Journal of Network Communications and Emerging Technologies (JNCET) www.jncet.org
Volume 6, Issue 4, April (2016)

A Low Power Single-Rail Pipelined Adder Based on Partial Element Reuse

Anu Priyanka R
PG Scholar,Nandha Engineering college ,Erode,India.

Prabakaran G
Assistant professor,Nandha Engineering college ,Erode,India.

Abstract – 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 circuits. This brief presents a parallel single-rail self-timed adder. It is based on a recursive formulation for performing multi bit 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 outs. 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.

Index Terms – Asynchronous circuits, binary adders, CMOS design, digital arithmetic.

1. INTRODUCTION

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 clockless/ 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 singlerail wave pipelined approach and quite different from conventional pipelined adders using dual-rail encoding to implicitly represent the pipelining of carry signals.

The remainder of this brief is organized as follows. Section II provides a review of self-timed adders. Section III presents the architecture and theory behind the proposed adder. Sections IV and V provide CMOS implementation and simulation results for the proposed adder. Section VI draws the conclusion.

2. 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 presented in [8] while the conventional CMOS RCA uses 28 transistors.

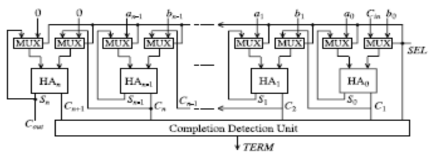


Fig. 1. General block diagram of PASTA.

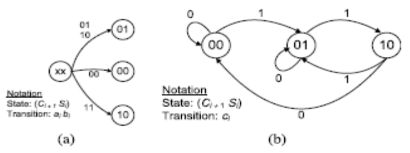


Fig. 2. State diagrams for PASTA. (a) Initial phase. (b) Iterative phase.

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).

3. DESIGN OF PASTA

In this section, 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 [4].

C. Recursive Formula for Binary Addition

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

$$S^0 i = ai \oplus bi$$

$$C^0 i+1 = aibi \,. \qquad (1)$$

The *j* th iteration for the recursive addition is formulated by

$$S^j i = Si^{j-1} \oplus Ci^{j-1}\,, \quad 0 \le i < n \qquad (2)$$

$$C^j i+1 = Si^{j-1}\, Ci^{j-1}\,, \quad 0 \le i \le n. \qquad (3)$$

The recursion is terminated at *k*th iteration when the following condition is met:

$$C^k n + C^k n-1 + \cdot \cdot \cdot + C^k 1 = 0, \quad 0 \le k \le n. \qquad (4)$$

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

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., $C^0\ I = 0$ 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.

Induction: Assume that $C^k\ i+1 \neq 0$ for some *i*th bit at *k*th iteration. Let *l* be such a bit for which $C^k\ l+1 = 1$. 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 *k*th iteration of *l*th bit state *(*$C^k\ l+1$, $S^k\ l$ *)* and *(l* + 1*)*th bit state *(*$C^k\ l+2$, $S^k\ l+1$*)* could be in any of *(*0, 0*)*, *(*0, 1*)*, or *(*1, 0*)* states. As $C^k\ l+1 = 1$, it implies that $S^k\ l = 0$. Hence, from (3), $C^{k+1}\ l+1 = 0$ for any input condition between 0 to *l* bits.

We now consider the *(l* + 1*)*th bit state *(*$C^k\ l+2$, $S^S\ l+1$*)* for *k*th 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 *k*th iteration will correctly produce output of *(*0, 1*)* following (2) and (3). For *(*0, 1*)* state, the carry successfully propagates through this bit level following (3).

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.

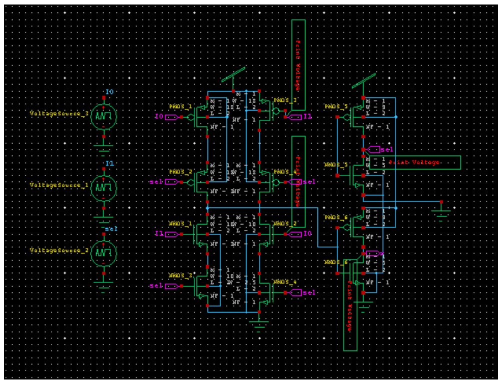


Fig.3 2X1 Multiplexer for the 1 bit adder

4. 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. 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. 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 this 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.

5. EXPREMENTAL RESULT

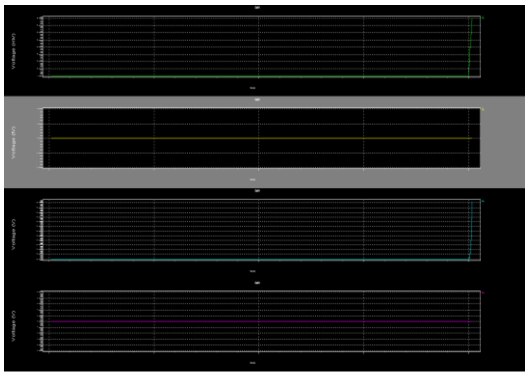
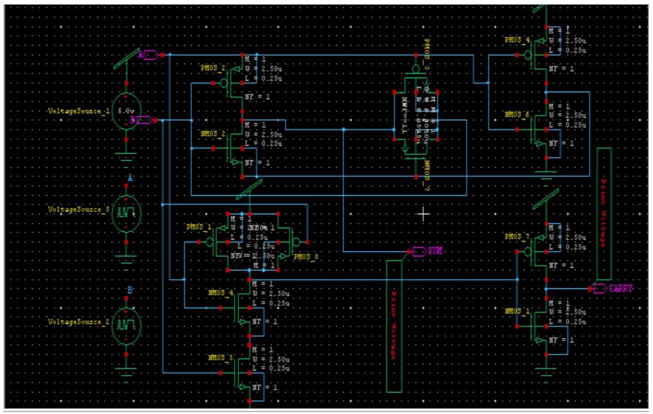


Fig4 2x1 Multiplexer for the 1 bit adder

Fig 3,4,5,6 shows the design and output analysis of CMOS implementation of PASTA.



F ig5 Half adder using transmission gate

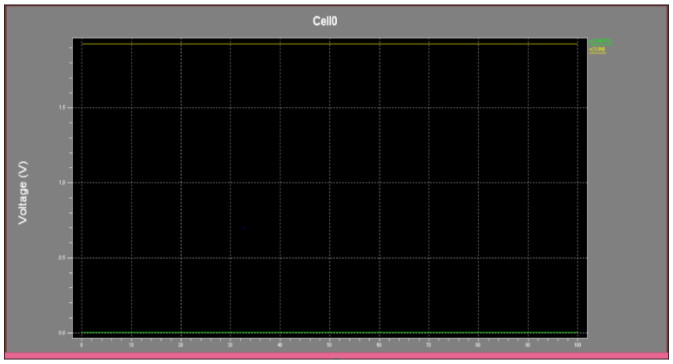


Fig6 Half adder using transmission gate

6. CONCLUSION

This brief presents an efficient implementation of a PASTA. Initially, the theoretical foundation for a single-rail wave pipelinedadder is established. Subsequently, the architectural design and CMOS implementations are spresented. 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 clockless 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 carrylookahead 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.
[10]N. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective. Reading, MA, USA: Addison-Wesley, 2005.
[11]C. Cornelius, S. Koppe, and D. Timmermann, “Dynamic circuit techniques in deep submicron technologies: Domino logic reconsidered,” in Proc. IEEE ICICDT, Feb. 2006, pp. 1–4.
[12]M. Anis, S. Member, M. Allam, and M. Elmasry, “Impact of technology scaling on CMOS logic styles,” IEEE Trans. Circuits Syst., Analog Digital Signal Process., vol. 49, no. 8, pp. 577–588, Aug. 2002.