International Journal of Electrical and Electronics Research ISSN 2348-6988 (online)
Vol. 3, Issue 4, pp: (39-50), Month: October - December 2015, Available at: www.researchpublish.com

Implementation of Han-Carlson Adder for Error Tolerent Applications

Sasireka.S[1], Marimuthu C.N[2]

[1]PG Scholar, [2]Dean (R & D), Department of ECE, Nandha Engineering College, Erode, India

Abstract: **Variable latency adders have been recently proposed in literature. A Variable latency adder employs speculation: the exact arithmetic function is replaced with an approximated one that is faster and gives the correct result most of the time, but not always. The approximated adder is augmented with an error detection network that asserts an error signal when speculation fails. Speculative variable latency adders have attracted strong interest thanks to their capability to reduce average delay compared to traditional architectures. This paper proposes a novel variable latency speculative adder based on Han-Carlson parallel- prefix topology that resulted more effective than variable latency Kogge-Stone topology. The paper describes the stages in which variable latency speculative prefix adders can be subdivided and presents a novel error detection network that reduces error probability compared to previous approaches. Several variable latency speculative adders, for various operand lengths, using both Han-Carlson and Kogge-Stone topology, have been synthesized using the UMC 65 nm library. Obtained results show that proposed variable latency Han-Carlson adder outperforms both previously proposed speculative Kogge-Stone architectures and non-speculative adders, when high-speed is required. It is also shown that non-speculative adders remain the best choice when the speed constraint is relaxed.**

Keywords**: Addition, digital arithmetic, parallel-prefix adders, speculative adders, speculative functional units, variable latency adders.**

I. INTRODUCTION

Adders are basic functional units in computer arithmetic. Binary adders are used in microprocessor for addition and subtraction operations as well as for floating point multiplication and division. Therefore adders are fundamental multiplication and division. Therefore adders are fundamental components and improving their performance is one of the major challenges in digital designs. Theoretical research has established lower bounds on area and delay of -bit adders: the former varies linearly with adder size, the latter has an O(log2(n)) behavior.

High speed adders are based on well-established parallel-prefix architectures, including Brent-Kung, Kogge-Stone, Sklansky, Han-Carlson, Ladner-Fischer, and Knowles. These standard architectures operate with fixed latency. Better average performances can be achieved by using variable latency adders, that have been recently proposed in literature. A variable latency adder employs speculation: the exact arithmetic function is replaced with an approximated one that is faster and gives the correct result most of the time, but not always. The approximated adder is augmented with an error detection network that asserts an output signal when speculation fails. In this case (misprediction), another clock cycle is needed to obtain the correct result with the help of a correction stage. Since the addition time is one clock cycle when no error occurs and two clock cycles when the speculation fails, the average addition time Tavg can be computed as

$$T_{avg} = P_{Err} \cdot 2 \cdot T_{clk} + (1 - P_{Err}) \cdot T_{clk} = T_{clk}(1 + P_{Err}) \quad (1)$$

Where Tclk is the clock period and PErr is the error probability of the speculative adder.

Speculative adders are built upon the observation that the critical path is rarely activated in traditional adders. In particular, this adders each output depends on all previous bits, so the most significant output depends on all the input bits. Instead, in speculative adders each output depends only on the previous K bits, where K goes as O(log2(n)). This reflects the fact that a propagate chain longer than log2(n) is a very rare event.

A first speculative approach to addition was proposed by asynchronous contest, which implements a variable latency adder cutting the lowest levels of a Kogge-Stone adder. In synchronous contest, propose a variable latency speculative adder; here the speculative addition is realized in the same way as, cutting the lower levels of a Kogge-Stone adder. In a variable latency carry-select adder is introduced, where the adder is fragmented in various windows, each one containing a Kogge-Stone adder.

The Kogge-Stone adder is often used when speed is the primary concern, since it uses the minimum number of logic levels and each cell in the adder tree has fanout of 2. This comes at the cost of using many propagate-generate cells and many wires that must be routed between stages.

In this paper we propose a novel variable latency speculative adder based on Han-Carlson parallel-prefix topology. The Han-Carlson topology uses one more stage than Kogge- Stone adder, while requiring a reduced number of cells and simplified wiring. Thus, it can achieve similar speed performance compared to Kogge-Stone adder, at lower power consumption and area. We show that a speculative carry tree can be obtained by pruning some intermediate levels of the classical Han-Carlson topology. The paper presents a rigorous derivation of the error detection network and shows that the error detection network required in speculative Han-Carlson adders is significantly faster than the one used by speculative Kogge-Stone architecture. An extensive set of implementation results for 65 nm CMOS technology shows that proposed Han-Carlson variable latency adders outperform previously developed variable latency Kogge-Stone architectures. Compared with traditional, non-speculative, adders, our analysis demonstrates that variable latency Han-Carlson adders show sensible improvements when the highest speed is required; otherwise the burden imposed by error detection and error correction stages overwhelms any advantage.

II. PRELIMINARIES

A. Prefix Addition:

The binary addition problem can be formulated as follows: given an n-bit augend A=an-1an-2…..a0 and an n-bit addend B=bn-1bn-2…b0 generate the -bit sum S=sn-1sn-2…s0. Let us indicate as ci the carry out of the i-th bit. The sum bit si and the carry ci can be computed as follows:

$$s_i = a_i \oplus b_i \oplus c_{i-1} \quad (2)$$

$$c_i = a_i b_i + a_i c_{i-1} + b_i c_{i-1} \quad (3)$$

In prefix addition we use three stages to compute the sum: pre-processing, prefix-processing and post-processing.

In the pre-processing stage the generate gi and propagate pi signal are computed as:

$$g_i = a_i b_i \quad (4)$$

$$p_i = a_i \oplus b_i \quad (5)$$

The condition gi=1 means that a carry is generated at bit, while the condition pi=1 means that a carry is propagated through bit i.

The concept of generate and propagate can be extended to a block of contiguous bits, from bit K to i bit i(with ≤ i) as follows:

$$g_{[i:k]} = \begin{cases} g_i & \text{if } i = k \\ g_{[i:j]} + p_{[i:j]} g_{[l:k]} & \text{otherwise} \end{cases} \quad (6)$$

$$p_{[i:k]} = \begin{cases} p_i & \text{if } i = k \\ p_{[i:j]} p_{[l:k]} & \text{otherwise} \end{cases} \quad (7)$$

Where: i≥ l≥ j≥ k.

The condition g[i:k]=1 means that a carry is generated in the block k-i , while the condition p[i:k]=1 means that a carry is propagated through the block. Thus, for any bit i, the carry ci can be expressed as:

$$c_i = g_{[i:0]} + p_{[i:0]} c_{-1} \quad (8)$$

Where c-1 is the input carry of the –bit adder. In the following, for the sake of simplicity, we assume that ci-1 , so that (8) simplifies as:

$$c_i = g_{[i:0]} \quad (9)$$

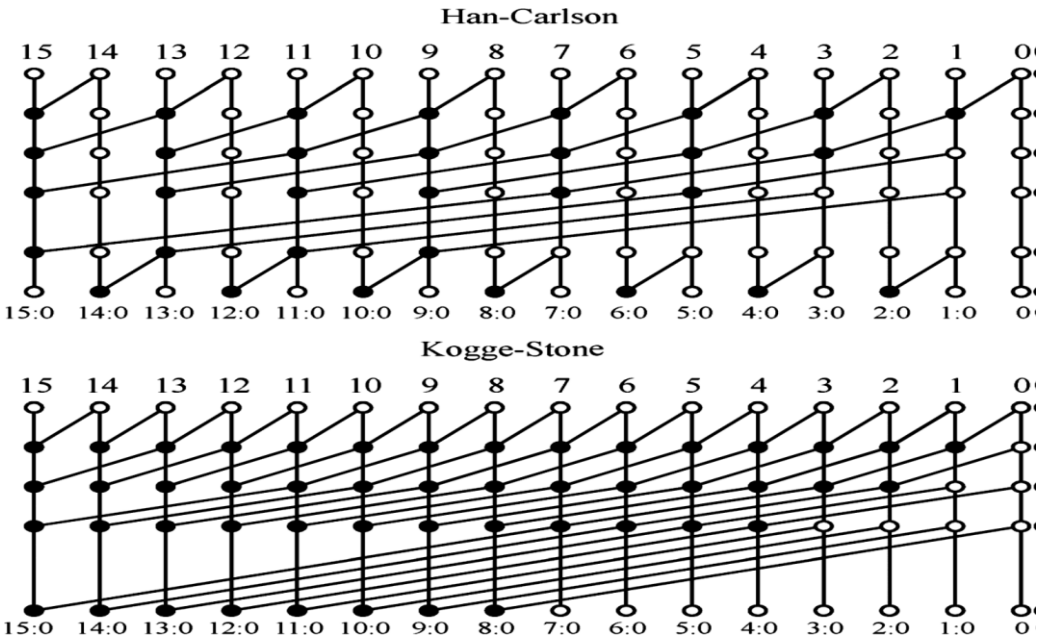

Fig. 1. Han-Carlson and Kogge-Stone parallel-prefix topologies.n=16

The block generate and propagate terms are computed in the prefix-processing stage of the adder. To that purpose, the (g[i:k], p[i:k]) couples are expressed with the help of the prefix operator defined as follows:

$$\left(g_{[i:k]}, p_{[i:k]}\right) = \left(g_{[i:j]}, p_{[i:j]}\right) \bullet \left(g_{[l:k]}, p_{[l:k]}\right) = \left(g_{[i:j]} + p_{[i:j]} g_{[l:k]}, p_{[i:j]} p_{[l:k]}\right) \quad (10)$$

Where: i≥ l≥ j≥ k. The prefix operator has two important properties: it is associative and it is idempotent. These properties are exploited in the prefix-processing stage to speed-up the computation.

Finally, in the post-processing stage, the sum bit are computed using (8) and:

$$s_i = p_i \oplus c_{i-1} \quad (11)$$

III. VARIABLE LATENCY SPECULATIVE PREFIX ADDERS

Variable latency speculative prefix adders can be subdivided in five stages: pre processing, speculative prefix-processing, post-processing, error detection and error correction. The error correction stage is off the critical path, as it has two clock cycles to obtain the exact sum when speculation fails.

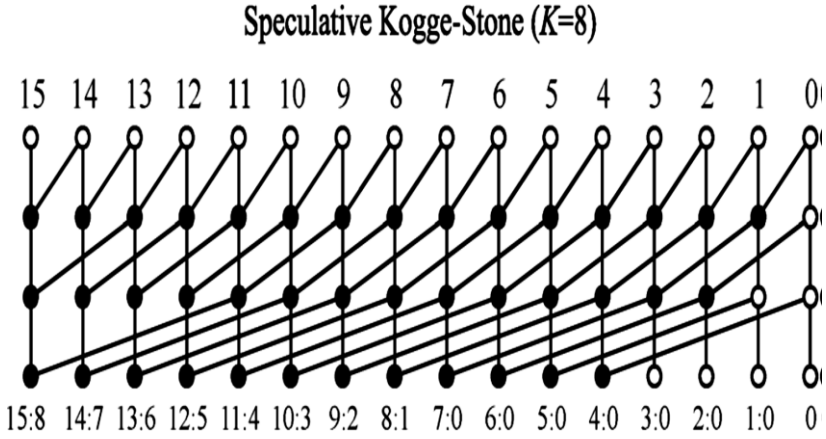

Fig. 2. Kogge-Stone speculative prefix-processing stage

The last row of An=16 bit Kogge-Stone adder is pruned, resulting in a speculative prefix-processing stage with k=8.

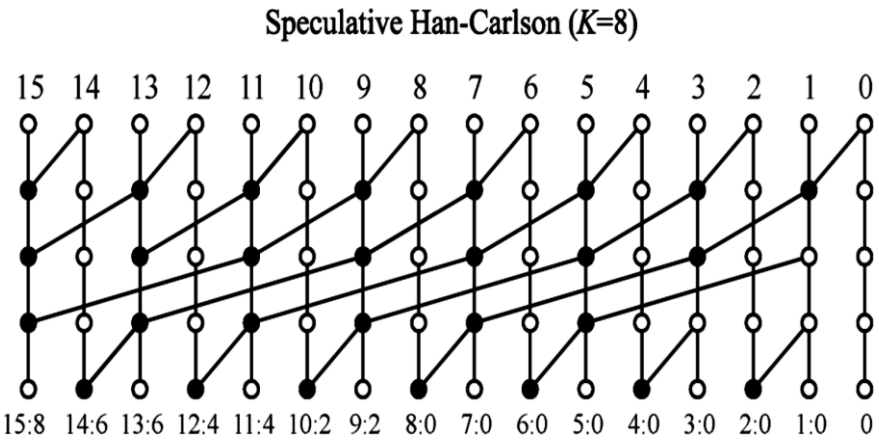

Fig. 3. Han-Carlson speculative prefix-processing stage. The last Kogge-Stone row of the n=16 bit graph is pruned, resulting in a speculative prefix processing stage with k=8

A. Pre-Processing:

In the pre-processing stage the generate gi and propagate pi signals are computed as in (4), (5).

B. Speculative Prefix-Processing:

The speculative prefix-processing stage is one of the main differences compared with the standard prefix adders recalled in previous section. Instead of computing all g[i:0] and p[i:0] the and required in (8) to obtain the exact carry values, only a subset of block generate and propagate signals is calculated; in the postprocessing stage approximate carry values are obtained from this subset. The output of the speculative prefix-processing stage will also be used in the error detection and in the error correction stages.

1) Kogge-Stone Topology*:* The Kogge-Stone speculative prefix-processing stage has been proposed in and can be obtained by pruning the last levels of a traditional Kogge-Stone adder. In the example shown in Fig. 2, the last level of a n=16 bit Kogge-Stone adder is pruned. As it can be observed, for i≥8 the length of propagate chains extends for 8 bits, resulting in a speculative prefix-processing stage with k=8.

In general, the computed propagate and generate signals for the speculative Kogge-Stone architecture are:

$$\begin{array}{ll}(g,p)_{[i:0]} & \text{for}: i \leq K-1 \\ (g,p)_{[i:i-K+1]} & \text{otherwise}\end{array} \qquad (12)$$

2) Han-Carlson Topology: Han-Carlson adder constitutes a good trade-off between fanout, number of logic levels and number of black cells. Because of this, Han-Carlson adder can achieve equal speed performance respect to Kogge-Stone adder, at lower power consumption and area.Therefore it is interesting to implement a speculative Han-Carlson adder.

In general, the computed propagate and generate signals for the speculative Han-Carlson architecture are:

$$\begin{array}{ll}(g,p)_{[i:0]} & \text{for}: i \leq K \\ (g,p)_{[i:i-K+1]} & \text{for}: i > K,\ i \text{ odd} \\ (g,p)_{[i:i-K]} & \text{for}: i > K,\ i \text{ even}\end{array} \qquad (13)$$

C. Post-Processing:

In the post-processing stage we firstly compute the approximate carries,ci , and then use them to obtain the approximate sum bits si as follows:

$$\tilde{s_i} = p_i \oplus \tilde{c}_{i-1} \qquad (14)$$

Similarly to (9), the approximate carries are obtained as the generate signals available in the last level of the prefix-processing stage. We have:

$$\tilde{c}_i = \begin{cases} g_{[i:0]} & \text{for: } i \leq K-1 \\ g_{[i:i-K+1]} & \text{otherwise} \end{cases} \quad (\text{Kogge} - \text{Stone}) \tag{15}$$

and:

$$\tilde{c}_i = \begin{cases} g_{[i:0]} & \text{for: } i \leq K \\ g_{[i:i-K+1]} & \text{for: } i > K,\ i \text{ odd} \\ g_{[i:i-K]} & \text{for: } i > K,\ i \text{ even} \end{cases} \quad (\text{Han} - \text{Carlson}) \tag{16}$$

D. Error Detection:

The conditions in which at least one of the approximate carries is wrong (misprediction) are signaled by the error detection stage. In case of misprediction, an error signal is asserted by error detection stage and the output of the post-processing stage is discarded. The error correction stage will give the correct sum in the next clock period.

1) Kogge-Stone: The error condition for carry can be obtained from (9),(15) and using the properties of propagate generate signals as:

$$e_i = \begin{cases} 0 & \text{for: } i \leq K-1 \\ p_{[i:i-K+1]} g_{[i-K:0]} & \text{otherwise} \end{cases} \tag{17}$$

Thus, the error signal can be expressed as:

$$E_{KS} = \sum_{i=K}^{n-1} p_{[i:i-K+1]} g_{[i-K:0]} \tag{18}$$

It is important to note that (18) is a necessary and sufficient error condition that requires the calculation of g[i-k:0]. Unfortunately, these terms are actually *not* computed by the speculative prefix-processing stage (avoiding the computation of these terms is the key idea of speculative adders). Thus, in previous papers,(18) is replaced by the following looser relation:

$$\tilde{E}_{KS} = \sum_{i=K}^{n-1} p_{[i:i-K+1]} \tag{19}$$

The last equation is a necessary-only error condition. By using (19), the error signal can be triggered even in absence of actual misprediction. While this does not harm the correct operation of the speculative adder, having an high rate of such "false positive" errors degrades the average addition time (1).In this paper, instead, we rewrite the g[i-k:0] necessary and sufficient condition (18) in a form that does not require the terms. To that purpose, let us consider the last two terms of the OR in (18), with index n-1and n-2:

$$p_{[n-1:n-K]} g_{[n-K-1:0]} + p_{[n-2:n-K-1]} g_{[n-K-2:0]} \tag{20}$$

One has:

$$g_{[n-K-1:0]} = g_{n-K-1} + p_{n-K-1} g_{[n-K-2:0]} \tag{21}$$

Substituting (21) in (20), the terms with index n-1 and n-2 of (18) can be simplified as:

$$p_{[n-1:n-K]} g_{n-K-1} + p_{[n-2:n-K-1]} g_{[n-K-2:0]} \tag{22}$$

Similar simplifications can be realized by considering in (18) the terms n-2 and n-3 and so on.

Finally one obtains:

$$E_{KS} = \sum_{\substack{i=K \\ i \text{ odd}}}^{n-1} p_{[i:i-K+1]} g_{i-K} \qquad (23)$$

Let us consider, as an example, the prefix-processing stage inFig. 2. The error signal (23) is given by:

$$E_{KS} = p_{[8:1]} g_0 + p_{[9:2]} g_1 + p_{[10:3]} g_2 + \ldots + p_{[15:8]} g_7 \quad (24)$$

2) Han-Carlson: The error condition for carry can be obtained from (9), (16) as:

$$e_i = \begin{cases} 0 & \text{for: } i \leq K \\ p_{[i:i-K+1]} g_{[i-K:0]} & i > K,\ i \text{ odd} \\ p_{[i:i-K]} g_{[i-K-1:0]} & i > K,\ i \text{ even} \end{cases} \qquad (25)$$

The error signal can be written as:

$$E_{HC} = \sum_{\substack{i=K+1 \\ i \text{ odd}}}^{n-1} p_{[i:i-K+1]} g_{[i-K:0]} + \sum_{\substack{i=K+1 \\ i \text{ even}}}^{n-1} p_{[i:i-K]} g_{[i-K-1:0]} \qquad (26)$$

It can easily be seen that in (26) the terms in the second OR are implied by the terms in the first OR. Let us consider, for instance, the first two terms of the OR (assuming that is even).We have:

$$p_{[K+1:2]} g_{[1:0]} + p_{[K+2:2]} g_{[1:0]} = p_{[K+1:2]} g_{[1:0]} \qquad (27)$$

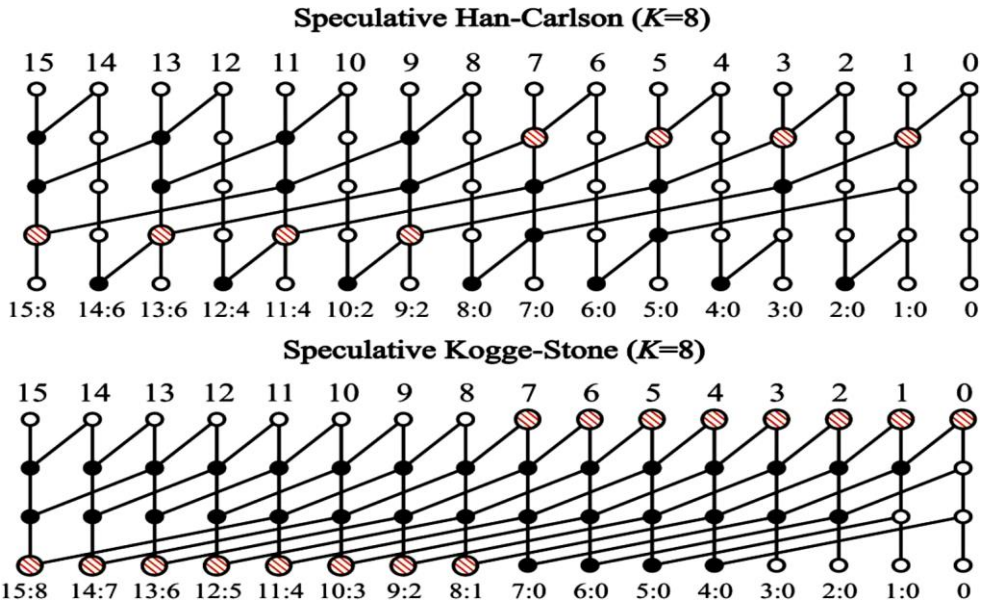

Fig. 4. The nodes of the prefix-processing stage, whose outputs are needed to compute the error signal, are named "checking nodes" and are highlighted as big hatched dots, for the topologies in Fig. 2–3.

Thus, we can write:

$$E_{HC} = \sum_{\substack{i=K+1 \\ i \text{ odd}}}^{n-1} p_{[i:i-K+1]} g_{[i-K:0]} \qquad (28)$$

The last equation can be simplified, following an approach similar to previous subsection.

Let us consider the last two terms of the OR in (28), with index n-1 and n-3 (assuming that n is even):

$$p_{[n-1:n-K]}g_{[n-1-K:0]} + p_{[n-3:n-2-K]}g_{[n-3-K:0]} \quad (29)$$

One has:

$$g_{[n-1-K:0]} = g_{[n-1-K:n-2-K]} + p_{[n-1-K:n-2-K]}g_{[n-3-K:0]} \quad (30)$$

Substituting (30) in (29), the terms with index n-1 and n-3 of (28) can be simplified as:

$$p_{[n-1:n-K]}g_{[n-K-1:n-K-2]} + p_{[n-3:n-K-2]}g_{[n-K-3:0]} \quad (31)$$

Similar simplifications can be realized by considering in (28) the terms n-3 and n-5 and so on. Finally one obtains:

$$E_{HC} = \sum_{\substack{i=K+1 \\ i \text{ odd}}}^{n-1} p_{[i:i-K+1]}g_{[i-K:i-K-1]} \quad (32)$$

Let us consider, as an example, the prefix-processing stage in Fig. 3. The error signal (32) is given by:

$$E_{HC} = p_{[9:2]}g_{[1:0]} + p_{[11:4]}g_{[3:2]} + \ldots + p_{[15:8]}g_{[7:6]} \quad (33)$$

By comparing (23) and (32), it can easily be seen that the number of terms to be OR-ed to obtain the error signal is halved in the Han-Carlson topology, compared to Kogge-Stone.

We name "checking nodes" the nodes of the prefix-processing stage, whose outputs are needed to compute the error signal. The checking nodes for both the Kogge-Stone example of Fig. 2 and the Han-Carlson example of Fig. 3 are highlighted as big hatched dots in Fig. 4.

As it can be observed, in Kogge-Stone some of the checking cells are at the last level of the graph; their output signals are available after three black cells delay. In Han-Carlson the critical checking cells are in the second last level of the graph and are also available after three black cells delay, in spite of the larger number of levels of the Han-Carlson prefix-processing stage.

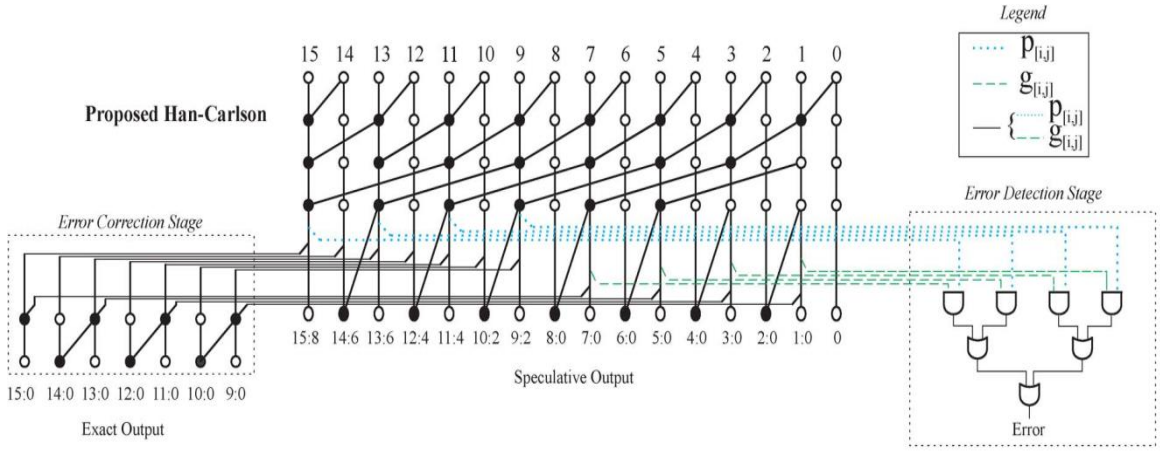

Fig. 5. Error correction and detection stages for the proposed speculative Han-Carlson adder of Fig. 3

From the above observations, it can be concluded that error detection is sensibly simplified and potentially faster in Han-Carlson, compared to Kogge-Stone.

As an additional note, the need of driving the gates of the error detection stage increases the fanout of the checking cells, slowing the speculative prefix-processing stage.

E. Error Correction:

The error correction stage computes the exact carry signals (9), to be used in case of misprediction. The error correction stage is composed by the levels of the prefix-processing stage pruned to obtain the speculative adder. The Fig. 5 shows the error correction stage of the proposed speculative Han-Carlson adder; the error correction for Kogge-Stone topology can be obtained similarly. It can be observed that the inclusion of the error correction stage increases the fanout of some of the cells of the speculative prefix-processing stage, with adverse effect on adder speed.

F. Post-Processing:

The approximate carries are already available at the output of the prefix-processing stage. The post-processing, according to (14), is equal to the one of a non-speculative adder and consists of xor gates.

IV. ADDERS CHARACTERIZATION

In this section we provide a characterization of the spatial and timing complexity of the investigated variable latency speculative adders, using either Han-Carlson or Kogge-Stone topologies. Results for non-speculative adders are also reported, for comparison. This will be achieved with the help of simplistic hypotheses on area and speed of employed gates, with the aim of obtaining an analytic comparison (albeit approximated) between

the various topologies. Accurate values of area, speed and power for 65 nm technology will be presented in the next section for a quantitative assessment of variable latency speculative adders. Results of error rate analysis will also be reported at the end of this section.

A. Spatial and Timing Complexity:

In order to estimate adder complexity, we make some simplistic hypotheses. We assume that the spatial complexity of speculative prefix processing and error correction is proportional to the number of employed black cells (AO gates). Error detection spatial complexity is simply estimated assuming that it is composed by a set of AND gates (to compute the p.g terms in (23) or in (32)) followed by a tree of two-input OR to compute the error signal (see Fig. 5 for an example). we assume as unit gate a basic 2-input gate, such as AND gates and OR gates, while we count black cells (AO gates) as two unit gates.

Regarding delay, we assume that speculative sum delay is proportional to the number of levels of speculative parallel prefix stage, plus two additional levels to take into account preprocessing and post-processing. Error detection delay is estimated as the number of OR-tree levels, plus one additional level to take into account the AND gates computing the p.g terms in (23) or in (32). Assuming unit gate delay model of [18], we count the basic 2-input gates such as AND and OR as one gate delay with the exception of the XOR gates which we count as two gate delays.

For speculative adders, spatial complexity is reported as the sum of two contributions. The first one (curly brackets) is the contribution of speculative prefix stage and error correction stage, the second one (square brackets) is the contribution of error detection stage. As it can be observed, the two area contributions are both lower in the proposed Han-Carlson speculative adder, compared to Kogge- Stone. It also worth noting that the spatial complexity of speculative adders is higher than non-speculative ones, because of error detection and correction stages.

The Kogge-Stone adder saves two gate levels to perform the speculative sum. However, the critical path traverses the error detection stage and hence the proposed Han-Carlson architecture appears faster than Kogge-Stone speculative adder, owing to the halved number of terms to be OR-ed to obtain the error signal.

B. Error Rate Analysis:

The value of error probability is fundamental to understand the degradation of average addition time (1) caused by misprediction. In order to evaluate error probabilities, the proposed speculative Han-Carlson and the Kogge-Stone topologies have been simulated by using a Monte Carlo approach with a 1% relative error and a 99% confidence level.

For Kogge-Stone we name "Precise" the error detection stage based on equations (23), while we name "Coarse" the one based on the necessary-only error condition (19); similar naming convention is used for Han-Carlson topology.

The Precise error detection stage significantly reduces the error probability, compared to the Coarse one. Moreover, Han-Carlson speculative adder exhibits a lower error probability than Kogge-Stone one. This can be interpreted as follows: in Kogge- Stone speculative prefix stage all the carries are computed independently from each other, instead in Han-Carlson, half of the carries (those in even bit-positions) are calculated from "parents" carries (those in odd bit-positions), through an additional level of the tree. This reduces error probability (if a parent carry is correct the "child" carry will be correct, too).

In the following we will consider only implementations with the Precise error detection stage which, as shown, provides lower error probabilities than Coarse detection.

TABLE I ERROR PROBABILITY VALUES

Speculative Adder	*n*	*K=8*		*K=16*	
		Precise	*Coarse*	*Precise*	*Coarse*
Han-Carlson (proposed)	32	1.74×10^{-2}	3.58×10^{-2}	4.57×10^{-5}	9.53×10^{-5}
Kogge-Stone	32	2.32×10^{-2}	4.82×10^{-2}	6.09×10^{-5}	1.29×10^{-4}
Han-Carlson (proposed)	64	4.06×10^{-2}	8.05×10^{-2}	1.37×10^{-4}	2.78×10^{-4}
Kogge-Stone	64	5.39×10^{-2}	1.07×10^{-1}	1.83×10^{-4}	3.74×10^{-4}
Han-Carlson (proposed)	128	8.45×10^{-2}	1.63×10^{-1}	3.21×10^{-4}	6.44×10^{-4}
Kogge-Stone	128	1.11×10^{-1}	2.12×10^{-1}	4.27×10^{-4}	8.61×10^{-4}

V. SYNTHESIS RESULTS

We have developed Matlab scripts which generate Verilog descriptions of the proposed variable latency speculative adders, and of their non-speculative counterpart. The set_multicycle_path synthesis command was used to mark the non-speculative outputs of the speculative adders. We have synthesized these adders in UMC 65 nm library, for 32 bit, 64 bit, and 128 bit operands.

It is not easy to compare performances (in terms of power, speed, and area) of different designs, since they strongly depend on timing constraint used during synthesis. The results reported in the following have been obtained by performing several syntheses of the circuits under investigation, by varying the timing constraint. In this way we can compare the various topologies and find the most effective ones depending on the required speed.

The dynamic power dissipation has been evaluated after synthesis by extracting the nodes activities from a back-annotated simulation.

A. The Optimal K Choice:

The optimal K value descends from the following trade-off: if we increase K we reduce the error probability (with positive effects on average delay (1)) but we make parallel-prefix stage slower (a little number of levels is pruned) and we make also error detection slower (because the checking-cells descend toward last levels).

Comparison between variable latency adder and the non-speculative Han-Carlson topology reveal that variable latency adders allow to reduce the minimum achievable delay. For instance, in the 64 bit case, the minimum achievable delay is about 280 ps for the non-speculative adder and reduces up to 225 ps in the variable latency architecture.

The analysis of Area Occupation and Power Dissipation shows that speculative adders are not effective for large average delay. As the timing constraint imposed during synthesis is made tighter speculative adders become advantageous.

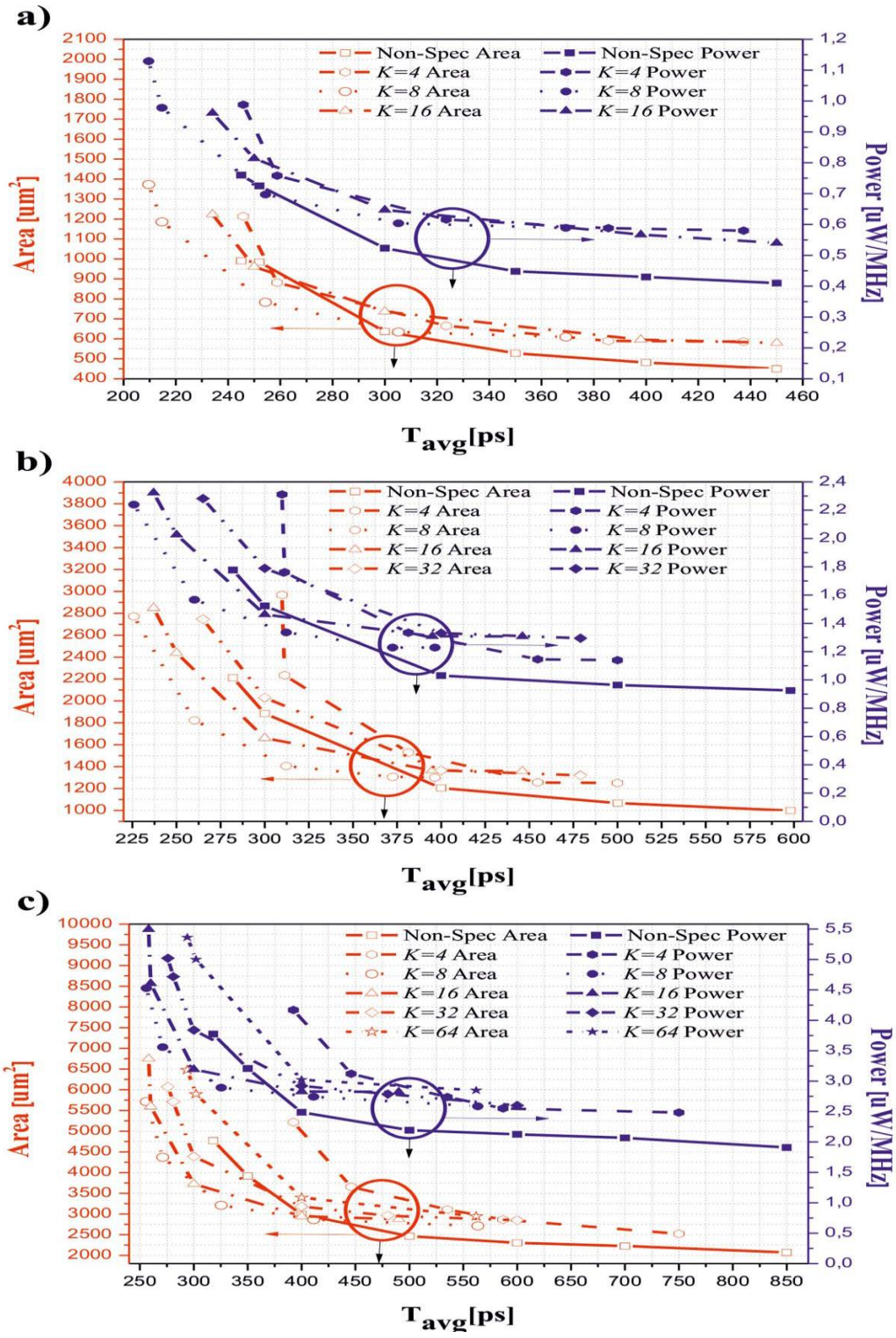

Fig. 6. Area and power of Han-Carlson speculative and non-speculative adders, as a function of the timing constraint. (a) 32 bit, (b) 64 bit, (c) 128 bit. Different K values are used for speculative adders.

B. Comparison with Kogge-Stone Variable Latency Speculative Adder:

Fig. 7 shows the comparison between proposed speculative adder and Kogge-Stone one. Also in this case, we report the performance of non-speculative adders, in order to identify the region where the speculative approach is effective (the optimum value for the variable latency speculative Kogge-Stone adder is: K=8 for n=32bit ,K=16 for n=16 and n=128bit).

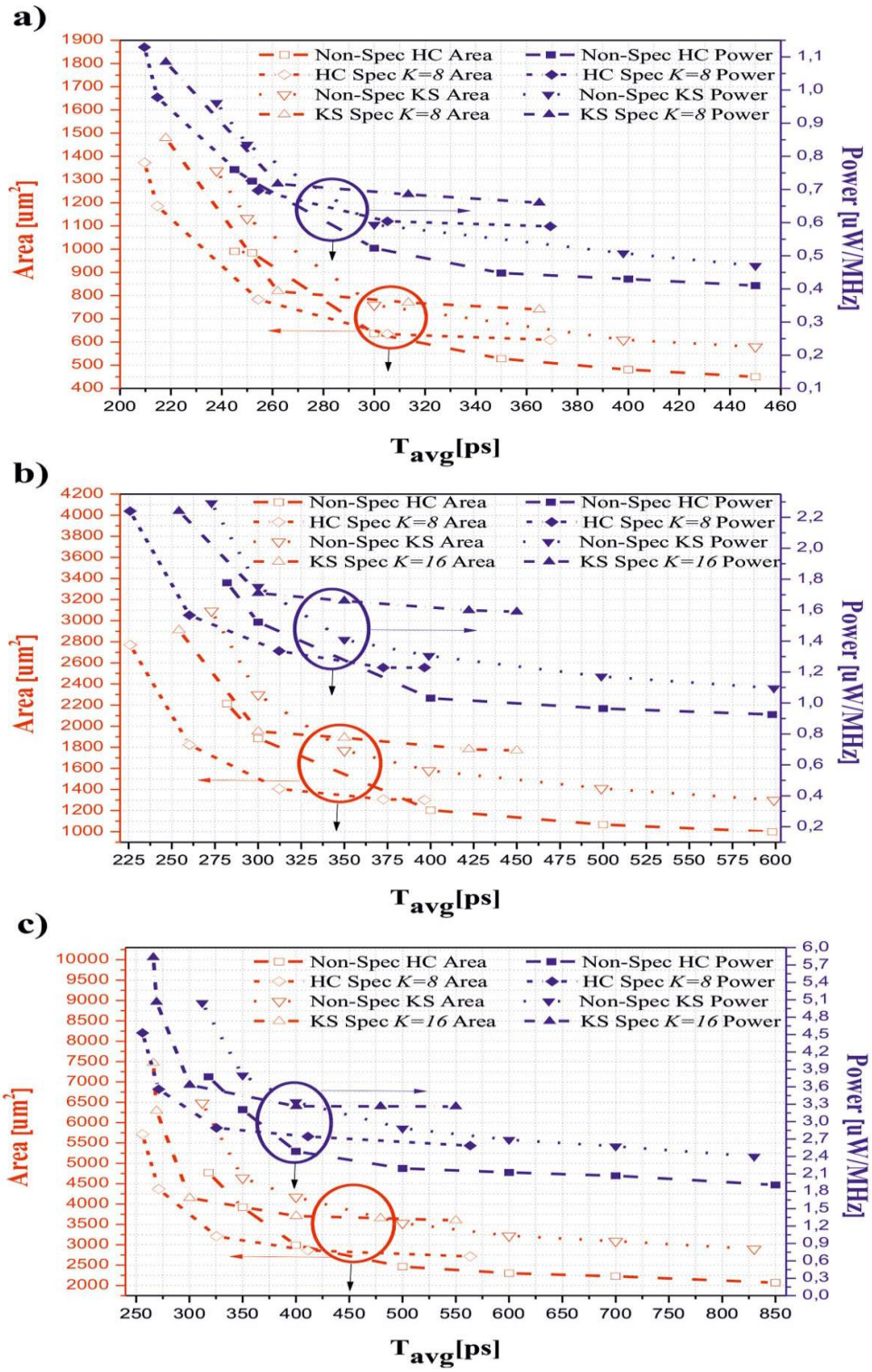

Fig.7. Comparison of Han-Carlson and Kogge-Stone speculative and non-speculative adders, as a function of the timing constraint. (a) 32 bit, (b) 64 bit, (c) 128 bit.

The proposed variable latency Han-Carlson adder outperforms the speculative Kogge-Stone architecture in all the considered cases, confirming the trend highlighted in Table I.

As an example, focusing on 64-bit adders, for Tavg lower than 350 ps, the proposed Han-Carlson speculative adder is the best choice in terms of silicon area and power consumption. Moreover, it allows to reduce the minimum achievable Tavg to 225 ps, with a 18% improvement respect to Kogge-Stone non-speculative adder and a 11% improvement respect to Kogge-Stone speculative adder. For Tavg = 273 ps, proposed speculative adders offer 45% area reduction and 35% power saving compared to Kogge-Stone non-speculative adder.

VI. CONCLUSION

In this paper a novel variable latency Han-Carlson parallel prefix speculative adder for high-speed application is proposed. A new, more accurate, error detection network is introduced, which allows reducing the error probability compared to the previous approaches.

An extensive set of implementation results for 65 nm CMOS technology shows that proposed Han-Carlson variable latency adders outperforms previously developed variable latency Kogge-Stone architectures. Compared with traditional, non-speculative, adders, our analysis demonstrates that variable latency Han-Carlson adders show sensible improvements when the highest speed is required; otherwise the burden imposed by error detection and error correction stages overwhelms any advantage. Additional work is required to extend the speculative approach to other parallel-prefix architectures, such as Brent-Kung, Ladner-Fisher, and Knowles.

REFERENCES

[1] I. Koren, *Computer Arithmetic Algorithms*. Natick, MA, USA: A K Peters, 2002.

[2] R. Zimmermann, "Binary adder architectures for cell-based VLSI and their synthesis," Ph.D. thesis, Swiss Federal Institute of Technology, (ETH) Zurich, Zurich, Switzerland, 1998, Hartung-Gorre Verlag.

[3] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Trans. Comput.*, vol. C-31, no. 3, pp. 260–264, Mar. 1982.

[4] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Comput.*, vol. C-22, no. 8, pp. 786–793, Aug. 1973.

[5] J. Sklansky, "Conditional-sum addition logic," *IRE Trans. Electron. Comput.*, vol. EC-9, pp. 226–231, Jun. 1960.

[6] T. Han and D. A. Carlson, "Fast area-efficient VLSI adders," in *Proc. IEEE 8th Symp. Comput. Arith. (ARITH)*, May 18–21, 1987, pp. 49–56.

[7] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," *J. ACM*, vol. 27, no. 4, pp. 831–838, Oct. 1980.

[8] S. Knowles, "A Family of Adders," in *Proc. 14th IEEE Symp. Comput. Arith.*, Vail, CO, USA, Jun. 2001, pp. 277–281.

[9] S.-L. Lu, "Speeding up processing with approximation circuits," *Computer*, vol. 37, no. 3, pp. 67–73, Mar. 2004.

[10] T. Liu and S.-L. Lu, "Performance improvement with circuit-level speculation," in *Proc. 33rd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO-33)*, 2000, pp. 348–355.

[11] N. Zhu, W.-L. Goh, and K.-S. Yeo, "An enhanced low-power highspeed Adder For Error-Tolerant application," in *Proc. 2009 12th Int. Symp. Integr. Circuits (ISIC '09)*, Dec. 14–16, 2009, pp. 69–72.

[12] S. M. Nowick, "Design of a low-latency asynchronous adder using speculative completion," *IEE Proc. Comput. Digit. Tech.*, vol. 143, no. 5, pp. 301–307, Sep. 1996.

[13] A. K. Verma, P. Brisk, and P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design," in *Proc. Design, Autom., Test Eur. (DATE '08)*, Mar. 2008, pp. 1250–1255.

[14] A. Cilardo, "A new speculative addition architecture suitable for two's complement operations," in *Proc. Design, Autom., Test Eur. Conf. Exhib. (DATE '09)*, Apr. 2009, pp. 664–669.

[15] K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in *Proc. Design, Autom., Test Eur. Conf. Exhib. (DATE '12)*, Mar. 2012, pp. 1257–1262.

[16] S. K. Mathew, R. K. Krishnamurthy, M. A. Anders, R. Rios, K. R. Mistry, and K. Soumyanath, "Sub-500-ps 64-b ALUs in 0.18- m SOI/ bulk CMOS: Design and scaling trends," *IEEE J. Solid-State Circuits*, vol. 36, no. 11, pp. 1636–1646, Nov. 2001.

[17] Parhami, *Computer Arithmetic: Algorithms and Hardware Design*. New York: Oxford Univ. Press, 2000.