International Journal of Trend in Research and Development, Volume 2(2), ISSN: 2394-9333
www.ijtrd.com

Self-Repetative Approach to the Design of a Parallel Self-Timed Adder

Reshma K. P.
PG Student,
Department of ECE,
Thanthai Periyar Government Institute of Technology
Vellore, India.

Vijaya J
Assitant Professor,
Department of ECE,
Thanthai Periyar Government Institute of Technology
Vellore,India.

***Abstract*—This 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 fan-out. 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 an industry standard toolkit that verifies 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. 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. However, this usually comes at the expense of a larger 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 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 behavior.

II. RELATED WORKS

Based on the research undertaken on self-timed combinational logic realization and especially with respect to data path elements, the original contributions of this thesis are summarized as follows. Formulation of speed-independent decomposition rules using set-theoretic principles. General multi-level synthesis models to realize strong or weak-indication combinational logic, which consider the entire input space.

A set theory based heuristic for compactly synthesizing combinational logic of arbitrary size as self-timed circuits and a system configuration in support of the proposed heuristic. Design of self-timed carry-ripple adders which feature local or global indication property and proposition of the concept of logic redundancy insertion for delay reduction.

Self-timed section carry based carry-look ahead architectures that greatly minimize the latency of dual-operand addition in comparison with the ripple carry topology. A combinational bit-partitioning strategy addressing self-timed multi-operand addition and the design of a self-timed logic compressor. Integer addition is one of the most important operations in digital computer systems. In addition to explicit arithmetic (such as addition, subtraction, multiplication, and division) performed in a program, additions are performed to increment program counters and calculate effective addresses. Statistics presented that, in a prototypical RISC machine (DLX), 72 percent of the instructions perform additions (or subtractions) in the data path.

The statistics reported in ARM processors even reaches 80 percent. Thus, the performance of processors is significantly influenced by the speed of their adders. Circuits may be classified as synchronous or asynchronous. Synchronous circuits have a clock to synchronize the operations of subsystems, while asynchronous circuits do not. Subsystems in asynchronous circuits usually need start and completion

mechanisms to synchronize with one another. One advantage of using asynchronous circuits is that these circuits operate at average rates, while synchronous circuits are required to operate at the worst rates. A good example for this is that n-bit ripple-carry adders have worst case computation time whereas n-bit carry-completion sensing adders have average computation time. With successful nanometer scale IC designs getting rolled out from semiconductor in huge volumes every year, and with the continuous venture intodeeper nano scale device geometries, the semiconductor industry is contemplating several options to push the limits of conventional digital IC design in terms of devices, dielectric materials, interconnects, foundry processes, fabrication methods, testing techniques, and manufacturing and packaging technologies.

Given the aggressive technological trend fuelled by an ever-increasing market demand for mobile and portable electronic products, the Semiconductor Industry Association‟s 2011 International Technology Roadmap on Semiconductors (ITRS) report has identified „design for reliability‟ as one of the long-term grand challenges. Indeed, taking cognizance of decreasing feature sizes and associated increases in variability of devices, the ITRS report mentions that the issue of „reliability‟ could assume comparable significance with quality-of-results in the nanometer regime.

In this context, the self-timed design paradigm is pegged to be a strong contender and a viable alternative to mainstream synchronous design style for implementing digital logic functionality such as arithmetic and logic units, circuits used in telecommunications, defense and security applications, and subsystems deployed in a wide range of industrial and consumer electronics. The primary motivation for adopting the self-timed design style arises from the fact that self-timed circuits consume power only when and where active, absorb process, temperature and parametric variations with ease, feature greater modularity, and inherently possess good noise and EMI tolerance capabilities.

Dynamic circuit techniques offer potential advantages over static CMOS. Domino circuits are the most widespread representative in high performance designs but suffer increasingly from deep submicron effects. This paper presents evaluations in terms of area, power dissipation, and propagation delay for static CMOS as well as for several Domino derivatives in a 90 nm technology. Finally, issues of reliability gained from practical experience for different test benches are discussed.

Integer addition is one of the most important operations in digital computer systems because the performance of processors is significantly influenced by the speed of their adders. This paper proposes a self-timed carry-look ahead adder in which the logic complexity is a linear function of n, the number of inputs, and the average computation time is proportional to the logarithm of the logarithm of n. To the best of our knowledge, our adder has the best area-time efficiency. An economic implementation of this adder in CMOS technology is also presented. SPICE simulation results show that, based on random inputs, our 32-bit self-timed carry-look ahead adder is 2.39 and 1.42 times faster than its synchronous counterpart and self-timed ripple-carry adder, respectively, and, based on statistical data gathered from a 32-bit ARM simulator, it is 1.99 and 1.83 times faster than its synchronous counterpart and self-timed ripple-carry adder, respectively

III. METHODOLOGY

There are 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

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

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, which uses full adder (FA) functional blocks adapted for dual-rail carry. On the other hand, a speculative completion adder is proposed. 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.

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 presented .While the conventional CMOS RCA uses 28 transistors.

Similar to CLA, the DICLA defines carry propagate, generate, and kill equations in terms of dual-rail encoding. 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.

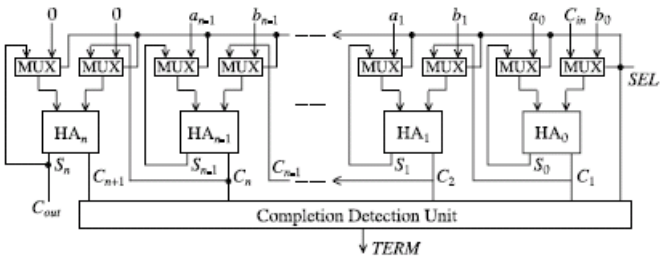

Fig.1. General Block Diagram of PASTA.

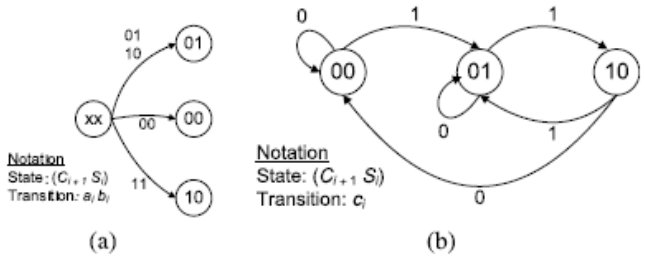

Fig.2. State diagrams for PASTA. (a) Initial phase. (b) Iterative phase.

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 and referred as DICLA with speedup circuitry (DICLASP).

C. *Architecture Of Pasta*

The general architecture of the adder is shown on above diagram. 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.

D. *State Diagrams*

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 represents carry out and sum values, respectively, from the ith 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.

E. *Recursive Formula For Binary Addition*

Let S ji and C ji+1 denote the sum and carry, respectively, for ith bit at the j th iteration. The initial condition (j = 0) for addition is formulated as follows:

The j th iteration for the recursive addition is formulated by

The recursion is terminated at kth iteration when the following

Condition is met:

Now, the correctness of the recursive formulation is inductively proved as follows.

Theorem 1: The recursive formulation 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., The proposed formulation will produce the correct result by a single-bit computation time and terminate instantly as met.

Induction: Assume that Cki+1_= 0 for some ith bit at kth iteration. Let l be such a bit for which Ck l+1 = 1. We show that it will be successfully transmitted to next higher bit in the (k + 1)th iteration.

As shown in the state diagram, the kth iteration of Ith bit state (Ckl+1, Skl) and (l + 1)th bit state (Ck l+2, Sk l+1) could be in any of (0, 0), (0, 1), or (1, 0) states. As Ck l+1 = 1, it implies that Skl = 0. Hence, Ck+1 l+1 = 0 for any input condition between 0 to l bits.

We now consider the (l + 1)th bit state (Ckl+2, Skl+1) for kth iteration. It could also be in any of (0, 0), (0, 1), or (1, 0) states. In (k+1)th iteration, the (0, 0) and (1, 0) states from the kth iteration will correctly produce output. For (0, 1) state, the carry successfully propagates through this bit level following.

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.

F. CMOS

A CMOS implementation for the recursive circuit is shown on below diagram. 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.

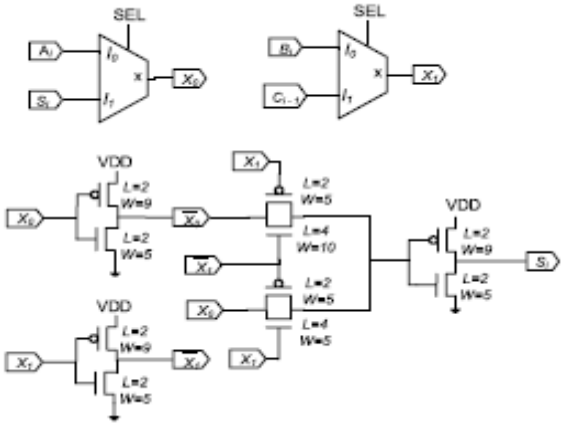

Fig 3. Single-bit sum module

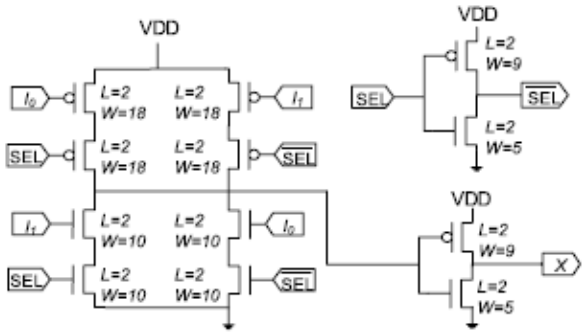

Fig 4. 2×1 MUX for the 1 bit adder

The completion detection following 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 ratioed design is used. Using the pseudo-nMOS design, the completion unit avoids the high fan-in problem as all the connections are parallel.

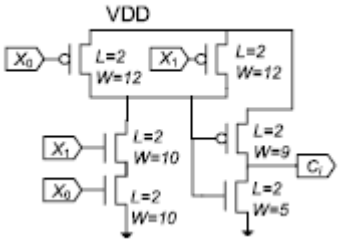

Fig 5. Single-bit carry module

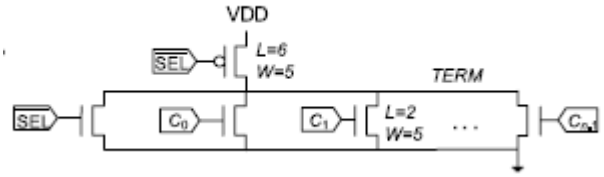

Fig 6. Completion signal detection circuit

The pMOS transistor connected to VDD of this ratioed 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 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.

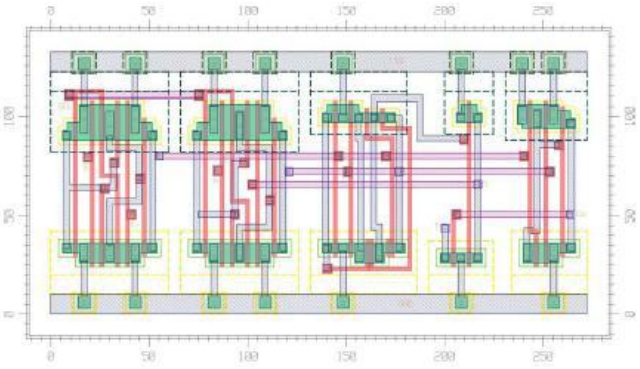

Fig .7 VLSI layout

VLSI layout has also been performed for a standard cell environment using two metal layers. The layout occupies 270 _ × 130 _ for 1-bit resulting in 1.123 M_2 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., the 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. The best-case corresponding to minimum length carry chain 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. This circuit works correctly for all process corners. This has no effects in the circuit and errors induced by the SF extreme corner case.

The delay performances of different adders we have used 1000 uniformly distributed random operands to represent the average case while best case, worst case correspond to specific test-cases representing zero, 32-bit carry propagation chains respectively. The delay for combinational adders is measured at 70% transition point for the result bit that experiences the

maximum delay. For self-timed adders, it is measured by the delay between SEL and TERM signals, as depicted. The combinational adders, such as RCA/B-CLA/BKA/ Kogge–Stone adder (KSA)/Sklansky‟s conditional sum adder (SCSA) can only work for the worst-case delay as they do not have any completion sensing mechanism.

Therefore, these results give an empirical upper bound of the performance enhancement that can be achieved using these adders as the basic unit and employing some kind of completion sensing technique In the worst case, KSA performs best as they (along with SCSA) have the minimal tree-depth. On the other hand, PASTA performs best among the self-timed adders. PASTA performance is comparable with the best case performances of conventional adders. Effectively, it varies between one and four times that of the best adder performances. Therefore, the best case delay represents the delay required to generate the TERM signal only and of the order of picoseconds.

Similar overhead is also present in dual-rail logic circuits where they have to be reset to the invalid state prior to any computation. The dynamic/nMOS only designs require a precharge phase to be completed during this interval. These overheads are not included in this comparison. The best case and worst case carry performances of DIRCA for the chosen operands are nearly the same, as one rail needs to be set from start to end? In contrast, the average cases can have carry generation and killing in any bit and thus providing a better case for DIRCA.

Another interesting observation is that the performances of the combinational adders and PASTA improve with the decreasing process width and VDD values while the performance of dual-rail adders decreases with scaling down of the technology. This results from the fact that dynamic logic requires technology specific energy delay optimization as performed. We also note that the dynamic logic switching speed advantage can be attributed to the nMOS threshold voltage being lower than a static CMOS threshold voltage (VDD/2), which diminishes with decreasing process width. The PASTA layout complies with all design rules for the TSMC 0.35 µm process and this was found to increase the delay by two to three times after taking into consideration layout specific parasitic capacitances.

IV. Result and Discussion

In this paper, we present simulation results for different adders using Modelsim Altera running on 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.

A. Synthesis Report Of Parallel Self-Timed Adder

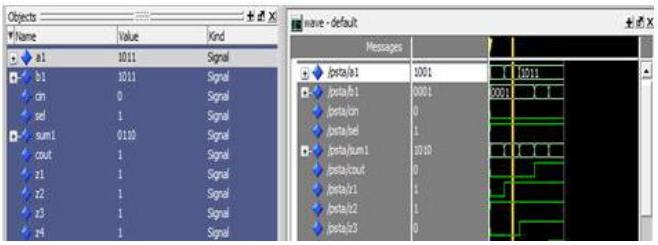

Fig: 8. synthesis of parallel self-timed adder

B. Schematic View Of Pasta

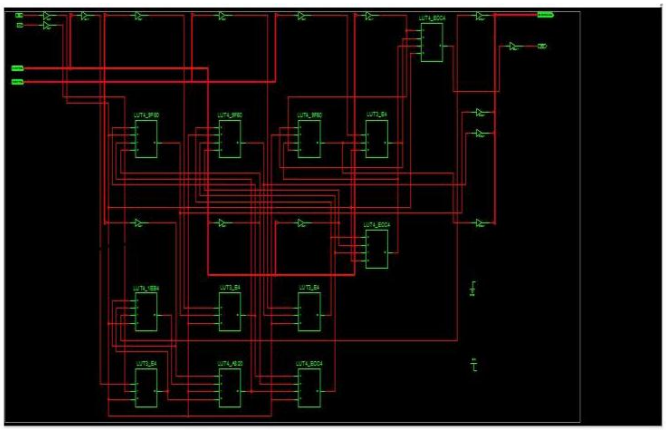

Fig: 9 View of technology schematic

Conclusion

This brief presents 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] D. Geer, “Is it time for clock less chips? [Asynchronous processor chips],” IEEE Comput., vol. 38, no. 3, pp. 18–19, Mar. 2005.
[2] J. Sparsø and S. Furber, Principles of Asynchronous Circuit Design. Boston, MA, USA: Kluwer Academic, 2001.
[3] P. Choudhury, S. Sahoo, and M. Chakraborty, “Implementation of basic arithmetic operations using cellular automaton,” in Proc. ICIT, 2008, pp. 79–80.
[4] M. Z. Rahman and L. Kleeman, “A delay matched approach for the design of asynchronous sequential circuits,” Dept. Comput. Syst. Technol., Univ. Malaya, Kuala Lumpur, Malaysia, Tech. Rep. 05042013, 2013.
[5] M. D. Riedel, “Cyclic combinational circuits,” Ph.D. dissertation, Dept. Comput. Sci., California Inst. Technol., Pasadena, CA, USA, May 2004.
[6] R. F. Tinder, Asynchronous Sequential Machine Design and Analysis: A Comprehensive Development of the Design and Analysis of Clock-Independent State Machines and Systems. San Mateo, CA, USA: Morgan, 2009.

[7] W. Liu, C. T. Gray, D. Fan, and W. J. Farlow, "A 250-MHz wave pipelined adder in 2-μm CMOS," IEEE J. Solid-State Circuits, vol. 29, no. 9, pp. 1117–1128, Sep. 1994.
[8] F.-C. Cheng, S. H. Unger, and M. Theobald, "Self-timed carry-lookahead adders," IEEE Trans. Comput., vol. 49, no. 7, pp. 659–672, Jul. 2000.
[9] S. Nowick, "Design of a low-latency asynchronous adder using speculative completion," IEE Proc. Comput. Digital Tech., vol. 143, no. 5, pp. 301–307, Sep. 1996.