An Ultra-Fast Parallel Prefix Adder

26TH IEEE International Symposium on Computer Arithmetic

Kumar Sambhav Pandey, Hitesh Shrimali, Dinesh Kumar B
School of Computing and Electrical Engineering, Indian Institute of Technology, Mandi

Neeraj Goel
Department of Computer Science & Engineering, Indian Institute of Technology, Ropar

Friday, June 28, 2019

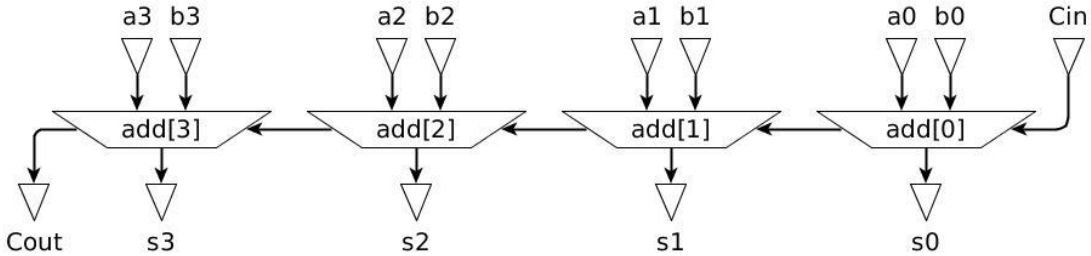
Agenda

- Background
  - Carry Propagation as a Serializing Point
  - Mathematical Preliminaries
  - Weinberger Recurrence and concept of Carry Look-Ahead Adder
  - Ling Recurrence and Ling Adder
  - Dimitrakopoulos Insight
  - Higher Order Factorization and Jackson-Talwar Adders
  - Higher Valency Adders
- Our Proposal
  - Motivation
  - Mathematical Formalism
  - An Instance of Our Adders
- Area-Throughput Characteristics Estimates
- Conclusions
- Discussions

Background

Carry Propagation as a Serializing Point

In the given instance unless the carry out from the least significant bit adder is produced, more significant bits cannot be computed (Carry Ripple Adders).



Can we compute all the sum bits in parrallel

?

Background

Mathematical Preliminaries

Define two signals, carry generate (g_i) and carry propagate (p_i), which being local to each bit position can all be computed in parallel:

$g_i = a_i \cdot b_i$

$p_i = a_i + b_i$

and one ancillary signal carry transmit (t_i):

$t_i = a_i \oplus b_i$

With these signals, it is trivial to compute Carry at bit position i+1 (C_{i+1}) in terms of carry at bit at position i (C_i) as:

$C_{i+1} = g_i + p_i \cdot C_i$

and the sum at bit position i as:

$s_i = t_i \oplus C_i$

Background

Mathematical Preliminaries

The first and the last stages are purely local in nature as they operate on signals only at their respective bit positions. Hence all of the bits can be operated upon concurrently. However, there is a data dependence between C_{i+1} and C_i.

Define a special binary prefix operator ($\circ$) on pairs of operands as:

$$\begin{pmatrix} g_i \\ p_i \end{pmatrix} \circ \begin{pmatrix} g_j \\ p_j \end{pmatrix} = \begin{pmatrix} g_i + p_i \cdot g_j \\ p_i \cdot p_j \end{pmatrix}$$

Thus,

$$\begin{pmatrix} C_{i+1} \\ p_i \end{pmatrix} = \begin{pmatrix} g_j \\ p_i \end{pmatrix} \circ \begin{pmatrix} C_i \\ 1 \end{pmatrix}$$

Background

Mathematical Preliminaries

Obviously, therefore, Input carry at any bit position as a function of C_{in} can thus be trivially computed using a sequence of the prefix operations (∘)s introduced before as:

$$\begin{pmatrix} C_{i+1} \\ p_i \cdot p_{i-1} \cdots p_0 \end{pmatrix} = \begin{pmatrix} g_i \\ p_i \end{pmatrix} \circ \begin{pmatrix} g_{i-1} \\ p_{i-1} \end{pmatrix} \cdots \begin{pmatrix} g_0 \\ p_0 \end{pmatrix} \circ \begin{pmatrix} C_{in} \\ 1 \end{pmatrix}$$

Define group generate signal $g_{i \cdots j}$ and a group propagate signal $p_{i \cdots j}$ as:

$$\begin{pmatrix} g_{i \cdots j} \\ p_{i \cdots j} \end{pmatrix} = \begin{pmatrix} g_i \\ p_i \end{pmatrix} \circ \begin{pmatrix} g_{i-1} \\ p_{i-1} \end{pmatrix} \cdots \begin{pmatrix} g_{j+1} \\ p_{j+1} \end{pmatrix} \circ \begin{pmatrix} g_j \\ p_j \end{pmatrix}$$

Thus,

$$\begin{pmatrix} C_{i+1} \\ p_{i \cdots 0} \end{pmatrix} = \begin{pmatrix} g_{i \cdots 0} \\ p_{i \cdots 0} \end{pmatrix} \circ \begin{pmatrix} C_{in} \\ 1 \end{pmatrix}$$

Background

Mathematical Preliminaries

Two Important Properties:

- Associativity:

$$\left(\left(\begin{pmatrix} g_i \\ p_i \end{pmatrix} \circ \begin{pmatrix} g_j \\ p_j \end{pmatrix}\right) \circ \begin{pmatrix} g_k \\ p_k \end{pmatrix}\right) = \left(\begin{pmatrix} g_i \\ p_i \end{pmatrix} \circ \left(\begin{pmatrix} g_j \\ p_j \end{pmatrix} \circ \begin{pmatrix} g_k \\ p_k \end{pmatrix}\right)\right)$$

- Idempotency:

$$\begin{pmatrix} g_{h\cdots i} \\ p_{h\cdots i} \end{pmatrix} \circ \begin{pmatrix} g_{j\cdots k} \\ p_{j\cdots k} \end{pmatrix} = \begin{pmatrix} g_{h\cdots k} \\ p_{h\cdots k} \end{pmatrix}$$

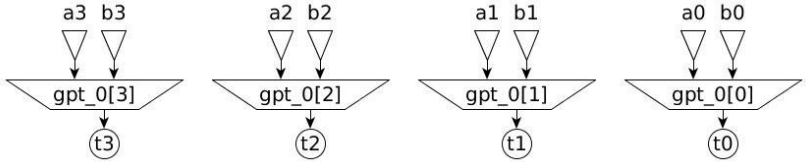
provided, $h > i$, $i \leq j + 1$ and $j > k$

Background

Weinberger Recurrence and Concept of Carry Look-ahead Adders [1]

$\forall i \geq 0$

g_i, p_i and t_i are all computed in parallel.



[1] A. Weinberger and J. Smith, "A logic for high-speed addition," Nat. Bur. Stand. Circ., vol. 591, pp. 3–12, 1958.

Background

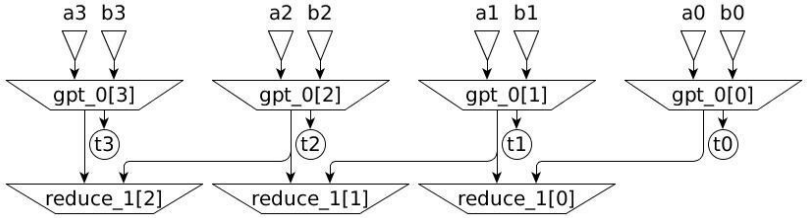
Weinberger Recurrence and Concept of Carry Look-ahead Adders [1]

Following are all computed in parallel:

$$\begin{pmatrix} g_{1\cdots0} \\ p_{1\cdots0} \end{pmatrix} = \begin{pmatrix} g_1 \\ p_1 \end{pmatrix} \circ \begin{pmatrix} g_0 \\ p_0 \end{pmatrix}$$

$$\begin{pmatrix} g_{2\cdots1} \\ p_{2\cdots1} \end{pmatrix} = \begin{pmatrix} g_2 \\ p_2 \end{pmatrix} \circ \begin{pmatrix} g_1 \\ p_1 \end{pmatrix}$$

$$\begin{pmatrix} g_{3\cdots2} \\ p_{3\cdots2} \end{pmatrix} = \begin{pmatrix} g_3 \\ p_3 \end{pmatrix} \circ \begin{pmatrix} g_2 \\ p_2 \end{pmatrix}$$



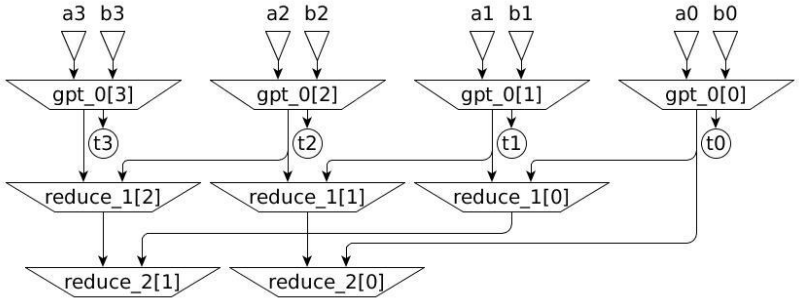
Background

Weinberger Recurrence and Concept of Carry Look-ahead Adders [1]

Following are all computed in parallel:

$$\begin{pmatrix} g_{2\cdots0} \\ p_{2\cdots0} \end{pmatrix} = \begin{pmatrix} g_{2\cdots1} \\ p_{2\cdots1} \end{pmatrix} \circ \begin{pmatrix} g_0 \\ p_0 \end{pmatrix}$$

$$\begin{pmatrix} g_{3\cdots0} \\ p_{3\cdots0} \end{pmatrix} = \begin{pmatrix} g_{3\cdots2} \\ p_{3\cdots2} \end{pmatrix} \circ \begin{pmatrix} g_{1\cdots0} \\ p_{1\cdots0} \end{pmatrix}$$



Background

Weinberger Recurrence and Concept of Carry Look-ahead Adders [1]

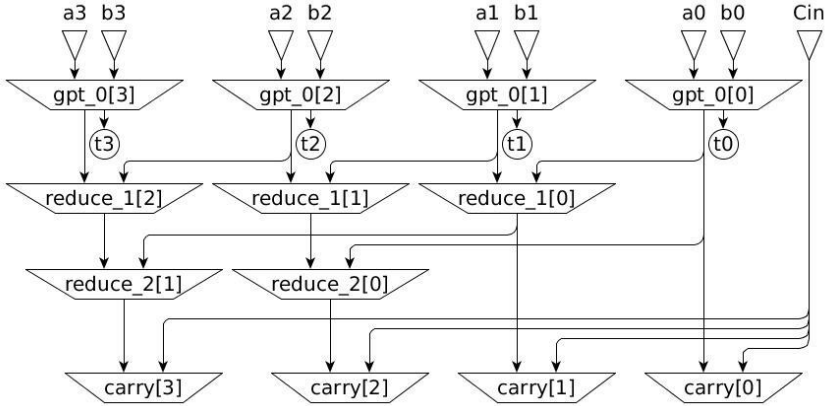
Following are all computed in parallel:

$$\begin{pmatrix} C_1 \\ p_0 \end{pmatrix} = \begin{pmatrix} g_0 \\ p_0 \end{pmatrix} \circ \begin{pmatrix} C_{in} \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} C_2 \\ p_{1\cdots0} \end{pmatrix} = \begin{pmatrix} g_{1\cdots0} \\ p_{1\cdots0} \end{pmatrix} \circ \begin{pmatrix} C_{in} \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} C_3 \\ p_{2\cdots0} \end{pmatrix} = \begin{pmatrix} g_{2\cdots0} \\ p_{2\cdots0} \end{pmatrix} \circ \begin{pmatrix} C_{in} \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} C_4 \\ p_{3\cdots0} \end{pmatrix} = \begin{pmatrix} g_{3\cdots0} \\ p_{3\cdots0} \end{pmatrix} \circ \begin{pmatrix} C_{in} \\ 1 \end{pmatrix}$$

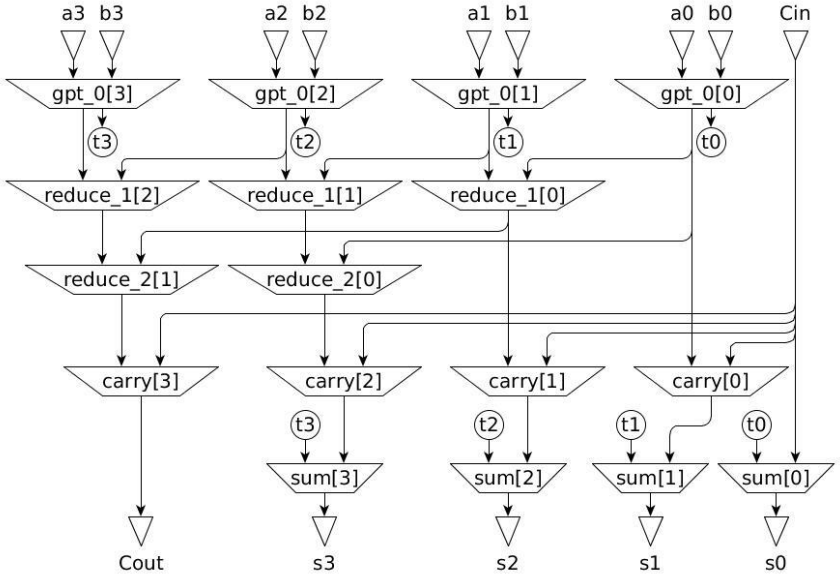


Background

Weinberger Recurrence and Concept of Carry Look-ahead Adders [1]

$\forall i \geq 0$

S_i are all computed in parallel.



Background

Ling Recurrence and Ling Adder [2]

A pseudo carry $H_i = C_i + C_{i-1}$ is propagated in lieu of the conventional carry C_i.

Conventional carries can be extracted from them by $C_i = p_{i-1} \cdot H_i$ as proved below:

$$
\begin{aligned}
& p_{i-1} \cdot H_i \\
&= p_{i-1} \cdot (C_i + C_{i-1}) \\
&= p_{i-1} \cdot (g_{i-1} + p_{i-1} \cdot C_{i-1} + C_{i-1}) \\
&= p_{i-1} \cdot (g_{i-1} + C_{i-1}) \\
&= p_{i-1} \cdot g_{i-1} + p_{i-1} \cdot C_{i-1} \\
&= g_{i-1} + p_{i-1} \cdot C_{i-1} \\
&= C_i
\end{aligned}
$$

[2] H. Ling, "High-speed binary adder," IBM Journal of Research and Development, vol. 25, no. 3, pp. 156–166, March 1981.

Background

Ling Recurrence and Ling Adder [2]

Define two more local signals h_i and q_i as:

$h_i = g_i + g_{i-1}$

and

$q_i = p_i \cdot p_{i-1}$

Starting with the definition of $H_{i+1}$and the fact that $g_i \cdot p_i = g_i$:

H_{i+1}

$= C_{i+1} + C_i$

$= g_i + p_i \cdot C_i + C_i$

$= g_i + C_i$

$= g_i + g_{i-1} + p_{i-1} \cdot (g_{i-2} + p_{i-2} \cdot C_{i-2})$

$= g_i + g_{i-1} + p_{i-1} \cdot p_{i-2}(g_{i-2} + C_{i-2})$

$= h_i + H_{i-1}$

Background

Ling Recurrence and Ling Adder [2]

In terms of the binary prefix operator defined before:

$$\begin{pmatrix} H_{i+1} \\ q_{i-1} \end{pmatrix} = \begin{pmatrix} h_i \\ q_{i-1} \end{pmatrix} \circ \begin{pmatrix} H_{i-1} \\ 1 \end{pmatrix}$$

Ling carry (pseudo carry) at any bit position as a function of $H_{in} = C_{in}$ can thus be trivially computed by a sequence of prefix operations as:

$$\begin{pmatrix} H_{i+1} \\ q_{i-1} \cdot q_{i-3} \cdots q_0 \end{pmatrix} = \begin{pmatrix} h_i \\ q_{i-1} \end{pmatrix} \circ \begin{pmatrix} h_{i-2} \\ q_{i-3} \end{pmatrix} \cdots \begin{pmatrix} H_{in} \\ 1 \end{pmatrix}$$

It is noted that ,

$$H_4 = h_3 + q_2 \cdot h_1 + q_2 \cdot q_0 \cdot H_{in}$$

$$= g_3 + g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot C_{in}$$

which is logically much more simpler than the corresponding expression for the conventional carry C_4:

$$C_4 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_{in}$$

Background

Dimitrakopoulos-Nikolos Insight [3]

Following is an expansion of Ling recurrence relation for Ling pseudo carries for a radix-4 adder:

$$\begin{pmatrix} H_4 \\ q_2 \cdot q_0 \end{pmatrix} = \begin{pmatrix} h_3 \\ q_2 \end{pmatrix} \circ \begin{pmatrix} h_1 \\ q_0 \end{pmatrix} \circ \begin{pmatrix} C_{in} \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} H_3 \\ q_1 \end{pmatrix} = \begin{pmatrix} h_2 \\ q_1 \end{pmatrix} \circ \begin{pmatrix} h_0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} H_2 \\ q_0 \end{pmatrix} = \begin{pmatrix} h_1 \\ q_0 \end{pmatrix} \circ \begin{pmatrix} C_{in} \\ 1 \end{pmatrix}$$

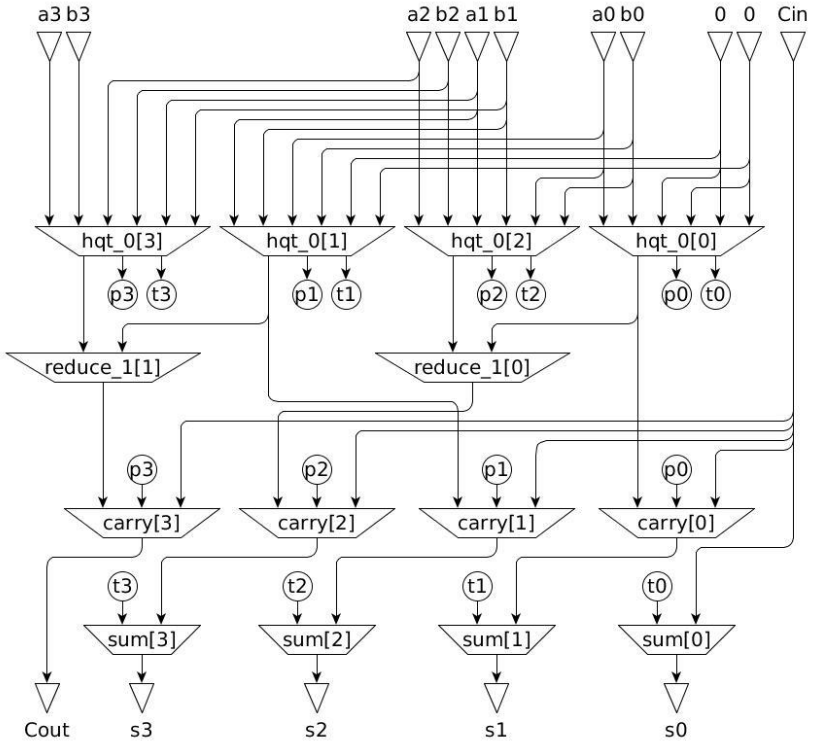
$$\begin{pmatrix} H_1 \\ 1 \end{pmatrix} = \begin{pmatrix} h_0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} H_0 \\ 1 \end{pmatrix} = \begin{pmatrix} C_{in} \\ 1 \end{pmatrix}$$

Dimitrakopoulos and Nikolos observed that in the above expansion the even and odd subscripted pseudo Ling carries are independent of each other.

Background

Dimitrakopoulos-Nikolos Insight [3]



[3] G. Dimitrakopoulos and D. Nikolos, "High-speed parallel prefix VLSI Ling adders," IEEE Transactions on Computers, vol. 54, no. 2, pp. 225–231, Feb 2005.

Background

Higher Order Factorization and Jackson-Talwar Adders [4]

Jackson and Talwar generalized the Ling factorization to further speed up multi-bit addition and proved that the relations for carries can be factorized even beyond what was established by Ling.

For example, the expression for the conventional carry (C_4) can be further factorized as given below:

$$C_4$$
$$= (g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_{in})$$
$$= (p_3) \cdot (g_3 + g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot C_{in})$$
$$= (g_3 + p_3 \cdot p_2) \cdot (g_3 + g_2 + g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot C_{in})$$
$$= (g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot p_1) \cdot (g_3 + g_2 + g_1 + g_0 + p_0 \cdot C_{in})$$

[4] R. Jackson and S. Talwar, "High speed binary addition," in Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004., vol. 2, Nov 2004, pp. 1350–1353 Vol.2.

Background

Higher Order Factorization and Jackson-Talwar Adders [4]

The 2nd equality in the previous slide is the Ling factorization and the quantity in the second factor is actually the Ling pseudo group carry (H_4). The 3rd and the 4th equalities are the higher order factorizations. The quantities in the second factors in these equalities may be called the Jackson-Talwar pseudo carries of order 3 and 4 respectively.

In order to understand the theory define 2 more signals (b_i) and (d_i) as:

$$b_i = g_i + g_{i-1} + g_{i-2} + g_{i-3}$$

$$d_i = g_i + p_i \cdot g_{i-1} + p_i \cdot p_{i-1} \cdot p_{i-2}$$

Define 4th order Jackson-Talwar group carry (J_i) as:

$$J_i = C_i + C_{i-1} + C_{i-2} + C_{i-3}$$

Background

Higher Order Factorization and Jackson-Talwar Adders [4]

Starting with this definition for (J_{i+1}), we can define it in terms of (J_{i-3}) as:

$$
\begin{aligned}
& J_{i+1} \\
&= C_{i+1} + C_i + C_{i-1} + C_{i-2} \\
&= g_i + p_i \cdot C_i + C_i + C_{i-1} + C_{i-2} \\
&= g_i + C_i + C_{i-1} + C_{i-2} \\
&= g_i + g_{i-1} + p_{i-1} \cdot C_{i-1} + C_{i-1} + C_{i-2} \\
&= g_i + g_{i-1} + C_{i-1} + C_{i-2} \\
&= g_i + g_{i-1} + g_{i-2} + p_{i-2} \cdot C_{i-2} + C_{i-2} \\
&= g_i + g_{i-1} + g_{i-2} + C_{i-2} \\
&= g_i + g_{i-1} + g_{i-2} + g_{i-3} + p_{i-3} \cdot C_{i-3} \\
&= b_i + d_{i-3} \cdot J_{i-3}
\end{aligned}
$$

Background

Higher Order Factorization and Jackson-Talwar Adders [4]

Thus the above equation and the definition of (d_{i-3}) can be collected together using the binary prefix operator (°) as:

$$\begin{pmatrix} J_{i+1} \\ d_{i-3} \end{pmatrix} = \begin{pmatrix} b_i \\ d_{i-3} \end{pmatrix} \circ \begin{pmatrix} J_{i-3} \\ 1 \end{pmatrix}$$

Strikingly similar!

Background

Higher Valency Adders

- The discussions in the above subsections are based on the assumption that the generate signals (g_i) and the propagate signals (p_i) are computed for each bit.
- There is no reason why group generate signals ($g_{i\ldots j}$) and group propagate signals ($p_{i\ldots j}$) can not be computed for groups of adjacent bits in place of individual bits.
- These signals defined over such groups can then be reduced in similar treelike structures as discussed earlier.
- Such parallel prefix adders are known as higher valency adders.
- However, a parallel prefix adder with valency 2 is not the same as a Ling adder or that with valency 4 is not the same as a Jackson-Talwar adder which are different and architecturally more efficient.

Our Proposal

Motivation

- In case of Ling adders, the generate signals (g_i) are combined as conjunctions and the propagate signals (p_i) are combined as disjunctions respectively. These combinations are, however, limited to only 2 adjacent signals.
- Arguably Jackson-Talwar adders are motivated by the fact that more than 2 adjacent generate signals can be combined as conjunctions (Reduced Generate) and corresponding propagate signals (Hyper Propagate) were calculated in such a way that the overall addition of multibit integers remains correct.
- In case more than 2 propagate signals are combined as disjunctions and the corresponding generate signals are calculated in similar way to preserve the multi-bit addition semantics, one can create a new family of adders which when looked from the Dimitrakopoulos-Nikolos perspective, can be decoupled in more than 2 subtrees and are consequently faster and more efficient.

Our Proposal

Mathematical Formalism

A pseudo carry $K_i = H_i + q_{i-2} \cdot H_{i-2} = C_i + C_{i-1} + p_{i-2} \cdot p_{i-3} \cdot (C_{i-2} + C_{i-3})$ is propagated in lieu of the conventional carry C_i or pseudo Ling carry H_i.

Conventional carries can be extracted from them by $C_i = p_{i-1} \cdot K_i$ as proved below:

$$p_{i-1} \cdot K_i$$
$$= p_{i-1} \cdot g_{i-1} + p_{i-1} \cdot g_{i-2} + p_{i-1} \cdot q_{i-2} \cdot g_{i-3} + p_{i-1} \cdot q_{i-2} \cdot g_{i-4} + p_{i-1} \cdot q_{i-2} \cdot p_{i-4} \cdot C_{i-4}$$
$$= g_{i-1} + p_{i-1} \cdot g_{i-2} + p_{i-1} \cdot p_{i-2} \cdot g_{i-3} + p_{i-1} \cdot p_{i-2} \cdot p_{i-3} \cdot g_{i-4} + p_{i-1} \cdot p_{i-2} \cdot p_{i-3} \cdot p_{i-4} \cdot C_{i-4}$$
$$= C_i$$

Our Proposal

Mathematical Formalism

Define two more local signals (k_i) and (r_i) as:

$$k_i = g_i + g_{i-1} + q_{i-2} \cdot (g_{i-2} + g_{i-3}) = h_i + q_{i-1} \cdot h_{i-2}$$

and

$$r_i = p_i \cdot p_{i-1} \cdot p_{i-2} \cdot p_{i-3} = q_i \cdot q_{i-2}$$

Starting with the definition of (K_{i+1})we can define it in terms of (K_{i-3}) as:

$$\begin{aligned} & K_{i+1} \\ &= g_i + g_{i-1} + q_{i-1} \cdot g_{i-2} + q_{i-1} \cdot g_{i-3} + q_{i-1} \cdot p_{i-3} \cdot C_{i-3} \\ &= k_i + q_{i-1} \cdot p_{i-3} \cdot C_{i-3} \\ &= k_i + q_{i-1} \cdot p_{i-3} \cdot p_{i-4} \cdot K_{i-3} \\ &= k_i + r_{i-1} \cdot K_{i-3} \end{aligned}$$

Our Proposal
Mathematical Formalism

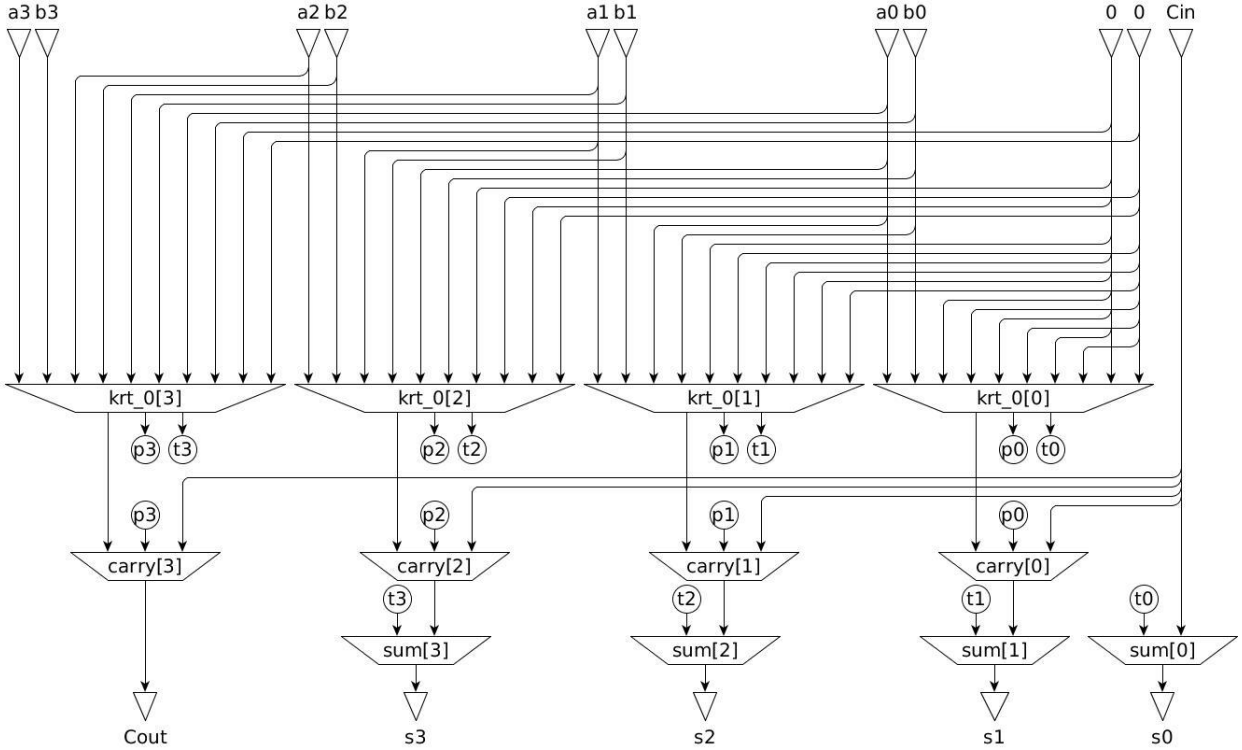
Thus the above equation and the definition of (r_{i-1}) can be collected together using the binary prefix operator (°) as:

$$\binom{K_{i+1}}{r_{i-1}} = \binom{k_i}{r_{i-1}} \circ \binom{K_{i-3}}{1}$$

Strikingly similar!

Our Proposal

An Instance of Our Adder



Area-Throughput Characteristics Estimates

Recurrence Relations	Number of Bits	Logic Levels in Critical Path	Number of Gates
Weinberger-Smith	8	11	107
	16	13	259
	32	15	611
	64	17	1,411
	128	19	3,203

Area-Throughput Characteristics Estimates

Recurrence Relations	Number of Bits	Logic Levels in Critical Path	Number of Gates
Ling	8	10	118
	16	12	274
	32	14	646
	64	16	1,478
	128	18	3,334

Area-Throughput Characteristics Estimates

Recurrence Relations	Number of Bits	Logic Levels in Critical Path	Number of Gates
Jackson-Talwar	8	9	212
	16	11	460
	32	13	1,004
	64	15	2,188
	128	17	5,260

Area-Throughput Characteristics Estimates

Recurrence Relations	Number of Bits	Logic Levels in Critical Path	Number of Gates
Our Proposal	8	9	164
	16	11	364
	32	13	812
	64	15	1,804
	128	17	3,980

Conclusions

- The number of logic levels in the critical path for all the adders based on Ling recurrence are 1 less than the values for the corresponding adders based on Weinberger-Smith recurrence. These levels in case of adders based on Jackson-Talwar recurrence as well as those based on our proposed novel recurrence are still lower by 1, as expected.
- The total gate count for all the adders are increasing from Weinberger-Smith adder to Ling adder to Jackson-Talwar adder.
- The comparison between Jackson-Talwar adder and our proposed adder is particularly interesting. Though the speeds achieved by both the adders is the same, yet the total gate count in case of our proposed adder is much lower as compared to the former.

Discussions

May I answer any of your questions?

Thanks for your Attention!