Cell Designs for Self-Timed FPGAs

Cherrice Traver, Robert B. Reese, Mitch A. Thornton

***Abstract*-- A self-timed programmable architecture used for the implementation of Phased Logic (PL) systems is described. PL systems are automatically translated from clocked designs and result in self-timed circuits that are insensitive to delays between gates. The target implementation is a self-timed FPGA architecture composed of PL gates. A PL gate design based on a 4-input lookup table is presented. Power and performance estimates of two designs are given and are compared to their clocked counterparts.**

***Index Terms*--self-timed, asynchronous, programmable**

I. Introduction

THROUGHOUT the ITRS-99 Roadmap on Design, continual references are made to clock and timing related challenges facing designers through and beyond 2005. Many of the problems stem from simply taking the current design methodology for computing systems (a global clock between components, with higher rate clocks within components) and projecting this usage within the System-on-a-Chip circa 2005 and beyond. Asynchronous and self-timed systems have been proposed in the past to avoid global clock timing and to achieve improved performance, noise immunity, and low power [1]. These systems typically require special design methods and result in circuits with area overhead to implement asynchronous signaling such as dual-rail or bundled data and handshaking. In this paper an architecture for a programmable logic implementation of Phased Logic (PL) systems is described [2] [3] [4]. This approach takes advantage of the well-established synchronous design methodology and commercial synthesis tools. Level Encoded two-phase Dual-Rail (LEDR) encoding of signals [5] [6] and phased logic gates are used in the implementation. The resulting self-timed PL systems are insensitive to delays between PL gates. The area overhead of the phased logic gates using a lookup table (LUT) implementation of the logic function is comparable to the overhead in synchronous FPGA and CPLD logic cells and we show some potential advantages in performance and power. In section II we will put this work in context of others who have researched similar topics. This will be followed by an overview of the PL design method and a specification of the PL gate behavior. Section IV will describe the PL gate design based on a 4-input LUT, Section V will describe an enhanced version of this gate that takes advantage of data-dependent early evaluation, and Section VI will present some power and performance results from designs composed from these gates. A summary and description of further work is given in Section VII.

This work was supported in part by an internal grant from the MSU/NSF Engineering Research Center..

Cherrice Traver is with the Electrical Engineering and Computer Science Department, Union College, Schenectady N.Y., USA, (e-mail:traverc@doc.union.edu).

Robert B. Reese is with the Electrical and Computer Engineering Department, Mississippi State University, Starkville MS, USA, (e-mail:reese@ece.msstate.edu).

Mitch A. Thornton is with the Electrical and Computer Engineering Department, Mississippi State University, Starkville MS, USA, (e-mail:mitch@ece.msstate.edu).

II. Background

A self-timed FPGA based upon 3-input LUTs (LUT3) and using LEDR encoding was presented by How in [7]. The author uses the cell in the context of Sutherland's micropipelines [14] and self-timed iterative rings [15] and provides support specifically for those architectures. Another FPGA architecture for asynchronous logic was proposed in [16]. Two types of function blocks were provided; a LUT3+Dlatch block and a block that could implement an arbiter, enabled arbiter, or synchronizer. This FPGA architecture was aimed at accommodating a range of asynchronous design styles, and allowed for mixed synchronous and asynchronous designs. By contrast, our proposed function block is aimed at only supporting the PL design style, and would implement PL designs more efficiently than [16] or [7].

The asynchronous design methodology known as Null Convention Logic (NCL) also offers automated synthesis of asynchronous designs using commercial synthesis tools [[8]. NCL uses a NULL/DATA/NULL encoding instead of LEDR and their designs are targeted at standard cell implementations [17] [18] rather than FPGAs.

In comparing physical implementation characteristics, NCL has some delay sensitivity between NCL gates whereas PL systems are insensitive to delays between PL gates.

III. Phased Logic Design

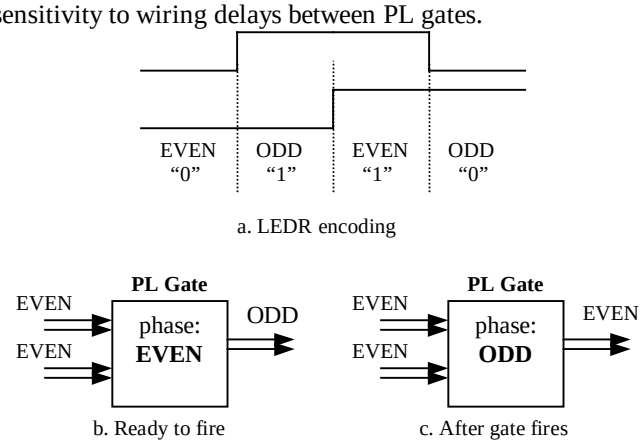
A. PL Design Methodology

Phased Logic designs are specified in VHDL RTL and synthesized by Synopsys Design Compiler to a netlist of D flip-flops (DFF) and 4-input logic gates. This netlist is then translated to a netlist of PL gates.

B. LEDR Encoding and Netlist Translation

A Phased Logic netlist can be thought of as a marked graph with data tokens flowing throughout the graph. Each data token, encoded using LEDR, has a *phase* that is either *even* or *odd*. A PL gate also has an internal even or odd phase (state). A PL gate *fires* and changes phase whenever all of the phases of the inputs matches the internal gate phase.

Figure 1a illustrates the encoding of the dual rail signals used between PL gates. Notice that the "value signal" (top), is the logical value and two signals together define the phase. A sample gate firing is shown in 1b. The controlled firing of PL gates is what gives a phased logic system its delay insensitivity to wiring delays between PL gates.



a. LEDR encoding

b. Ready to fire

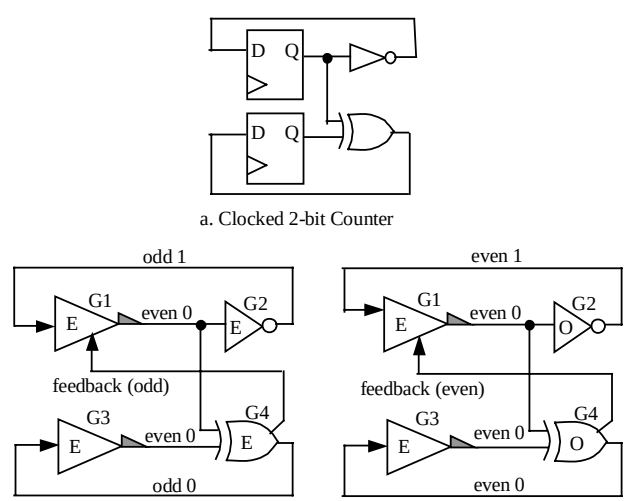
c. After gate fires

Figure 1. LEDR Encoding and PL Gate Firing

The mapping of a clocked netlist (DFFs+ combinational logic) to a phased logic netlist consists of the following operations:

- All gates are replaced by PL gate equivalents. A PL gate equivalent is the original combinational logic function plus the control logic necessary for gate firing. A DFF in the original netlist can be absorbed into the combinational gate that provides the DFF's input signal.
- Feedback signals are synthesized for the PL netlist to ensure safety and liveness of the resulting netlist. Feedback signals carry no value information, just phase information and are single wires. The details of feedback generation are described in [2] [3] [4]. Muller C-elements [9] [10]are used to concentrate feedbacks at a particular gate when needed.

Figure 2. Translation and First Fire of 2-bit Counter



a. Clocked 2-bit Counter

b. Translated PL Counter (reset state)

c. PL Counter after G2,G4 fire

Figure 2 illustrates the translation of a 2-bit counter to a PL netlist and a sample firing of the circuit. Buffers replace the DFFs in this case, although most of the time they can be absorbed into neighboring gates. The "wedge" placed on the outputs of gates G1 and G2 indicate that he net is connected to the inverted phase outputs of the gate to ensure that a gate is ready to fire in the loops formed by gates G1-G2, G1-G4 and G3-G4. The feedback net from gate G4 to G1 is added to make the netlist safe. Figure 2a shows the circuit after reset and Figure 2b shows the state of the circuit after it fires once.

C. PL Gate Behavior

Each PL gate must compute a logical function and enforce the firing rule. The logical function is the function translated from the synchronous netlist, such as the XOR and NOT in Figure 2. The firing rule specifies that the PL gate wait until its inputs (including the feedback inputs) are the same phase as the gate before it fires. When the gate fires, the internal phase changes, the value signal of the output becomes the logical output, and the output phase becomes the opposite of the gate phase.

IV. PHASED LOGIC GATE DESIGN

In [3] a SRAM-based PL gate is described. This gate uses decoders and an SRAM to implement the detection of LEDR inputs and gate firing (control) as well as the computation of the logical function. This means that the SRAM array is read anytime the input phases match, even if the data values have not changed. This has a negative impact on power consumption in a phased logic system since the logical function must be switched every time the gate fires.

A new gate design, illustrated in Figure 3, has been developed with separate control and compute logic. This separation is important as it eliminates computation transitions for cases when the gate fires, but the input values remain constant.

In this gate design, the input completion is detected using a Muller-C element (C-gate) and the gate phase is held at its output. The LUT4 computes the logical function in parallel with the input completion. When a new set of inputs are detected, the C-gate toggles and causes the output latches to be updated. The delay block is necessary to ensure that the internal timing constraints are met.

A. PL Gate Delay Constraints

The delay constraints that must be met for proper operation of the gate are derived from the timing requirements of the latches and the PL gate firing rules. A D-latch must have a minimum pulse width on the enable input, and the D input must be stable one setup time before the trailing edge of the pulse. In addition, the firing rule requires that the PL gate t and v outputs change only once per gate fire, which puts a further restriction on the D inputs: that they arrive before the latch is enabled.

When these constraints are applied to the PL gate design,

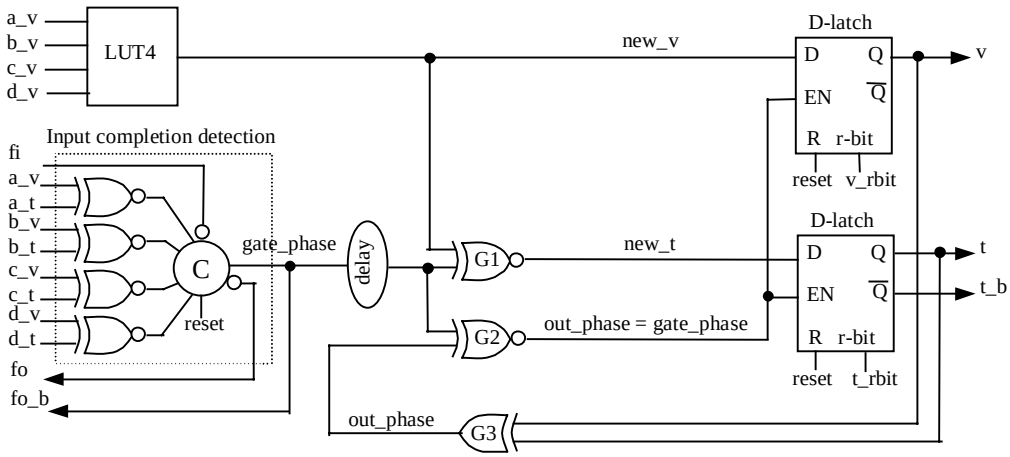


Figure 3. LUT4-based PL Gate

we find that the delay of the completion logic ($D_{complete}$) and delay element (D_{delay}) must be greater than the delay of the LUT4 gate (D_{LUT4}). This ensures that the new_v value is defined before the latches are enabled. In addition, the delay of G2 (D_{G2}) must be larger than the delay of G1 (D_{G1}) so that only the final value of new_t is latched, enforcing the requirement of a single transition per PL gate fire. Since the pulse width of the enable signal is defined by the enable to Q delay of the latch ($D_{EN\text{-}Q}$) and the delays of G3 and G4, the sum of these delays must be greater than the minimum pulse width of the latch ($T_{pulse\text{-}min}$). These one-sided delay constraints are not difficult to satisfy in the implementation.

$$D_{complete} + D_{delay} > D_{LUT4} \quad (1)$$

$$D_{G2} > D_{G1} \quad (2)$$

$$D_{EN\text{-}Q} + D_{G2} + D_{G2} > T_{pulse\text{-}min} \quad (3)$$

The worst case delay through this gate is given by the sum of the completion logic delay, the delay block, Dg2 and the latch delay.

$$D_{PL\text{-}GATE} = D_{complete} + D_{delay} + D_{EN\text{-}Q} \quad (4)$$

Note that the PL gate delay can be approximated by the sum of the LUT4 and the latch delay.

V. An Early-Evaluation PL Gate

Because there is no global clock, the performance of any self-timed system is a function of the average case of data-dependent delays in the system. A feature that improves the performance of PL systems by speeding up computation for some cases is called *early evaluation* (EE). Early evaluation is a speedup mechanism by which a PL gate is allowed to fire when only a subset of its inputs has arrived. Figure 4 illustrates how two PL4gates (termed a *master* and a *trigger*) can be combined to form one early evaluation PL4gate. The master contains the normal evaluation function while the trigger contains the early evaluation function. When the early evaluation function is true, then the master gate is fired and the current value of the master LUT4 is used. Obviously the trigger gate should be based upon early arriving signals, and the trigger function should depend upon a subset of the master function signals such that the late arriving signals are don't cares.

An example of an early evaluation function is Carry generation within a ripple carry adder. The master LUT would have the normal carry equation, while the trigger LUT would have the function F = AB + A'B', which causes the carry to be generated early for either generate (AB) or kill (A'B'). Note that the Cin value for these cases is a don't care, so the master gate can be fired early.

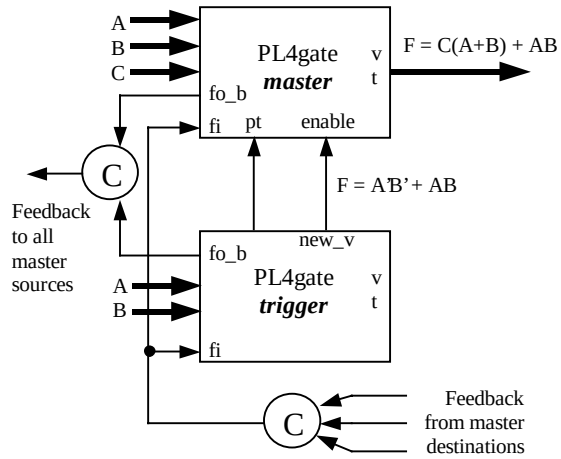


Figure 4. Early Evaluation Gate from Two PL4gates

A. Early Evaluate Gate Operation

The trigger gate shown in Figure 4 is the same gate as shown as in Figure 3. The master gate is augmented with additional gating to support early evaluation. Configuration bits in the master gate allow the gate pair to function as two independent PL gates, or as one Early Evaluation (EE) gate. Two signals link the master gate and trigger gate during early evaluation operation - the output of the trigger gate LUT4 (*enable*) and the phase of the trigger gate (*pt*). An early firing occurs if the trigger gate fires before the master gate and if the enable from the trigger gate is a '1'. The circuitry

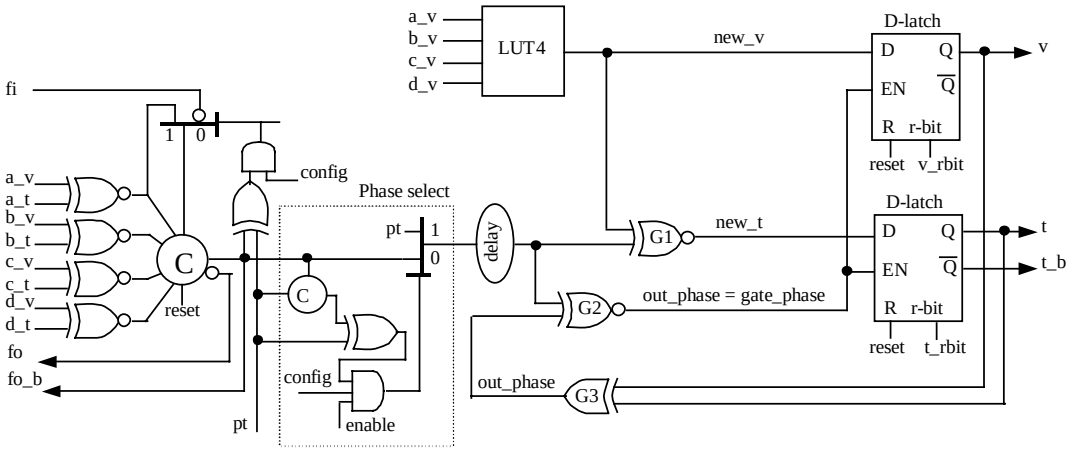


Figure 5. Master gate for Early Evaluation

in the master gate labeled as *phase select* (see Figure 5) in the master gate detects an early fire (enable = '1' and master phase not equal to trigger phase) and uses the trigger phase for the output phase.

Early Evaluation (EE) gates cause a slight modification of the feedback generation rules presented in [2]. Because it is not known which gate will fire first (master or trigger), both the master feedback signal and the trigger feedback signal must go to all master sources (these two signals can be combined via a 2-input C element to form one feedback signal for the master/trigger pair). Also, feedback signals from a master destination must terminate on both the master and trigger gate instead of being allowed to terminate on a source feeding the master/trigger pair. This is needed in order to provide a single point of control for a preventing a possible deadlock situation. The deadlock arises if a destination gate for the master's output value consumes an early 'fire' output value and provides the feedback signal before all of the tardy inputs have arrived at the master. This causes the feedback signal to be out of phase with the tardy inputs, preventing the master gate from ever firing again. The master gate in Figure 4 has additional gating on the feedback input that causes the master gate to ignore the feedback input until all inputs have arrived only in the case of an early firing (trigger phase not equal to master phase).

B. EE-PL Gate Delay Constraints

The equations presented in section IV can be used for master gate operation as long as the extra delay due to the phase select circuitry (Figure 5) is lumped into the input completion detection delay ($D_{complete}$) of equation (1). In order to determine the minimum value of D_{delay} we need to consider the shortest path through the selection logic of Figure 5. After a completed EE gate firing, both master phase and trigger phase will be equal, selecting the master phase as the phase value for the next output value. It is easy to see that this is the shortest path of the selection logic and is exercised for a non-early firing of the master/trigger pair. An early firing (trigger fires first) will exercise a longer delay path in the select circuit, causing the master output latches to be enabled well after the master LUT value is ready.

VI. Power and Performance Comparisons

In this section we present two PL design examples and compare them to their clocked counterparts. We do not have enough design experience with PL circuits yet to draw any sweeping conclusions, and present these for information purposes only. For clocked versus PL delay comparisons, we assigned a LUT4 a normalized delay of 1.0, a PL gate (Figure 4) a delay of 1.4 (40% output latch delay penalty), and an early firing of a master/trigger pair as 1.6 (recall that an early firing exercises a longer delay path in the master gate phase selection logic). For power comparisons, the VHDL simulations are instrumented to track compute and control signal transitions. A compute transition is counted as any change of value on a LUT4 input (simultaneous or near simultaneous arrivals were only counted once). For the clocked netlist, an active clock edge arrival at a D-Flip-Flop is a control transition. For the PL netlist, the firing of a PL gate is a control transition. From Altera Apex [11] and Xilinx Virtex [12] FPGA power estimation spreadsheets for 0.18µ technologies, a LUT4 is estimated to switch approximately 1.05 pF, and a D-Flip-Flop 0.14 pF. HSPICE simulations based on a 0.25µ technology of the *pl4gate* control indicate that it switches approximately 0.2 pF per firing (this value would be expected to shrink somewhat for a 0.18µ process but we will use this somewhat inflated value so as to help factor in the output token phase or 't' bit wiring capacitance). Four-input Muller C-elements were found to switch 0.75 pF per output change.

The first example design is a 32-bit accumulator with a synchronous clear. A carry look-ahead adder is used for the clocked netlist and a ripple carry adder with early evaluation is used for the PL netlist. Figure 6 shows the second design, a shift/add 16x16 iterative multiplier using a single adder. Again, the clocked design used a CLA while the PL design used a ripple adder with early evaluation. The PL design had an extra speedup path in that a kill line was used to early fire

every carry bit in case the multiplier bit was '0'. This did not cost any extra LUT4s since there was a free input available on the LUT4 for the carry function. A multiplication required 17 clocks, one clock for input of multiplier and multiplicand operands, and 16 clocks for computation. A four bit counter was used in the FSM for counting purposes. Handshaking lines *irdy* and *ordy* were used for I/O purposes.

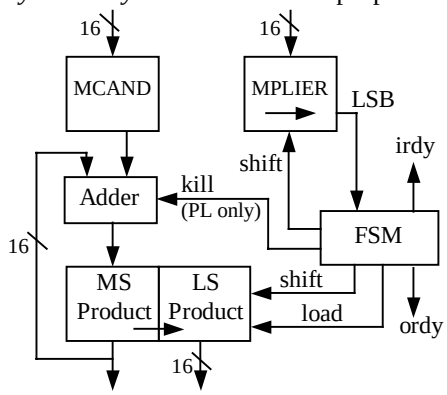


Figure 6. 16 x 16 Iterative Multiplier

Table 1 gives the delay, capacitance and energy per sample measured from the VHDL gate level simulations. The delay values are normalized to LUT4 delays. The energy value is simply a figure of merit for work obtained by multiplying the capacitance column times the delay column, and dividing by a constant scale factor.

Table 1: Performance values for Design Examples

Design	dly(LUT4s)	cap(fF)	Energy	%diff
Clk (acc)	12	205	24.7	
PL (acc)	8.2	193.1	15.8	-36.0%
Clk (mult)	187	3415	6385	
PL (mult)	151	3557	5357	-16.1%

The accumulator design is designated as 'acc' and the iterative multiplier as 'mult'. In both cases, the PL designs are more energy efficient than the clocked designs. The PL iterative multiplier actually switched slightly more capacitance than the clocked design, but was more energy efficient due to higher performance.

In looking at the performance figures of Table 1, the ripple early evaluation capability allows the PL designs to overcome the 40% gate delay penalty and to outperform the clocked LUT4 netlists. We are aware that both Xilinx and Altera include fast carry generation logic within their cells, while our clocked netlists had the carry logic as a dedicated LUT4. Obviously, the carry logic along with the early evaluation for the carry could be integrated into the *pl4gate* design. The exact effect upon the performance and capacitance values in Table 1 is an area of future study.

Table 2 shows the transition counts from which the capacitance values of Table 1 were generated. The reduction in compute transitions in the PL designs is due the different adder structures (CLA versus ripple) and also from the filtering of transient computations.

Table 2: Transition Counts for Design Examples

Design	Compute	%diff	Control	Cgate
Clk (acc)	110133		38115	0
PL (acc)	81031	-26.4%	112135	80454
Clk (mult)	1806621		791863	0
PL (mult)	1155446	-36.0%	3735097	1750466

While PL designs can reduce compute transitions, PL dramatically increases control transitions since there is now control in every cell, and every cell changes phase during a compute cycle. Because of this control overhead, it is critical that the ratio of compute capacitance to control capacitance be large. In our LUT4 based *pl4gate* design, this ratio is 5 to 1, and we believe that this might be near the lowest ratio in order for a PL system to be power competitive with clocked control.

VII. Summary and Future Work

We have presented LUT4-based cell designs for a self-timed design methodology called Phased Logic. The cell design supports data dependent computation by pairing two normal gates into a master/trigger combination that allows firing of the pair based upon arrival of a subset of the inputs. We presented two sample designs that made use of this early evaluation capability. Comparison of the two designs to their clocked counterparts showed that the designs were competitive in both performance and power. The impact of early evaluation on more complex designs is an area of future study.

References

[1] C. H. (Kees) van Berkel, Mark B. Josephs, and Steven M. Nowick. Scanning the technology: Applications of asynchronous circuits. Proceedings of the IEEE, 87(2):223-233, February 1999.

[2] Daniel H. Linder and James C. Harden, "Phased Logic: Supporting the Synchronous Design Paradigm with Delay-insensitive Circuitry." *IEEE Transactions on Computers*, Vol 45, No 9, September 1996.

[3] Daniel H. Linder, *Phased Logic: A Design Methodology for Delay-Insensitive Synchronous Circuitry*, PhD thesis, Mississippi State Univ., 1994.

[4] R. Reese, and C. Traver, "Synthesis and Simulation of Phased Logic Systems", Technical Report MSSU-COE-ERC-00-09, MSU/NSF Engineering Research Center, June 2000. Presented at International Workshop on Logic Synthesis (IWLS 2000), Dana Point, CA, June 2, 2000.

[5] A.J. McAuley, "Four State Asynchronous Architectures," *IEEE Transactions on Computers*, vol. 41, February 1992.

[6] M.E. Dean, T.E. Williams, and D.L. Dill, "Efficient Self-Timing with Level-Encoded 2-Phase Dual-Rail (LEDR)," in *Advanced Research in VLSI*, 1991.

[7] Dana L. How, "A Self Clocked FPGA for General Purpose Logic Emulation", in proceedings of *IEEE 1996 Custom Integrated Circuits Conference*, 1996, pp. 148-151.

[8] Michiel Ligthart, Karl Fant, Ross Smith, Alexander Taubin, Alex Kondratyev, "Asynchronous Design Using Commercial HDL Synthesis Tools", In *Sixth Int. Symp. on Advanced Research in Asynchronous Circuits and Systems (Async 2000)*, Eilat, Israel, April 2000.

[9] D.E. Muller and W. S. Bartky, "A Theory of Asynchronous Circuits", in *Proc. Int. Symp. on Theory of Switching*, vol. 29, pp.204-243, 1959.

[10] Tzyh-Yung Wuu and Sarma B. K. Vrudhula, "A Design of a Fast and Area Efficient Mult-Input Muller C-element", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol 1, No. 2, June 1993.

[11] Altera Apex Power Estimator, http://www.altera.com/html/products/power_calc.html

[12] Xilinx Virtex Power Estimator, http://www.xilinx.com/support/techsup/powerest/index.htm

[13] Rabaey, Jan M., *Digital Integrated Circuits: A Design Perspective*, Prentice Hall, pp 408-412.

[14] I. Sutherland, "Micropipelines", *Communications of the ACM*, Vol 32, No. 6, June 1989, pp. 720-738.

[15] M.R. Greenstreet, T.E. Williams, and J . Staunstrup, "Self-Timed Iteration", *VLSI '87*, C. H. Sequin (Ed.), Elsevier Science Publishers, 1988, pp. 309-322.

[16] Scot Hauck, Steven Burns, Gaetano Borriello, Carl Ebeling, "An FPGA for Implementing Asychronous Circuits", *IEEE Design and Test of Computers*, Fall 1994, pp. 60-69.

[17] Gerald E. Sobelman, Karl Fant, "CMOS Circuit Design of Threshold Gates With Hysteresis", In *1998 IEEE International Symposium on Circuits and Systems*, Monterey, CA, May 1998.

[18] Ross Smith, Karl Fant, Dave Parker, Rick Stephani, Ching-Yi Wang, "An Asychronous 2-D Discrete Cosine Transform Chip", In *Fourth Int. Symp. on Advanced Research in Asychronous Circuits and Systems (Async'98)* , San Diego, California, March, 1998.