Power Analysis of Parallel Self-timed Adder by Recursive method

J.Dhanasekar, S.Sarmila, V.Kavitha, S.Rathimeena
Assistant Professor, ECE dept, Sri Eshwar College of Engineering, Coimbatore
ME VLSI Design, Sri Eshwar College of Engineering, Coimbatore

ABSTRACT
This brief presents a parallel single-rail self-timed adder. It uses recursive method for performing multibit binary addition. This design attains good performance without any special speedup circuitry. 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. The recursive method based adder consumes least power among other Self-timed adders. 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 tanner tool and superiority of the proposed approach over existing asynchronous adders.

Keywords: Asynchronous circuit, CMOS design, digital arithmetic Binary adders, Recursive adder.

I. 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 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. On the other hand, wave pipelining (or max-imal 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.

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

acknowledgement handshaking protocol to establish a pipeline in the absence of clocks.

Explicit Handshaking blocks for small elements, such as bit adders, are expensive. A valid dual-rail carry output also provides acknowledgment from a single-bit adder block. 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 ef-icient alternative approach is worthy of consideration that can address these problems.

This brief presents a Parallel Asyn-chronous 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- us sequential adder [4]. Cyclic circuits can be more resource efficient than their acyclic counterparts [5], [6].

II. BACKGROUND
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 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.

2.1. 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
conven- tion is used for internal bitwise flow of carry outputs. These dual-rail signal 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 [7].

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.

2.2. 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. 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. 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.

3.1. 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 [9]. It will initially select the actual operands during SEL = 0 and will switch to feedback 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.

![](image)

**Fig. 1. General block diagram of PASTA.**

3.2. 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 \( C_{i+1}, S_i \) pair where \( C_{i+1}, S_i \) 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].
IV. CMOS IMPLEMENTATION OF PASTA

A CMOS implementation for the recursive circuit is shown in Fig. 3. For multiplexers and AND gates we have used implementations of 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. Using the pseudo-nMOS design, the completion unit avoids the high fan-in problem as all the connections are parallel. 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.

V. RECURSIVE METHOD

Let $S_i^j$, $C_i^{j+1}$ 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:

\[
S_0^0 = a_i \oplus b_i \\
C_0^{i+1} = a_ib_i.
\]  

(1)

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

\[
S_i^j = S_i^{j-1} \oplus C_i^{j-1}, \quad 0 \leq i < n \tag{2}
\]

\[
C_i^{j+1} = S_i^{j-1}C_i^{j-1}, \quad 0 \leq i \leq n. \tag{3}
\]
The recursion is terminated at $k$th iteration when the following condition is met

$$C_k^1 + C_{k-1}^2 + \cdots + C_1^n = 0, \quad 0 \leq k \leq n. \quad (4)$$

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.

**Basis:** Consider the operand choices for which no carry propagation is required, i.e., $C_i^0 = 0$ 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 [10].

**VI. SIMULATION RESULTS**

<table>
<thead>
<tr>
<th>PARAMETERS</th>
<th>4-BIT ADDER</th>
<th>8-BIT ADDER</th>
<th>16-BIT ADDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVERAGE POWER</td>
<td>$1.1157 \times 10^{-3}$ W</td>
<td>$2.2991 \times 10^{-3}$ W</td>
<td>$5.8240 \times 10^{-3}$ W</td>
</tr>
<tr>
<td>MAXIMUM POWER</td>
<td>$1.0702 \times 10^{-2}$ W at 2.000*10^{-5} secs</td>
<td>$1.8619 \times 10^{-2}$ W at 1.500*10^{-5} secs</td>
<td>$3.5777 \times 10^{-2}$ W at 1.500*10^{-5} secs</td>
</tr>
<tr>
<td>MINIMUM POWER</td>
<td>$1.3006 \times 10^{-8}$ W at 2.717*10^{-5} secs</td>
<td>$1.3906 \times 10^{-7}$ W at 7.1561*10^{-6} secs</td>
<td>$3.6648 \times 10^{-5}$ W at 1.000*10^{-5} secs</td>
</tr>
<tr>
<td>POWER DELAY PRODUCT</td>
<td>0.214 µws</td>
<td>0.279 µws</td>
<td>0.536 µws</td>
</tr>
<tr>
<td>ENERGY DELAY PRODUCT</td>
<td>4.2808pws</td>
<td>4.189pws</td>
<td>8.049pws</td>
</tr>
<tr>
<td>STATIC CURRENT</td>
<td>0.5945amps</td>
<td>1.0343amps</td>
<td>1.9876amps</td>
</tr>
</tbody>
</table>

Table 1. Comparison of parameters for 4-bit Adder, 8-bit Adder and 16-bit Adder.
In this section, we present simulation results for different adders using tanner tool version 13.0. 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 from [8]. 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.5 and Fig.6 shows the timing diagram of 16-bit Parallel Adder when initial phase: SEL=0 and iterative phase: SEL=1. Comparison of parameters such as power, delays for 4bit, 8-bit, 16-bit adders are shown in table 1.

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. It is even shown to be the fastest for 0.35 μm process [11]. For average cases, PASTA performance remains within two times to that of the best average case performances while for the worst case, it behaves similar to the RCA. Note that, PASTA completes the first iteration of the recursive formulation when “SEL = 0.” 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/e nMOS only designs require a precharge phase to be completed during this interval [12]. 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 the technology.

VII. 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