for use with the three most prevalent digital design tool suites (i.e., Mentor Graphics, Synopsys, and Cadence), which are used in almost all U.S. universities.
- 5) **Develop an asynchronous FPGA**, such that students can implement and test their asynchronous circuit designs in hardware.
- 6) **Broadly disseminate the developed materials to faculty members at other institutions, and integrate and evaluate the materials through course offerings at numerous institutions throughout the nation.**

Overall, the developed materials provide an easy way to integrate cutting-edge technology into standard educational practices to provide a low-cost, innovative addition to the Computer Engineering curriculum, which will prepare students for the challenges faced by the digital design community for years to come.

Bibliography

1. <http://www.itrs.net/Links/2003ITRS/Design2003.pdf> (available March 2007).
2. <http://www.itrs.net/Links/2005ITRS/Design2005.pdf> (available March 2007).
3. I. E. Sutherland, "Micropipelines," *Communications of the ACM*, Vol. 32/6, pp. 720-738, 1989.
4. A.J. Martin, "Programming in VLSI: From Communicating Processes to Delay-Insensitive Circuits," in *Developments in Concurrency and Communication*, UT Year of Programming Institute on Concurrent Programming, Addison-Wesley, 1990, pp. 1-64.
5. K. Van Berkel, "Beware the Isochronic Fork," *Integration, the VLSI Journal*, Vol. 13/2, pp. 103-128, 1992.
6. Y. Kim and F. Lombardi, "Guest Editors' Introduction: Clockless VLSI Systems," *IEEE Design and Test of Computers: Special Issue on Clockless VLSI Design*, Vol. 30/6, pp. 26-36, November-December 2003.
7. D. E. Muller, "Asynchronous Logics and Application to Information Processing," in *Switching Theory in Space Technology*, Stanford University Press, pp. 289-297, 1963.
8. C. L. Seitz, "System Timing," in *Introduction to VLSI Systems*, Addison-Wesley, pp. 218-262, 1980.
9. J. Sparso, J. Staunstrup, M. Dantzer-Sorensen, "Design of Delay Insensitive Circuits using Multi-Ring Structures," *Proceedings of the European Design Automation Conference*, pp. 15-20, 1992.
10. T. S. Anantharaman, "A Delay Insensitive Regular Expression Recognizer," *IEEE VLSI Technical Bulletin*, Sept. 1986.
11. N. P. Singh, *A Design Methodology for Self-Timed Systems*, Master's Thesis, MIT/LCS/TR-258, Laboratory for Computer Science, MIT, 1981.
12. I. David, R. Ginosar, and M. Yoeli, "An Efficient Implementation of Boolean Functions as Self-Timed Circuits," *IEEE Transactions on Computers*, Vol. 41/1, pp. 2-10, 1992.
13. D. H. Linder and J. H. Harden, "Phased logic: supporting the synchronous design paradigm with delay-insensitive circuitry," *IEEE Transactions on Computers*, Vol. 45/9, pp. 1031-1044, 1996.

14. K. M. Fant and S. A. Brandt, "NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," *International Conference on Application Specific Systems, Architectures, and Processors*, pp. 261-273, 1996.
15. Gerald E. Sobelman and Karl M. Fant, "CMOS Circuit Design of Threshold Gates with Hysteresis," *IEEE International Symposium on Circuits and Systems (II)*, pp. 61-65, 1998.
16. S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb, "Optimization of NULL Convention Self-Timed Circuits," *Integration, the VLSI Journal*, Vol. 37/3, pp. 135-165, August 2004.
17. A. J. Martin, "Compiling Communicating Processes into Delay-Insensitive VLSI Circuits," *Distributed Computing*, Vol. 1/4, pp. 226-234, 1986.
18. S. C. Smith, R. F. DeMara, J. S. Yuan, M. Hagedorn, and D. Ferguson, "Delay-Insensitive Gate-Level Pipelining," *Integration, the VLSI Journal*, Vol. 30/2, pp. 103-131, October 2001.
19. S. C. Smith, "Completion-Completeness for NULL Convention Digital Circuits Utilizing the Bit-wise Completion Strategy," *International Conference on VLSI*, pp. 143-149, June 2003.
20. M. V. Joshi, S. Gosavi, V. Jagadeesan, A. Basu, S. Jaiswal, W. K. Al-Assadi, and S. C. Smith, "NCL Implementation of Dual-Rail 2^s Complement 8×8 Booth2 Multiplier using Static and Semi-Static Primitives," *IEEE Region 5 Technical Conference*, April 2007.
21. S. R. Mallepalli, S. Kakarla, S. Burugapalli, S. Beerla, S. Kotla, P. K. Sunkara, W. K. Al-Assadi, and S. C. Smith, "Implementation of Static and Semi-Static Versions of a Quad-Rail NCL $24+8 \times 8$ Multiply and Accumulate Unit," *IEEE Region 5 Technical Conference*, April 2007.
22. R. S. P. Nair, F. Kacani, R. Bonam, S. M. Gandla, S. K. Chitneni, V. Kadiyala, W. K. Al-Assadi, and S. C. Smith, "Implementation of Static and Semi-Static Versions of a Bit-Wise Pipelined Dual-Rail NCL 2^s Complement Multiplier," *IEEE Region 5 Technical Conference*, April 2007.