International Journal of VLSI System Design and Communication Systems

ISSN 2322-0929
Vol.04, Issue.12,
November-2016,
Pages:1439-1443

www.ijvdcs.org

Design of Efficient Han-Carlson Adder

P. ANNAPURNA BAI[1], B. THIRUPATHAIAH[2]

[1]Assistant Professor, Dept of ECE, CRIT Engineering College, Anantapuramu, AP, India.
[2]PG Scholar, Dept of ECE, CRIT Engineering College, Anantapuramu, AP, India.

Abstract: Variable latency adders have been as of late proposed in writing. A variable latency viper utilizes hypothesis the careful arithmetic capacity is supplanted with an approximated one that is speedier and gives the right result more often than not, yet not generally. The approximated viper is enlarged with a blunder discovery arrange that declares a mistake sign when hypothesis comes up short. Theoretical variable latency adders have pulled in solid interest on account of their ability to lessen normal postponement contrasted with conventional structures. This paper proposes a novel variable latency theoretical viper taking into account Han-Carlson parallel-prefix topology that came about more successful than variable latency Kogge-Stone topology. The paper depicts the phases in which variable latency theoretical prefix adders can be subdivided and displays a novel error location organize that decreases error likelihood contrasted with past methodologies. A few variable latency theoretical adders, for different operand lengths, utilizing both Han-Carlson and Kogge-Stone topology, have been synthesized utilizing the UMC 65 nm library. Gotten results demonstrate that proposed variable latency Han-Carlson snake beats both already proposed theoretical Kogge-Stone models and non-speculative adders, when rapid is required. It is additionally demonstrated that non-theoretical adders remain the best decision when the velocity requirement is relaxed.

Keywords: Variable Latency, Adders, Speculative Adders, Parallel-Prefix Adders.

I. INTRODUCTION

Adders are fundamental utilitarian units in PC number juggling. Double adders are utilized as a part of chip for addition and subtraction operations and for floating point multiplication and division**.** Therefore adders are fundamental components and improving their performance is one of the major challenges in digital designs. Theoretical research [1] has established lower bounds on area and delay of -bit adders: the former varies linearly with adder size, the latter has an O($\log_2$ (n)) behavior. High speed adders depend on entrenched parallel prefix structures [1], [2], including Brent-Kung [3], Kogge-Stone [4], Sklansky [5], Han-Carlson [6], Ladner-Fischer, Knowles. These standard structures work with fixed latency. Better average performances can be achieved by using variable latency adders, that have been recently proposed in literature. A variable latency viper utilizes hypothesis: the definite number-crunching capacity is supplanted with an approximated one that is quicker and gives the right result more often than not, however 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 happens and two clock cycles when the speculation fails, the normal expansion time can be computed as (1) where is the clock time frame and is the error probability of the theoretical viper. Theoretical adders are based upon the perception that the basic way is once in a while enacted in conventional adders [1]–[6].

In particular, in traditional adders each output depends on all previous bits, so the most significant output depends on all the input bits. instead, in theoretical adders every yield depends just on the past bits, where goes as [2]–[5]. This mirrors the way that a proliferate chain longer than is an exceptionally uncommon event.A first theoretical way to deal with expansion was proposed by Nowick [2] in asynchronous challenge, which executes a variable latency snake cutting the least levels of a Kogge-Stone adder. In synchronous challenge, Verma et al.[3] propose a variable latency speculative adder; here the theoretical expansion is acknowledged similarly as [2], cutting the lower levels of a Kogge-Stone adder. A comparative methodology is utilized in [4]. In [5] a variable latency pass on select snake is exhibited, where the adder is isolated in various windows, each one containing a Kogge-Stone adder. The Kogge-Stone adder is frequently utilized when velocity is the essential worry, since it utilizes the base number of rationale levels and every cell in the snake tree has fanout of 2.This comes at the expense of utilizing numerous propagate-generate cells and numerous wires that must be directed between stages. The paper presents a thorough determination of the mistake location system and demonstrates that the blunder discovery system required in theoretical Han-Carlson adders is fundamentally quicker than the one utilized by theoretical Kogge-Stone architecture. A broad arrangement of usage results for 65 nm CMOS innovation demonstrates that proposed Han-Carlson variable latency adders outflank already created variable latency Kogge-Stone architectures. Compared with customary, non-theoretical, adders, our

investigation shows that variable latency Han-Carlson adders show sensible improvements when the highest speed is required.

II. PRELIMINARIES

A. Prefix Addition

The binary addition problem can be formulated as follows: given an -bit augend $A=a_{n-1,}\ a_{n-2,\ldots\ldots}\ a_{0.}$ and an n-bit addend $B=b_{n-1,}\ b_{n-2,\ldots\ldots}\ b_{0.}$ generate the n-bit sum $S= s_{n-1,}\ s_{n-2,\ldots\ldots}\ s_0$ Give us a chance to show as Ci the carry out of the i-th bit The whole piece Si and the carry Ci can be computed asfollows:

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

In prefix addition we use three stages to register the sum: pre-processing, prefix-processing and post-preparing. In the pre-handling stage the generate Gi and propagate Pi signal are figured as:

$$g_i = a_i b_i$$
$$p_i = a_i \oplus b_i \quad (2)$$

The condition G_i =1 means that a carry is generated at bit I while the condition P_i=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 bit i (with k< i) as follows:

$$g_{[i:k]} = \begin{cases} g_i & \text{if } i = k \\ g_{[i:j]} + p_{[i:j]} g_{[l:k]} & \text{otherwise} \end{cases}$$
$$p_{[i:k]} = \begin{cases} p_i & \text{if } i = k \\ p_{[i:j]} p_{[l:k]} & \text{otherwise} \end{cases} \quad (3)$$

where: $i \geq l \geq j \geq k$.

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

$$C_i = g[i:0] + p[i:0]c\text{-}1 \quad (4)$$

Where C-1 is the information convey of the n-bit adder. In the accompanying, for straightforwardness, we accept that Ci-1=0, so that (4) follows as:

$$C_i = g[i:0] \quad (5)$$

The block generate and propagate terms are registered in the prefix-preparing phase of the adder. To that reason, the (g[i:k] ,p[i:k]) couples are communicated with the assistance of the prefix operator characterized as takes follows:

$$(g_{[i:k]}, p_{[i:k]}) = (g_{[i:j]}, p_{[i:j]}) \bullet (g_{[l:k]}, p_{[l:k]}) = (g_{[i:j]} + p_{[i:j]} g_{[l:k]}, p_{[i:j]} p_{[l:k]}) \quad (6)$$

Han carlson adder

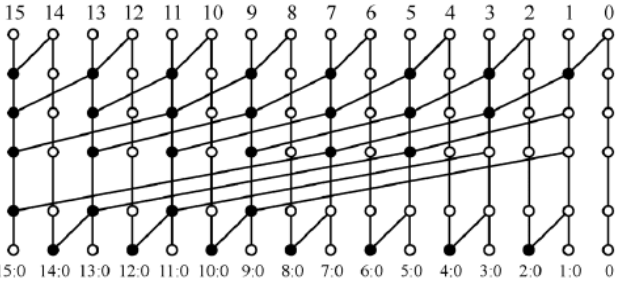

Fig.1. Han-Carlson Parallel-Prefix Topologies. n=16

III. VARIABLE LATENCY SPECULATIVE PREFIX ADDERS.

Variable latency speculative prefix adders can be subdivided in five phases: pre-handling, speculative prefix-processing, post-processing, blunder 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.

A. Pre-Processing

In the pre-preparing stage the generate G1 and propagate Pi signs are computed as in (4), (5).

B. Speculative Prefix-Processing

The speculative prefix-processing stage is one of the fundamental contrasts contrasted and the standard prefix adders reviewed in past area. Rather than figuring all the g[i:0] and p[i:0] required in (7) to get the accurate c cluster values, just a subset of square generate and propagate signs is computed; in the post processing stage surmised convey qualities are acquired from this subset. The yield of the speculative prefix-processing stage will likewise be utilized as a part of the error detection and in the error correction stages examined in the following. The basic assumption behind speculative prefix-processing stage is that carry signals propagate for no more than K bits, with k<n and k=O(log(n)). This assumption is corroborated by the analyses in [3],[6] that demonstrate that having a propagate chain longer that log(n) is a very rare event.

1. Kogge-Stone Topology: The Kogge-Stone speculative prefix-processing stage has been proposed in [2],[3] and can be obtained by pruning the last levels of a traditional Kogge-Stone adder. In the example appeared in Fig. 2, the last level of a n=16 bit Kogge-Stone adder is pruned. As it can be seen, for i>8 the length of propagate chains stretches out for 8 bits, bringing about a speculative prefix-processing stage with k=8. In general, one has $K=n/2^p$, where P is the number of pruned levels; the number of levels of the speculative stage is correspondingly reduced from $\log_2(n)$ to $\log_2(k)$ (assuming that K is a force of two). When all is said in done, the registered propagate and generate signals for the speculative Kogge-Stone design are:

$$(g,p)_{[i:0]} \quad \text{for} : i \leq K-1$$
$$(g,p)_{[i:i-K+1]} \quad \text{otherwise} \quad (7)$$

2. Han-Carlson Topology: Han-Carlson adder contains a good trade-off between fan out, number of logic levels and number of black cells. because of this, Han-Carlson adder can achieve equal to speed execution admiration to Kogge-Stone adder, at lower power utilization and territory [6]. In this manner it is fascinating to execute a speculative Han-Carlson adder. Moved by these reasons, we have generated a Han-Carlson speculative prefix-processing stage by removing the final rows of the Kogge-Stone part of the adder. As an example, the Fig.2 shows the Han-Carlson adder of Fig. 1 in which the two Brent-Kung rows at the initial and toward the end of the graph are unaltered, while the last Kogge-Stone row is pruned. This yields a speculative stage with

K=8=n/2.in general, one has K ==n/2p. where P is the quantity of pruned levels; the number of levels of the speculative Han-Carlson stage lessens from 1+ log(n) to1+ log(k) (assuming that is a power of two).

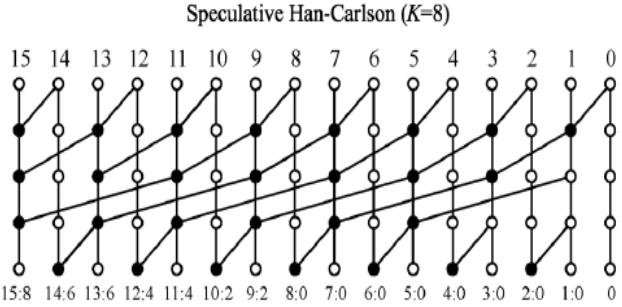

Fig.2. Han-Carlson Speculative Prefix-Processing Stage

As it can be shown in Fig.2, the length of the propagate chains is K=8 only for i=9,11,13,15 , while for i=10,12,14 the propagate chain length is K+1=9. In casual, the computed propagate and generate signals for the speculative Han-Carlson design are:

$$\begin{aligned} &(g,p)_{[i:0]} && \text{for}: i \le K \\ &(g,p)_{[i:i-K+1]} && \text{for}: i > K,\ i \text{ odd} \\ &(g,p)_{[i:i-K]} && \text{for}: i > K,\ i \text{ even} \end{aligned} \tag{8}$$

As it will be apparent in the following, having the propagate lengths equal to k+1 for half of the outputs greatly simplifies the error detection.

C. Post-Processing

In the post-processing stage we firstly compute the approximate carries, $\tilde{c}_i$, and then use them to find the approximate sum $\tilde{s}_i$, bits as follows:

$$\tilde{s}_i = p_i \oplus \tilde{c}_{i-1} \tag{9}$$

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

$$\tilde{c}_i = \begin{cases} g_{[i:0]} & \text{for: } i \le K-1 \\ g_{[i:i-K+1]} & \text{otherwise} \end{cases} \quad (\text{Kogge} - \text{Stone}) \tag{10}$$

And

$$\tilde{c}_i = \begin{cases} g_{[i:0]} & \text{for: } i \le K \\ g_{[i:i-K+1]} & \text{for: } i > K,\ i \text{ odd} \\ g_{[i:i-K]} & \text{for: } i > K,\ i \text{ even} \end{cases} \quad (\text{Han} - \text{Carlson}) \tag{11}$$

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 deleted. The error correction stage will give the correct sum in the next clock period.

1. Kogge-Stone: The error condition for convey can be gotten from (9),(12) and utilizing the properties of propagate generate signals as:

$$e_i = \begin{cases} 0 & \text{for: } i \le K-1 \\ p_{[i:i-K+1]} g_{[i-K:0]} & \text{otherwise} \end{cases} \tag{12}$$

Thus, the error signal can be expressed as:

$$E_{KS} = \sum_{i=K}^{n-1} p_{[i:i-K+1]} g_{[i-K:0]} \tag{13}$$

where the symbol represents the logical OR. It is important to note that (14) is a necessary and sufficient error condition that requires the calculation of . Unfortunately, these terms are really not registered by the speculative prefix-processing stage (dodging the calculation of these terms is the key thought of speculative adders). Therefore, in past papers,(14) is supplanted by the accompanying looser relation:

$$\tilde{E}_{KS} = \sum_{i=K}^{n-1} p_{[i:i-K+1]} \tag{14}$$

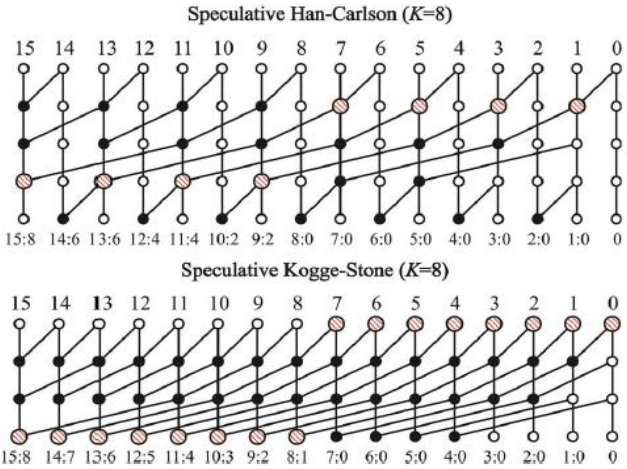

Fig.3. The Nodes of the Prefix-Processing Stage, Whose Outputs are Needed to Compute the Error Signal, are Named "Checking Nodes" and Are Highlighted as Big Hatched Dots, for the Topologies

As it can be obtained, 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 .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 blunder remedy stage registers the careful convey signals (14), to be utilized as a part of instance of misprediction. The mistake remedy stage is made by the levels out of the prefix-handling stage pruned to acquire the speculative adder.The Fig.4 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.

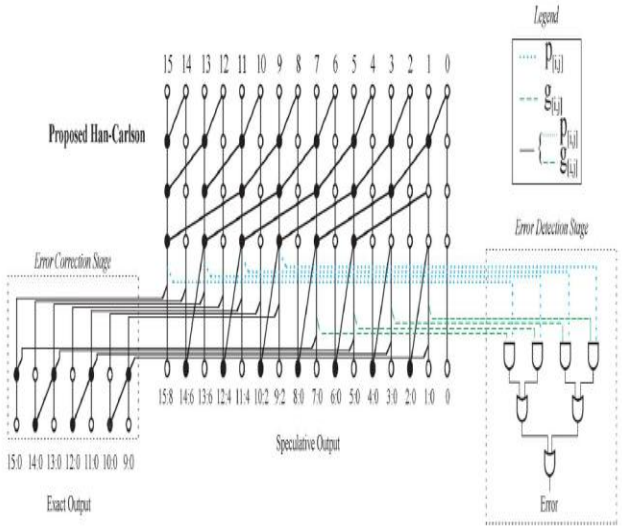

Fig.4. Error Correction, Detection Stages for the Proposed Speculative Han-Carlson Adder of Fig. 3.

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 terms in (13) or in (14) followed by a tree of two-input OR to compute the error signal (see Fig. 5 for an example). According to the model proposed in [6] 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 terms in (13) or in (14). Assuming unit gate delay model of [1], 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.

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. Input vectors have been chosen uniformly distributed [12],[13]. Table II reports the results of the analysis. 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.

V. RESULTS

We have developed Mat lab scripts which generate Verilog descriptions of the proposed variable latency speculative adders, and of their non-speculative counterpart.

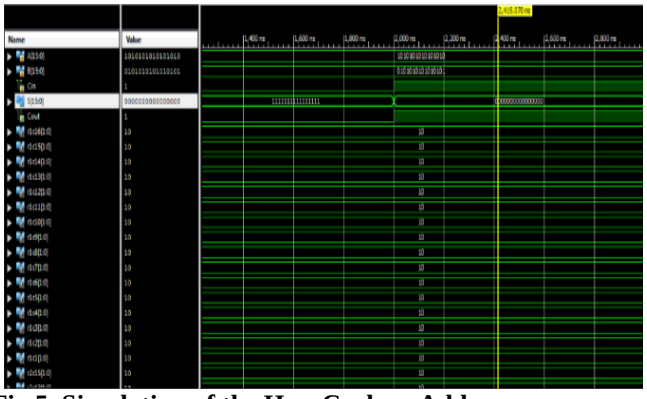

Fig.5. Simulation of the Han-Carlson Adder.

carlson_128bit Project Status (05/06/2016 - 11:20:35)			
Project File:	kachi.xise	Parser Errors:	No Errors
Module Name:	carlson_16bit	Implementation State:	Placed and Routed
Target Device:	xc3s500e-5fg320	• Errors:	
Product Version:	ISE 14.7	• Warnings:	
Design Goal:	Balanced	• Routing Results:	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	0 (Timing Report)

Device Utilization Summary				[-]
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	38	9,312	1%	
Number of occupied Slices	24	4,656	1%	
Number of Slices containing only related logic	24	24	100%	
Number of Slices containing unrelated logic	0	24	0%	
Total Number of 4 input LUTs	38	9,312	1%	
Number of bonded IOBs	50	232	21%	
Average Fanout of Non-Clock Nets	2.27			

Fig.6. Area Report of the Han-Carlson Adder.

```
Data Path: B<0> to Cout
                            Gate     Net
  Cell:in->out      fanout  Delay    Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O             2   1.106   0.532  B_0_IBUF (B_0_IBUF)
   LUT3:I0->O            2   0.612   0.449  ir1c1/pgo1 (r2c1)
   LUT3:I1->O            2   0.612   0.449  ixor16/Mxor_S_Result<2>11 (N3)
   LUT3:I1->O            2   0.612   0.449  ir2c3/pgo1 (r3c3)
   LUT3:I1->O            2   0.612   0.449  ixor16/Mxor_S_Result<4>11 (N4)
   LUT3:I1->O            2   0.612   0.449  ir5c5/pgo1 (r6c5)
   LUT3:I1->O            3   0.612   0.454  ixor16/Mxor_S_Result<6>11 (N5)
   LUT4:I3->O            1   0.612   0.360  ir6c11/pgo41 (ir6c11/pgo41)
   LUT4:I3->O            1   0.612   0.387  ir6c11/pgo89_SW0 (N36)
   LUT3:I2->O            2   0.612   0.449  ir6c11/pgo89 (r5c11)
   LUT3:I1->O            2   0.612   0.410  ixor16/Mxor_S_Result<12>11 (N7)
   LUT3:I2->O            3   0.612   0.451  ir5c13/pgo1 (r6c13)
   MUXF5:S->O            2   0.641   0.449  ir4c15/pgo_f5 (r5c15)
   LUT3:I1->O            1   0.612   0.357  qcout/pgo1 (Cout_OBUF)
   OBUF:I->O                 3.169          Cout_OBUF (Cout)
  ----------------------------------------
  Total                     18.354ns (12.260ns logic, 6.094ns route)
                                     (66.8% logic, 33.2% route)
```

Fig.7. Timing Diagram of the Han-Carlson Adder.

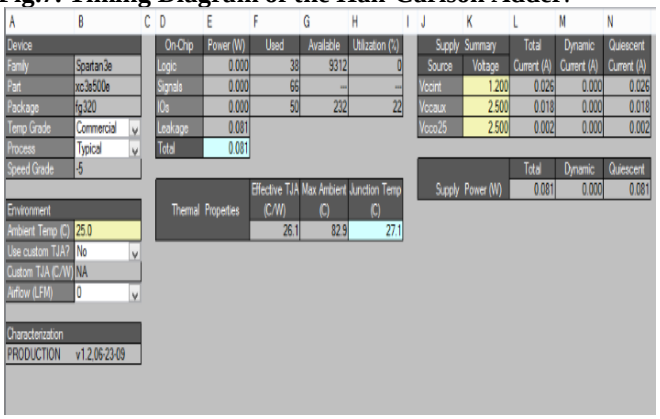

Fig.8. Power Report of the Han-Carlson Adder

The set multiple 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.

VI. CONCLUSION

In this paper a novel variable inertness Han-Carlson parallel prefix speculative adder for fast application is proposed. A new, more accurate, error detection network is introduced, which allows reducing the error probability compared to the previous approaches. A broad arrangement of execution results for 65 nm CMOS technology shows that proposed Han-Carlson variable dormancy adders beats beforehand created variable idleness Kogge-Stone design. compared with traditional, non-speculative, adders, our analysis exhibits that variable inertness Han-Carlson adders show sensible upgrades when the most elevated velocity is required; generally the weight forced by error detection and error correction stages overpowers any advantage. Additional work is required to extend the speculative approach to other parallel-prefix architectures, such as Brent-Kung, Ladner-Fisher, and Knowles.

VII. REFERENCES

[1]I. Koren, com. Arithmetic Arith. Natick, MA, USA: A .K Peters, 2002.

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

[3]R. P. Brent and H. T. Kung, "A regular layout for parallel adders,"IEEE Trans. Compu., 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.Compu., vol. EC-9, pp. 226–231, Jun. 1960.

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