IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735
PP 01-10
www.iosrjournals.org

VLSI Implementation of Self Time Adder Using Recursive Approach

M.Anand [1], A.Saravanan Asst Proff [2]
PG scholor ME/ VLSI, Magabarathi Engineering College, Chinnasalem
Department of ECE, Magabarathi Engineering College,Chinnasalem

Abstract: *A brief presents a parallel single-rail self-timed adder. It is based on a recursive formulation for performing multibit binary addition. The operation is parallel for those bits that do not need any carry chain propagation. Thus, the design attains logarithmic performance over random operand conditions without any special speedup circuitry or look-ahead schema. A practical implementation is provided along with a completion detection unit. The implementation is regular and does not have any practical limitations of high fanouts. A high fan-in gate is required though but this is unavoidable for asynchronous logic and is managed by connecting the transistors in parallel. Simulations have been performed using industry standard toolkits that verify the practicality and superiority of the proposed approach over existing asynchronous adders.*
Keywords: *Asynchronous circuits, binary adders, CMOS design, digital arithmetic.*

I. INTRODUCTION

A majority of the present-day digital systems are clock based or synchronous, which assume that signals are binary and time is discrete. In general, synchronous systems comprise a number of subsystems that change from one state to another depending on a global clock signal, with flip-flops (registers) being used to store the different states of the subsystems. A conventional synchronous system is portrayed by figure 1.1. The state updates within the registers are carried out on the rising edge (positive edge) or falling edge (negative edge) of the global clock – single edge triggering. The state of the global clock permits either data loading or data storage. Since the overall clock utilization is only 50% for single edge triggered systems, double edge triggered flip-flops were subsequently proposed in the literature with the motive of increasing the system throughput as data can be loaded on both the rising and falling clock edges and data is retained when the clock signal does not toggle [1] [2]. However, this usually comes at the expense of a larger

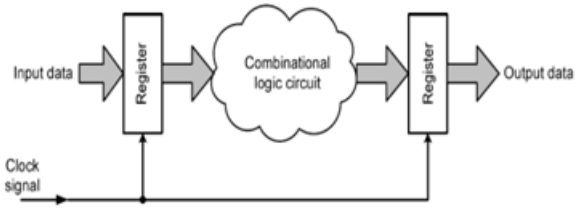

Fig 1.1: A typical synchronous system stage

Silicon footprint due to greater number of transistors and more interconnects for the dual edge triggered flip-flop and consequently leads to more power consumption. Preserving the original data rate as that of single edge triggered flip-flop designs whilst operating at half the system clock frequency might be helpful in reducing the dynamic power dissipation as the transitions could be reduced by half, but eventually this may be offset by more leakage power dissipation [2], which is becoming dominant in deep submicron technologies. Moreover, this mechanism tends to forego the advantages associated with single edge triggering in that its set-up and hold times are larger compared to conventional flip-flops and any deviation from its 50% duty cycle can lead to timing failures in critical paths upsetting the system behaviour [3]. In addition, it is more sensitive to noise apart from introducing complexity in system design and as such, the specification on jitter tolerance is more stringent which complicates the design of the system phase lock loop. As a result, synchronous designs with rising or falling edge triggering have been predominant, being the mainstream of digital system architectures; nevertheless, it is becoming increasingly difficult to overcome some fundamental limitations inherent in this approach.

The International Technology Roadmap for Semiconductors (ITRS) predicts that system-wide synchronization is becoming infeasible owing to increasing silicon complexity. A clock-based system can operate correctly only if all parts of the system see the clock at the same time, which can happen only if the delay on the clock wire is negligible. However, with advances in technology, the systems tend to get bigger and bigger in terms of the number of transistors and as a result the delay on the clock wires can no longer be ignored. The problem of clock skew is thus a major bottleneck for many system designers. Since the clock signal controls all flip-flops to sample and store their input data synchronously, it tends to be highly loaded and the problem becomes more severe. A widely preferred solution is to distribute the global clock using a clock network (clock tree) with clock buffers and thereby control the clock skew. Consequently, this results in an increase in the capacitance of the clock net and also suffers from increased activity (typically two transitions per net per cycle), even ignoring possible hazard activity on such nets.

The primary factors that govern the clock skew in a typical synchronous digital system are as follows:

- Resistance, capacitance and inductance of the interconnection material used for the clock distribution network
- Clock distribution network architecture, buffering schemes and clock buffers used · fabrication process variation over the chip area
- Number of processing elements in the system and the load presented by each element to the clock distribution network
- Rise and fall times and the clock frequency

Various clock distribution strategies have been developed, with the most common and general approach being the use of buffered trees for equipotential clock distribution. However, to distribute high-speed clock signals, symmetric trees like the H-tree are preferred compared to the asymmetric buffered clock distribution tree structure. The H-tree network is the most widely used clock distribution network [4] – [6] to minimize the clock skew. It was shown in [7] that for an N × N array of processing elements, the clock pulse rise time and the clock skew associated with it are O(N3). Hence, with increase in N, the clock skew is likely to increase rapidly and become a stumbling block. Therefore, a distributed buffering scheme is often resorted to for synchronous digital integrated circuits by introducing buffers in the clock distribution network. However, the disadvantages of this approach are the extra area overhead and the increase in design sensitivity to process variations. Also, it has to be noted that buffers are the primary source of the total clock skew within a well-balanced clock distribution network. Since global clock periods are now commonly less than half a nanosecond, variations in delay by tens of picoseconds can seriously degrade the performance and reliability of high-speed synchronous systems [8]. With Moore's law [9] having been a driving force through process generations, supported by continual innovations in processes and device materials[10], to relentlessly pursue after greater integrated circuit densities, and with variability of process and device parameters assuming ever greater significance [11] [12] as devices are scaled down to more narrow dimensions, the above problem might only get exacerbated. The bottom-line is that clock management is becoming increasingly difficult and solving it in todays high-speed complex system-on-chip designs appears to be a complex and costly affair.

The second major problem faced by designers is power dissipation, which is a very important metric that has gained significance with the phenomenal growth of portable electronics. For mobile electronic applications, the average power consumption has become the most critical design concern. For maximum efficiency, all gates in the system should be performing useful work. However in synchronous systems, this is not usually the case. Consequently, synchronous systems tend to consume more power than necessary. Many gates switch unnecessarily since they are connected to the clock and not because they have to process new input data. However, to circumvent this problem, clock gating is widely employed so as not to enable those sub-systems that are not required for any useful activity. The biggest gate is the clock driver itself which might occupy considerable area and must switch even if a small part of the system has something useful to do: the global clock, in general, was found to account for 15%-45% of the system power budget [13] and in a processor case study [14], it was found to be responsible for 34% of the total system power dissipation.

1.1 Motivation and Context

The problems of clock skew and power dissipation have been the major drivers for the worldwide resurgence of interest in asynchronous design – notable major projects include [15] – [29]. The design of clock-free or asynchronous systems has thus become attractive for digital system designers during the past two decades although asynchronous logic was explored from the infancy of integrated circuit design [30] - [32]. But synchronous design provided a far more efficient vehicle for exploiting the technology in commercial applications. The 2006 Semiconductor Industry Association's (SIA's) ITRS report on design stated that the

percentage of designs driven by handshake clocking (asynchronous signaling) would rise from 11% in 2008 to 40% by 2020. The latest ITRS update on design [33] predicts that design re-use (as a percentage of all logic) would increase from a current figure of 38% to 55% by 2020. Over this period, parameter uncertainty (as a percentage effect on sign-off delay) is projected to increase from 10% to 25%. In fact, reliability has been labeled as one of the five crosscutting design challenges, which drives home the point that design robustness is becoming an increasing priority in deep submicron technologies.

The above projections tend to forecast and necessitate a considerable shift in the design paradigm from conventional synchronous logic to asynchronous logic, as the latter benefits owing to its ability to tolerate supply voltage, process parameter and temperature variations [15]. Due to the absence of a global clock reference, asynchronous circuits tend to have better noise and electro-magnetic compatibility properties than synchronous circuits [34]. Also, they feature greater modularity permitting convenient design reuse [36]. Asynchronous operation by itself does not imply low power, but often suggests low power opportunities based on the observation that asynchronous circuits only consume power when and where active [35] [36]. The recent demonstration of the potential advantages of the world's first 8-bit physically flexible asynchronous microprocessor design over a synchronous flexible version in terms of power and noise figures by Karaki et al. from Seiko Epson's Technology Platform Research Centre [37], which utilizes 4-phase handshaking and quasi-delay-insensitive design style, endorses the future of self timed design techniques for even unconventional electronics.

1.2 Existing System

Binary addition is the single most important operation that a processor performs. Most of the adders have been designed for synchronous circuits even though there is a strong interest in lockless/ asynchronous processors/circuits [1]. Asynchronous circuits do not assume any quantization of time. Therefore, they hold great potential for logic design as they are free from several problems of clocked (synchronous) circuits. In principle, logic flow in asynchronous circuits is controlled by a request-acknowledgment handshaking protocol to establish a pipeline in the absence of clocks. Explicit handshaking blocks for small elements, such as bit adders, are expensive. Therefore, it is implicitly and efficiently managed using dual-rail carry propagation in adders. A valid dual-rail carry output also provides acknowledgment from a single-bit adder block. Thus, asynchronous adders are either based on full dual-rail encoding of all signals (more formally using null convention logic [2] that uses symbolically correct logic instead of Boolean logic) or pipelined operation using single-rail data encoding and dual-rail carry representation for acknowledgments. While these constructs add robustness to circuit designs, they also introduce significant overhead to the average case performance benefits of asynchronous adders. Therefore, a more efficient alternative approach is worthy of consideration that can address these problems.

This brief presents an asynchronous parallel self-timed adder (PASTA) using the algorithm originally proposed in [3]. The design of PASTA is regular and uses half-adders (HAs) along with multiplexers requiring minimal interconnections. Thus, it is suitable for VLSI implementation. The design works in a parallel manner for independent carry chain blocks. The implementation in this brief is unique as it employs feedback through XOR logic gates to constitute a single-rail cyclic asynchronous sequential adder [4]. Cyclic circuits can be more resource efficient than their acyclic counterparts [5], [6]. On the other hand, wave pipelining (or maximal rate pipelining) is a technique that can apply pipelined inputs before the outputs are stabilized [7]. The proposed circuit manages automatic single-rail pipelining of the carry inputs separated by propagation and inertial delays of the gates in the circuit path. Thus, it is effectively a single rail wave-pipelined approach and quite different from conventional pipelined adders using dual-rail encoding to implicitly represent the pipelining of carry signals.

II. BACKGROUND

There are a myriad designs of binary adders and we focus here on asynchronous self-timed adders. Self-timed refers to logic circuits that depend on and/or engineer timing assumptions for the correct operation. Self-timed adders have the potential to run faster averaged for dynamic data, as early completion sensing can avoid the need for the worst case bundled delay mechanism of synchronous circuits. They can be further classified as follows.

A. Pipelined Adders Using Single-Rail Data Encoding

The asynchronous Req/Ack handshake can be used to enable the adder block as well as to establish the flow of carry signals. In most of the cases, a dual-rail carry convention is used for internal bitwise flow of carry outputs. These dual-rail signals can represent more than two logic values (invalid, 0, 1), and therefore can be used to generate bit-level acknowledgment when a bit operation is completed. Final completion is sensed when

all bit *Ack* signals are received (high). The carry-completion sensing adder is an example of a pipelined adder [8], which uses full adder (FA) functional blocks adapted for dual-rail carry. On the other hand, a speculative completion adder is proposed in [9]. It uses so-called abort logic and early completion to select the proper completion response from a number of fixed delay lines. However, the abort logic implementation is expensive due to high fan-in requirements.

B. Delay Insensitive Adders Using Dual-Rail Encoding

Delay insensitive (DI) adders are asynchronous adders that assert bundling constraints or DI operations. Therefore, they can correctly operate in presence of bounded but unknown gate and wire delays [2]. There are many variants of DI adders, such as DI ripple carry adder (DIRCA) and DI carry look-ahead adder (DICLA). DI adders use dual-rail encoding and are assumed to increase complexity. Though dual-rail encoding doubles the wire complexity, they can still be used to produce circuits nearly as efficient as that of the single-rail variants using dynamic logic or nMOS only designs. An example 40 transistors per bit DIRCA adder is presentedin [8] while the conventional CMOS RCA uses 28 transistors.

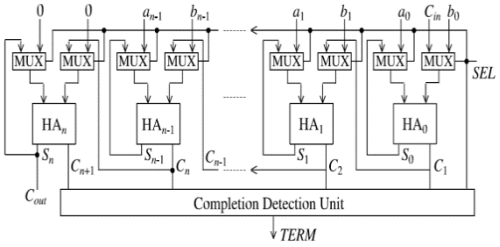

Fig. 1. General block diagram of PASTA.

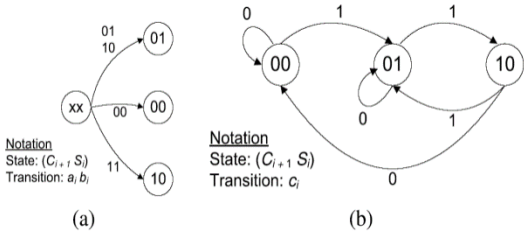

Fig. 2. State diagrams for PASTA. (a) Initial phase. (b) Iterative phase.

Similar to CLA, the DICLA defines carry propagate, generate, and kill equations in terms of dual-rail encoding [8]. They do not connect the carry signals in a chain but rather organize them in a hierarchical tree. Thus, they can potentially operate faster when there is long carry chain. A further optimization is provided from the observation that dual rail encoding logic can benefit from settling of either the 0 or 1 path. Dual-rail logic need not wait for both paths to be evaluated. Thus, it is possible to further speed up the carry look-ahead circuitry to send carry-generate/carry-kill signals to any level in the tree. This is elaborated in [8] and referred as DICLA with speedup circuitry (DICLASP).

III. PROPOSED SYSTEM

3.1 DESIGN OF PASTA

The architecture and theory behind PASTA is presented. The adder first accepts two input operands to perform half additions for each bit. Subsequently, it iterates using earlier generated carry and sums to perform half-additions repeatedly until all carry bits are consumed and settled at zero level.

A. Architecture of PASTA

The general architecture of the adder is shown in Fig. 1. The selection input for two-input multiplexers corresponds to the Req handshake signal and will be a single 0 to 1 transition denoted by SEL. It will initially select the actual operands during SEL = 0 and will switch to feedback/carry paths for subsequent iterations using SEL = 1. The feedback path from the HAs enables the multiple iterations to continue until the completion when all carry signals will assume zero values.

B. State Diagrams

In Fig. 2, two state diagrams are drawn for the initial phase and the iterative phase of the proposed architecture. Each state is represented by (*Ci*+1 *Si*) pair where *Ci*+1, *Si* represent carry out and sum values, respectively, from the i^{th} bit adder block. During the initial phase, the circuit merely works as a combinational HA operating in fundamental mode. It is apparent that due to the use of HAs instead of FAs, state *(*11*)* cannot appear.

During the iterative phase (SEL = 1), the feedback path through multiplexer block is activated. The carry transitions (*Ci*) are allowed as many times as needed to complete the recursion. From the definition of fundamental mode circuits, the present design cannot be considered as a fundamental mode circuit as the input–outputs will go through several transitions before producing the final output. It is not a Muller circuit working outside the fundamental mode either as internally; several transitions will take place, as shown in the state diagram. This is analogous to cyclic sequential circuits where gate delays are utilized to separate individual states.

C. Recursive Formula for Binary Addition

Theorem 1: The recursive formulation of (1)–(4) will produce correct sum for any number of bits and will terminate within a finite time. *Proof:* We prove the correctness of the algorithm by induction on the required number of iterations for completing the addition (meeting the terminating condition). *Basis:* Consider the operand choices for which no carry propagation is required, i.e., $C0\ i = 0$ for $\forall i,\ i \in [0..n]$. The proposed formulation will produce the correct result by a single-bit computation time and terminate instantly as (4) is met. Thus, all the single-bit adders will successfully kill or propagate the carries until all carries are zero fulfilling the terminating condition. The mathematical form presented above is valid under the condition that the iterations progress synchronously for all bit levels and the required input and outputs for a specific iteration will also be in synchrony with the progress of one iteration. In the next section, we present an implementation of the proposed architecture which is subsequently verified using simulations.

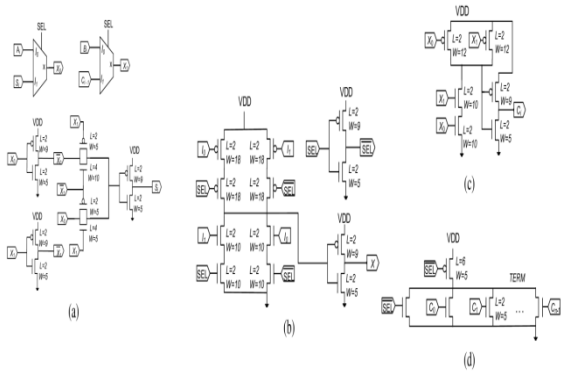

Fig. 3. CMOS implementation of PASTA. (a) Single-bit sum module. (b) 2×1 MUX for the 1 bit adder. (c) Single-bit carry module. (d) Completion signal detection circuit.

3.2 Implementation

A CMOS implementation for the recursive circuit is shown in Fig. 3. For multiplexers and AND gates we have used TSMC library implementations while for the XOR gate we have used the faster ten transistor implementation based on transmission gate XOR to match the delay with AND gates [4]. The completion detection following (4) is negated to obtain an active high completion signal (TERM). This requires a large fan-in *n*-input NOR gate. Therefore, an alternative more practical pseudo-nMOS ratio-ed design is used. The resulting design is shown in Fig. 3(d). Using the pseudo-nMOS design, the completion unit avoids the high fan-in problem as all the connections are parallel. The pMOS transistor connected to *V*DD of these ratio-ed design acts as a load register, resulting in static current drain when some of the nMOS transistors are on simultaneously. In addition to the *Ci* s, the negative of SEL signal is also included for the TERM signal to ensure that the completion cannot be accidentally turned on during the initial selection phase of the actual inputs. It also prevents the pMOS pull up transistor from being always on. Hence, static current will only be flowing for the duration of the actual computation. VLSI layout has also been performed [Fig. 3(e)] for a standard cell environment using two metal layers. The layout occupies 270 λ × 130 λ for 1-bit resulting in 1.123 Mλ2 area for 32-bit. The pull down transistors of the completion detection logic are included in the single-bit layout (the T terminal) while the pull-up transistor is additionally placed for the full 32-bit adder. It is nearly double the area required for RCA and is a little less than the most of the area efficient prefix tree adder, i.e., Brent–Kung adder (BKA).

IV. SIMULATION RESULTS

In this section, we present simulation results for different adders using Tanner EDA Tools version 14.3, running on 32-bit WINDOWS platform. For implementation of other adders, we have used standard library implementations of the basic gates. The custom adders such as DIRCA/DICLASP are implemented based on their most efficient designs.

Initially, we show how the present design of PASTA can effectively perform binary addition for different temperatures and process corners to validate the robustness under manufacturing and operational variations. In Fig. 4, the timing diagrams for worst and average cases corresponding to maximum and average length carry chain propagation over random input values are highlighted. The carry propagates through successive bit adders like a pulse as evident from Fig. 4(a). The best-case corresponding to minimum length carry chain (not shown here) does not involve any carry propagation, and hence incurs only a single-bit adder delay before producing the TERM signal. The worst-case involves maximum carry propagation cascaded delay due to the carry chain length of full 32 bit.

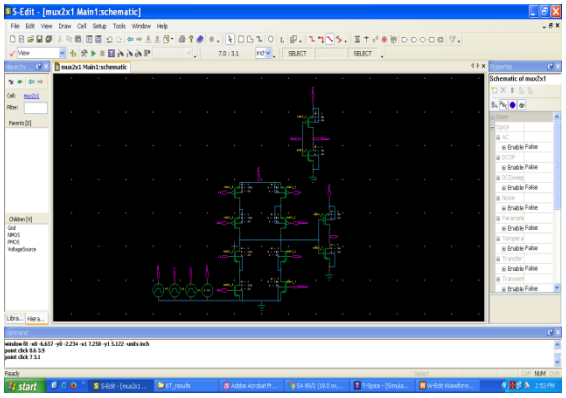

Fig 4: Mux 2 x 1 schematic

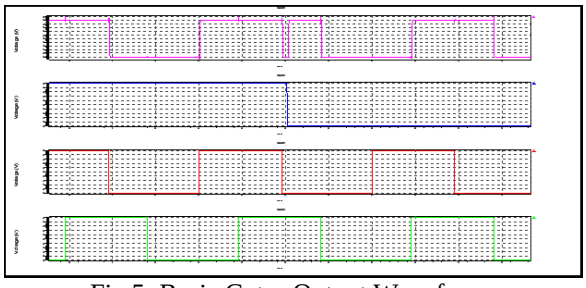

Fig 5: Basic Gates Output Waveforms

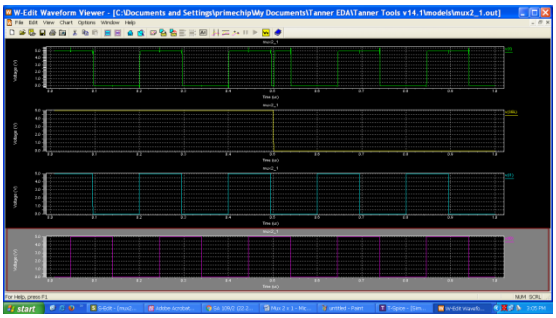

Fig 6 Basic Gates Output Waveforms

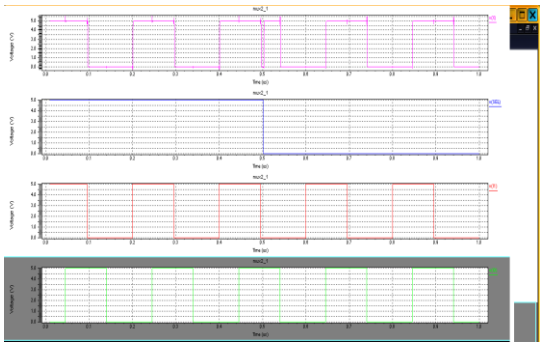

Fig 7: Basic Gates Output Waveforms

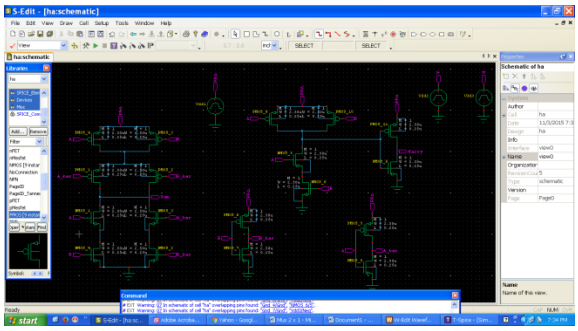

Fig 8: Half adder Schematic

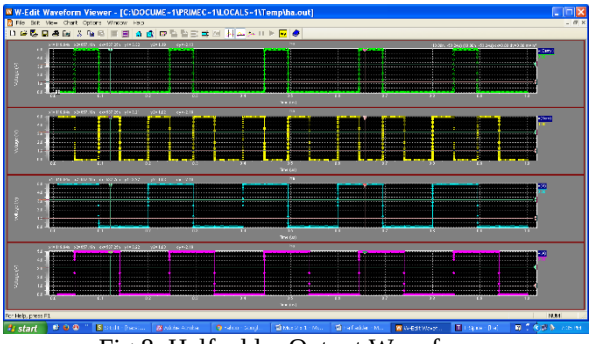

Fig 8: Half adder Output Waveforms

V. CONCLUSION

An efficient implementation of PASTA. Initially, the theoretical foundation for a single-rail wave-pipelined adder is established. Subsequently, the architectural design and CMOS implementations are presented. The design achieves a very simple *n*-bit adder that is area and interconnection-wise equivalent to the simplest adder namely the RCA. Moreover, the circuit works in a parallel manner for independent carry chains, and thus achieves logarithmic average time performance over random input values. The completion detection unit for the proposed adder is also practical and efficient. Simulation results are used to verify the advantages of the proposed approach.

REFERENCES

[1]. S.H. Unger, "Double-edge-triggered flip-flops," *IEEE Trans. on Computers*, vol. 30, no. 6,pp. 447-451, June 1981.

[2]. M. Pedram, Q. Wu and X. Wu, "A new design of double edge triggered flip-flops," *Proc. Asia and South Pacific Design Automation Conference*, pp. 417-421, 1998.

[3]. W. Chung, T. Lo and M. Sachdev, "A comparative analysis of low-power low-voltage dual-edge-triggered flip-flops," *IEEE Trans. on VLSI Systems*, vol. 10, no. 6, pp. 913-918, December 2002.

[4]. A.L. Fisher and H.T. Kung, "Synchronizing large VLSI processor arrays," *IEEE Trans. On Computers*, vol. C-34, no. 8, pp. 734-740, August 1985.

[5]. H.B. Bakoglu, J.T. Walker and J.D. Meindl, "A symmetric clock-distribution tree and optimized high-speed interconnections for reduced clock skew in ULSI and WSI circuit," *Proc. IEEE International Conf. on Computer Design*, pp. 118-122, 1986.

[6]. E.G. Friedman and S. Powell, "Design and analysis of a hierarchical clock distribution system for synchronous standard cell/macrocell VLSI," *IEEE Journal of Solid-State Circuits*, vol. SC-21, no. 2, pp. 240-246, April 1986.

[7]. S.Y. Kung and R.J. Gal-Ezer, "Synchronous versus asynchronous computation in VLSI array processors," *Proc. of the SPIE*, vol. 341, pp. 53-65, 1982.

[8]. E.G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proc. of the IEEE*, vol. 89, no. 5, pp. 665-692, May 2001.

[9]. G.E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, April 1965.

[10]. G.D. Wilk, R.M. Wallace and J.M. Anthony, "High-κ gate dielectrics: current status and materials properties considerations," *Journal of Applied Physics*, vol. 89, no. 10, pp. 5243-5275, 2001.

[11]. S.R. Nassif, "Design for variability in DSM technologies," *Proc. International Symp. On Quality Electronic Design*, pp. 451-454, 2000.

[12]. K. Bernstein, D.J. Frank, A.E. Gattiker, W. Haensch, B.L. Ji, S.R. Nassif, E.J. Nowak, D.J Pearson and N.J. Rohrer, "High-performance CMOS variability in the 65-nm regime and beyond," *IBM Journal of Research and Development*, vol. 50, no. 4/5, pp. 433-450, 2006.

[13]. D. Singh, "Prospects for low power microprocessor design," *Keynote Address, Proc. International Workshop on Low Power Design*, October 1994.

[14]. D. Brooks, V. Tiwari and M. Martonosi, "Wattch: a framework for architectural level power analysis and optimizations," *Proc. International Symp. on Computer Architecture*, pp. 83-94, 2000.

[15]. A.J. Martin, S.M. Burns, T.K. Lee, D. Borkovic and P.J. Hazewindus, "The first asynchronous microprocessor: the test results," *ACM SIGARCH Computer Architecture News*, vol. 17, no. 4, pp. 95-98, June 1989.

[16]. S.B. Furber, P. Day, J.D. Garside, N.C. Paver and J.V. Woods, "AMULET1: a micropipelined ARM," *Digest of Papers, IEEE Comp. Society International Conference (COMPCON)*, pp. 476-485, 1994.

[17]. T. Nanya, Y. Ueno, H. Kagotani, M. Kuwako and A. Takamura, "TITAC: design of a quasi-delay-insensitive microprocessor," *IEEE Design and Test of Computers*, vol. 11, no. 2, pp. 50-63, April 1994.

[18]. A.J. Martin, A. Lines, R. Manohar, M. Nystrom, P. Penzes, R. Southworth and U. Cummings, "The design of an asynchronous MIPS R3000 processor," *Proc. 17th Conf. on Advanced Research in VLSI*, pp. 164-181, 1997.

[19]. T. Nanya, A. Takamura, M. Kuwako, M. Imai, T. Fujii, M. Ozawa, I. Fukasaku, Y. Ueno, F. Okamoto, H. Fujimoto, O. Fujita, M. Yamashina and M. Fukuma, "TITAC-2: a 32-bit scalable-delay-insensitive microprocessor," *Proc. HOT Chips IX*, pp. 19-32, 1997.

[20]. H. van Gageldonk, K. van Berkel, Ad Peeters, D. Baumann, D. Gloor and G. Stegmann, "An asynchronous low power 80C51 microcontroller," *Proc. 4th International Symp. On Advanced Research in Asynchronous Circuits and Systems*, pp. 96-107, 1998.

[21]. M. Renaudin, P. Vivet and F. Robin, "ASPRO-216: a standard-cell QDI 16-bit RISC asynchronous microprocessor," *Proc. International Symp. on Advanced Research in Asynchronous Circuits and Systems*, pp. 22-31, 1998.

[22]. S.B. Furber, J.D. Garside, P. Riocreux, S. Temple, P. Day, J. Liu and N.C. Paver, "AMULET2e: an asynchronous embedded controller," *Proc. of the IEEE*, vol. 87, no. 2, pp. 243-256, February 1999.

[23]. S. Rotem, K. Stevens, R. Ginosar, P. Beerel, C. Myers, K. Yun, R. Kol, C. Dike, M. Roncken and B. Agapiev, "RAPPID: an asynchronous instruction length decoder," *Proc. International Symp. on Advanced Research in Asynchronous Circuits and Systems*, pp. 60-70, 1999.

[24]. S.B. Furber, D.A. Edwards and J.D. Garside, "AMULET3: a 100 MIPS asynchronous embedded processor," *Proc. International Conf. on Computer Design*, pp. 329-334, 2000.

[25]. J.D. Garside, W.J. Bainbridge, A. Bardsley, D.M. Clark, D.A. Edwards, S.B. Furber, J. Liu, D.W. Lloyd, S. Mohammadi, J.S. Pepper, O. Petlin, S. Temple and J.V. Woods, "AMULET3i – an asynchronous system-on-chip," *Proc. International Symp. on Advanced Research in Asynchronous Circuits and Systems*, pp. 162-175, 2000.

[26]. A. Semenov, A.M. Koelmans, L. Lloyd and A. Yakovlev, "Designing an asynchronous processor using Petri nets," *IEEE Micro*, vol. 17, no. 2, pp. 54-64, March-April 1997.

[27]. M. Lewis and L. Brackenbury, "CADRE: an asynchronous embedded DSP for mobile phone applications," *Design Automation for Embedded Systems*, vol. 6, pp. 451-475, 2002.

[28]. A.J. Martin, M. Nystrom, K. Papadantonakis, P.I. Penzes, P. Prakash, C.G. Wong, K.S. Ko, B. Lee, E. Ou, J. Pugh, E.-V. Talvala, J.T. Tong and A. Tura, "The Lutonium: a subnanojoule asynchronous 8051 microcontroller," *Proc. 9th International Symp. On Asynchronous Circuits and Systems*, pp. 14-23, 2003.
[29]. L.A. Plana, P.A. Riocreux, W.J. Bainbridge, A. Bardsley, S. Temple, J.D. Garside and Z.C. Yu, "SPA – a secure Amulet core for smartcard applications," *Microprocessors and Microsystems*, vol. 27, pp. 431-446, 2003.
[30]. D.E. Muller, "Asynchronous logics and application to information processing," in H. Aiken and W.F. Main (Eds.), *Switching Theory in Space Technology*, Stanford University Press, 1963.
[31]. R.E. Miller, *Switching Theory Volume 2: Sequential Circuits and Machines*, John Wiley & Sons, New York, 1965.
[32]. S.H. Unger, *Asynchronous Sequential Switching Circuits*, Wiley-Interscience, New York, 1969.
[33]. Semiconductor Industry Association's International Technology Roadmap for Semiconductors reports, Available: http://www.itrs.net/reports.html
[34]. G.F. Bouesse, G. Sicard, A. Baixas and M. Renaudin, "Quasi delay insensitive asynchronous circuits for low EMI," *Proc. 4th International Workshop on Electromagnetic Compatibility of Integrated Circuits*, pp. 27-31, 2004.
[35]. K. van Berkel, R. Burgess, J. Kessels, Ad Peeters, M. Roncken, and F. Schalij, "A fully
[36]. asynchronous low-power error corrector for the DCC player," *Proc. 41st IEEE International Solid State Circuits Conference*, pp. 88-89, 1994.
[37]. K. van Berkel, M.B. Josephs and S.M. Nowick, "Scanning the technology: applications of asynchronous circuits," *Proc. of the IEEE*, vol. 87, no. 2, pp. 223-233, February 1999.
[38]. N. Karaki, "Asynchronous design: an enabler for flexible microelectronics," Invited Talk, *Proc. 12th IEEE International Symp. on Asynchronous Circuits and Systems*, pp. xii, 2006.
[39]. S. Hauck, "Asynchronous design methodologies: an overview," *Proc. of the IEEE*, vol. 83, no. 1, pp. 69-93, January 1995.
[40]. Al Davis and S.M. Nowick, "An introduction to asynchronous circuit design," *Technical Report* UUCS-97-013, Computer Science Dept., University of Utah, September 1997.
[41]. J. Sparso and S.B. Furber (Eds.), *Principles of Asynchronous Circuit Design: A Systems Perspective*, Kluwer Academic Publishers, 2001.
[42]. I.E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720- 738, June 1989.
[43]. S.B. Furber and P. Woods, "Four-phase micropipeline latch control circuits," *IEEE Trans. on VLSI Systems*, vol. 4, no. 2, pp. 247-253, June 1996.
[44]. B. Bose, "On unordered codes," *IEEE Trans. on Computers*, vol. 40, no. 2, pp. 125-131, February 1991.
[45]. T. Verhoeff, "Delay-insensitive codes: an overview," *Distributed Computing*, vol. 3, no. 1, pp. 1-8, 1988.
[46]. D.W. Lloyd and J.D. Garside, "A practical comparison of asynchronous design styles," *Proc. 7th International Symp. on Asynchronous Circuits and Systems*, pp. 36-45, 2001.
[47]. V. Akella, N.H. Vaidya and G.R. Redinbo, "Limitations of VLSI implementation of delayinsensitive codes," *Proc. 26th Annual International Symp. on Fault-Tolerant Computing*,
[48]. pp. 208-217, 1996.
[49]. S.J. Piestrak, "Membership test logic for delay-insensitive codes," *Proc. International Symp. on Asynchronous Circuits and Systems*, pp. 194-204, 1998.
[50]. C.V. Freiman, "Optimal error detection codes for completely asymmetric binary channels," *Information Control*, vol. 5, pp. 64-71, March 1962.
[51]. S.B. Furber, A. Efthymiou and M. Singh, "A power-efficient duplex communication system," *Proc. Asynchronous INTerfaces: tools, techniques and implementations (AINT Workshop*, 2000.
[52]. W.J. Bainbridge, W.B. Toms, D.A. Edwards and S.B. Furber, "Delay-insensitive, point-to point interconnect using *m*-of-*n* codes," *Proc. 9th International Symp. on Asynchronous Circuits and Systems*, pp. 132-140, 2003.
[53]. S.J. Piestrak and T. Nanya, "Towards totally self-checking delay-insensitive systems," *Proc. 25th International Symp. on Fault-Tolerant Computing*, pp. 228-237, 1995.
[54]. S.M. Nowick and D.L. Dill, "Synthesis of asynchronous state machines using a local clock," *Proc. International Conf. on Computer Design*, pp. 192-197, 1991.
[55]. K. Yun, D. Dill and S.M. Nowick, "Synthesis of 3D asynchronous state machines," *Proc. International Conf. on Computer Design*, pp. 346-350, 1992.
[56]. Al Davis, B. Coates and K. Stevens, "The post office experience: designing a large asynchronous chip," *Proc. 26th Annual Hawaii International Conf. on Systems Sciences*, vol. 1, pp. 409-418, 1993.
[57]. J.A. Waicukauski, E. Lindbloom, B.K. Rosen and V.S. Iyengar, "Transition fault simulation," *IEEE Design and Test of Computers*, vol. 4, no. 2, April 1987.

[58]. G.L. Smith, "Model for delay faults based upon paths," *Proc. International Test Conference*, pp. 342-349, 1985.
[59]. E.J. McCluskey, *Logic Design Principles: with emphasis on testable semi-custom circuits*, Prentice-Hall, Englewood Cliffs, NJ, 1986.
[60]. J.A. Brzozowski and J.C. Ebergen, "Recent developments in the design of asynchronous circuits," *Technical Report* CS-89-18, Dept. of Computer Science, University of Waterloo, 1989.
[61]. W.A. Clark, "Macromodular computer systems," *AFIPS Proc. Spring Joint Computer Conference*, 1967.
[62]. W.A. Clark, and C.E. Molnar, "Macromodular computer systems," *Computers in Biomedical Research*, B.D. Waxman and R. Stacey (Eds.), pp. 45–85, vol. IV, 1974, Academic Press, NY.
[63]. J.T. Udding, "A formal model for defining and classifying delay-insensitive circuits and systems," *Distributed Computing*, vol. 1, no. 4, pp. 197-204, 1986.
[64]. A.J. Martin, "Compiling communicating processes into delay-insensitive VLSI circuits," *Distributed Computing*, vol. 1, no. 4, pp. 226-234, December 1986.
[65]. A.J. Martin, "The limitation to delay-insensitivity in asynchronous circuits," *Proc. 6th MIT Conf. on Advanced Research in VLSI*, pp. 263-278, MIT Press, 1990.
[66]. J.A. Brzozowski and Jo C. Ebergen, "On the delay-sensitivity of gate networks," *IEEE Trans. on Computers*, vol. 41, no. 11, pp. 1349-1360, November 1992.
[67]. J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory Languages and Computation*, 1st Edition, Addison-Wesley Publishing Company, 1979.
[68]. K. van Berkel, "Beware the isochronic fork," *Integration, the VLSI Journal*, vol. 13, no. 2, pp. 103-128, June 1992.