28

Adders synthesis

Alain Guyot, Mohammed Belrhiti°, Gilles Bosco°*
** INPG/TIMA ° INPG/CSI*
46, avenue Felix Viallet, 38031 Grenoble Cedex, FRANCE
Tel : (33) 76.57.47.34 Fax : (33) 76.50.34.21
E-mail: Alain.Guyot@imag.fr

Abstract

The aim of this paper is not to improve the fastest adder, but instead to give a method to balance complexity and speed.

The paper is organized as follows: the Δ operator introduced by Brent and Kung (Brent-Kung 1982) is first recalled. Then it is used to design "standard" adders (Coren 1993), where the critical path is the longest path from any input of the adder to any output. Several well known adder structural organizations exhibiting different speed/area trade off are described by an unified formalism. By mixing different styles (Guyot 1993), the lowest cost for an imposed delay can be achieved. Then the Δ operators will be replaced with a ROBDDs (Reduced Ordered Binary Decision Diagramms) representation, and we shall see why this representation makes the adders easier to manipulate and to optimize for different technologies (Detjens-Gannot 1993)(Sakouti 1992).

Keywords

Adders, Δ-cells, ROBDDs, mapping.

1 INTRODUCTION

To avoid ambiguity, the (+,*) symbols are used to indicate addition and multiplication, and the ($\wedge$,$\vee$,$\oplus$) for logical AND, OR and XOR. The digit positions are numbered from 0 to n-1, starting from the least significant position at the right of the figures. In an addition, at every position i, the next carry c_{i+1} is either generated, i.e $c_{i+1} = 1$, killed, i.e. $c_{i+1} = 0$ or propagated, i.e $c_{i+1} = c_i$ according to the values of a_i and b_i. Propagation is sometimes called transmission. So three signals can be defined, one for each case: $g_i = a_i \wedge b_i$, $p_i = a_i \oplus b_i$ and $k_i = \overline{a_i} \wedge \overline{b_i}$, but a minimum of only two signals is necessary. It turns out that p_i is the most convenient as explained later, so we retain (p_i , g_i) though (g_i , k_i) is sometimes chosen (Guyot 1993).The combinatorial addition of two numbers $A = \sum_{i=0}^{n-1} a_i * 2^i$ and $B = \sum_{i=0}^{n-1} b_i * 2^i$ can be decomposed in three steps. First (g_i , p_i) are computed $\forall\ i \in \{ 0 .. n-1\}$. From them the carries c_i are obtained. Finally the digits set $s_i = p_i \oplus c_i$ form the result is $S = \sum_{i=0}^{n-1} s_i * 2^i$.

G. Saucier et al. (eds.), *Logic and Architecture Synthesis*

We will assume that there is no carry input, that is $c_0 = 0$, and that we want the carry out c_n. If it were the other way around, we would take the positions from -1 to n-2, g_{-1} being c_0, and then renumber them. If we want both the carry-in and the carry-out, we would just add a position, and remove one if neither is wanted.

2 GROUP PROPAGATE AND GROUP GENERATE

Let us note $\mathbf{P}_i^j$ the group propagate and $\mathbf{G}_i^j$ the group generate, with $n-1 \geq i \geq j \geq 0$.
$\mathbf{P}_i^j$ means that the carry propagates from position j up to position i, that is that c_{i+1} is equal to c_j . $\mathbf{P}_i^j = \prod_{n=i}^{j} p_n$. $\mathbf{G}_i^j$ means that a carry is generated somewhere between j and i and propagated from this location up to position i and yield $c_{i+1} = 1$. $\mathbf{G}_i^j = g_i \vee \Sigma_{n=i}^{j} (\mathbf{P}_i^{n-1} \wedge g_n)$. Note that the expression of $\mathbf{P}_i^j$ is simpler than the one of $\mathbf{G}_i^j$. Clearly we have $\mathbf{P}_i^i = p_i = a_i \oplus b_i$, $\mathbf{G}_i^i = g_i = a_i \wedge b_i$, $\mathbf{P}_i^j \wedge \mathbf{G}_i^j = 0$ and $c_{i+1} = \mathbf{G}_i^0$ (remember $c_0 = 0$). Let us note also $\mathbf{PG}_i^j$ the couple of bits $(\mathbf{P}_i^j, \mathbf{G}_i^j)$. For any k such that $n-1 \geq i \geq k \geq j \geq 0$, $\mathbf{PG}_i^j$ can be computed from $\mathbf{PG}_i^k$ and $\mathbf{PG}_{k-1}^j$ in the following way:
$\mathbf{PG}_i^j = (\mathbf{P}_i^j, \mathbf{G}_i^j) = (\mathbf{P}_i^k \wedge \mathbf{P}_{k-1}^j, \mathbf{G}_i^k \vee \mathbf{P}_i^k \wedge \mathbf{G}_{k-1}^j)$.
We note Δ the operator such that $\mathbf{PG}_i^j = \mathbf{PG}_i^k \ \Delta \ \mathbf{PG}_{k-1}^j$. In the figures the icon for Δ-cells is  .
It is easy to prove the following properties:

- Δ is associative: $\mathbf{PG}_i^j = (\mathbf{PG}_i^k \ \Delta \ \mathbf{PG}_{k-1}^l) \ \Delta \ \mathbf{P}\,\mathbf{G}_{l-1}^j = \mathbf{P}\,\mathbf{G}_i^k \ \Delta \ (\mathbf{PG}_{k-1}^l \ \Delta \ \mathbf{P}\,\mathbf{G}_{l-1}^j)$
- Δ is non commutative $\mathbf{PG}_i^j \neq \mathbf{PG}_{k-1}^j \ \Delta \ \mathbf{P}\,\mathbf{G}_i^k$
- Δ is idempotent $\mathbf{PG}_i^j = \mathbf{PG}_i^k \ \Delta \ \mathbf{PG}_l^j$ with $i \geq l \geq k-1 \geq j$
- Δ is non decreasing.

Any $\mathbf{PG}_i^j$ requires (i-j-1) Δ-cells to be computed from the inputs. Intermediate results from the Δ-cells may be reused, thus reducing the total number of Δ-cells, but increasing the fan-out of some of them.

3 SOME ADDER ORGANIZATIONS

Let us examine now some well-known organizations, their delay given by the number of Δ-cells along the critical path, the cost given by the total number of Δ-cells used, their regularity and their construction. Examples of 32-bit adders are provided.

3.1 Carry ripple adder

The ripple carry adder delay and its cost are in O(n-1). It is inefficient and easily constructed by abutment of Δ-cells.

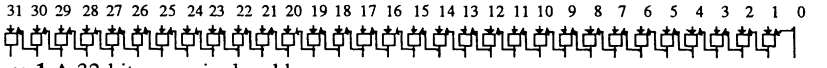

Figure 1 A 32-bit carry ripple adder

3.2 Carry select adder (Sklansky 1960)

The two level carry-select-adder, also named conditional-sum-adder is based on the previous one truncated into blocks of varying sizes. Its cost is in O(2n) and delay $O\lceil\sqrt{2n}\rceil$, more precisely with k Δ-cells along the critical path, an adder can accommodate up to $1 + \sum_{i=1}^{k} i = 1 + \frac{k\,(k+1)}{2}$ bits.

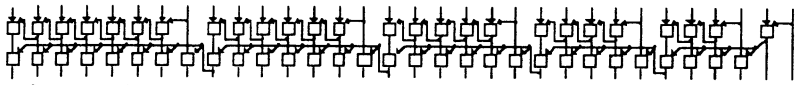

Figure 2 Topological view of a carry select adder

3.3 Brent-Kung adder (Brent-Kung 1982)

The Brent-Kung adder is based on binary Δ-cell trees. The cost is O(2n), the delay $O(2\lceil\log_2(n)\rceil - 2)$.

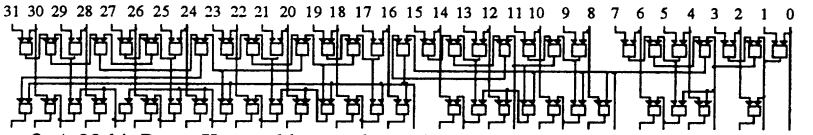

Figure 3 A 32-bit Brent-Kung adder topology view

3.4 Von Neumann adder

The Von Neuman adder has been proved that it is the fastest structural approach (Guyot 1993). The cost of the Von Neumann is $O\lceil\frac{n\log_2(n)}{2}\rceil$, and the delay $O\lceil\log_2(n)\rceil$.

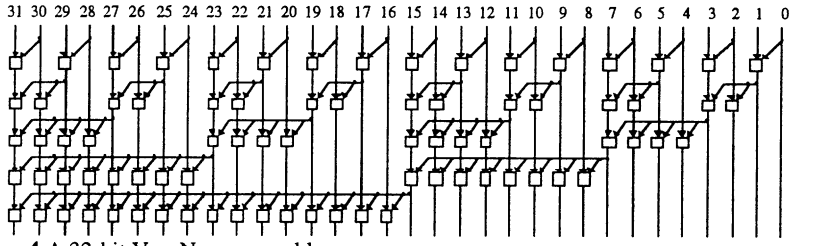

Figure 4 A 32-bit Von Neumann adder

4 STRUCTURAL COMPARISONS

Type of adder	# of Δ-cells	Delay (Δ-cell)	Max. fan-out	32-bit example		
Ripple	n-1	n-1	1	31	31	1
carry-select	$\lceil 2n-\sqrt{2n} \rceil$	$\lceil \sqrt{2n} \rceil$	$\lceil \sqrt{2n} \rceil$	56	8	8
Brent-Kung	$\lceil 2n - \log_2(n)-2 \rceil$	$\lceil 2 \log_2(n) -2 \rceil$	$\lceil \log_2(n) \rceil$	57	8	5
Von Neumann	$\lceil n/2 \log_2(n) \rceil$	$\lceil \log_2(n) \rceil$	n/2	80	5	16

5 FROM Δ-CELLS TO ROBDDS

5.1 Adder Generator

Let us try to find a good representation of the Δ-cells to make them easy to handle. The aim is to give a solution to build quickly the different adders organizations and make them mappable. We shall use the Reduced Ordered Binary Decision Diagrams (Bryant 1986), because they give the liberty to manipulate easily the Xor operations that are very present in the adders.
as we have mentioned in §1 :

$\mathbf{PG}_i^j = (\mathbf{P}_i^j, \mathbf{G}_i^j) = (\mathbf{P}_i^k \wedge \mathbf{P}_{k-1}^j, \mathbf{G}_i^k \vee \mathbf{P}_i^k \wedge \mathbf{G}_{k-1}^j)$.

so: $\mathbf{P}_i^j = (a_i \oplus b_i) \wedge \mathbf{P}_{i-1}^j$

& $\mathbf{G}_i^j = \mathbf{G}_i^i \vee \mathbf{P}_i^i \wedge \mathbf{G}_{i-1}^j = (a_i \wedge b_i) \vee ((a_i \oplus b_i) \wedge \mathbf{G}_{i-1}^j)$

$\mathbf{G}_i^i = a_i \wedge b_i$

$\mathbf{G}_i^j = a_i \wedge (b_i \vee (\overline{b_i} \wedge \mathbf{G}_{i-1}^j)) \vee \overline{a_i} \wedge (\overline{b_i} \wedge \mathbf{G}_{i-1}^j)$

We note a_i Ai and b_i Bi in the following figures. That will provide the following generators :

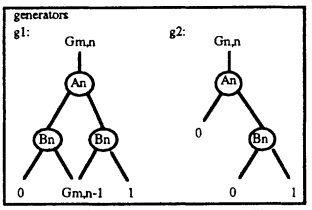

Figure 4 Generators for G

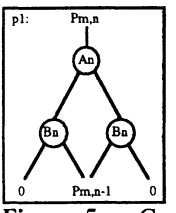

Figure 5 Generator for P

Decomposition of $\mathbf{PG}_i^j$ for k such that $n-1 \geq i \geq k \geq j \geq 0$.
We have got the formula: $\mathbf{PG}_i^j = (\mathbf{P}_i^j, \mathbf{G}_i^j) = (\mathbf{P}_i^k \wedge \mathbf{P}_{k-1}^j, \mathbf{G}_i^k \vee \mathbf{P}_i^k \wedge \mathbf{G}_{k-1}^j)$.

Let us consider we have the carry-ripple-adder ROBDDs corresponding to $(\mathbf{P}_i^j, \mathbf{G}_i^j)$; and let us see how we can easily compute $\mathbf{PG}_i^j$ from $\mathbf{PG}_i^k$ and $\mathbf{PG}_{k-1}^j$. The ROBDDs corresponding to $\mathbf{PG}_i^k$ and $\mathbf{PG}_{k-1}^j$ will be obtain by cutting the ROBDDs of $\mathbf{PG}_i^j$.

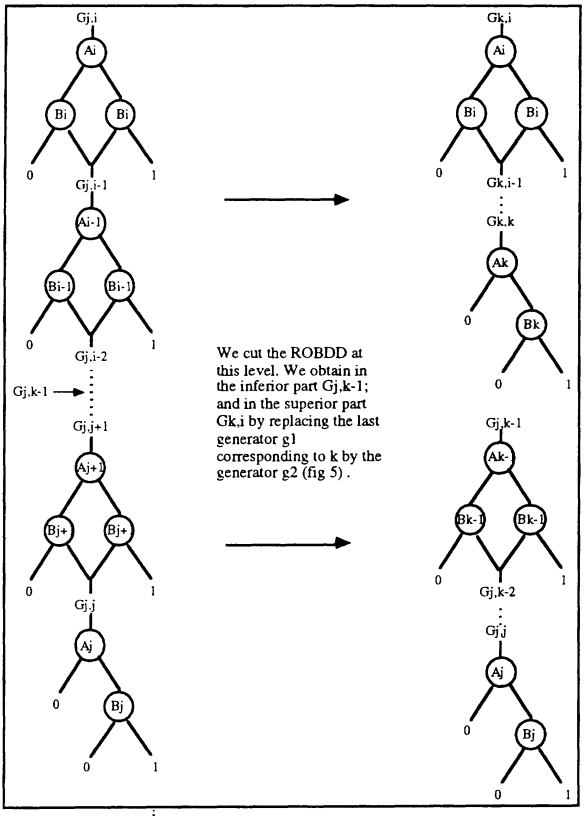

Figure 7 decomposition of $\mathbf{PG}_i^j$.

For the ROBDD corresponding to P, the same operation will be realised.
Next, to compute the value of $\mathbf{PG}_i^j$, we have got to build the two ROBBDs corresponding to the two equations: $\mathbf{G}_i^j = \mathbf{G}_i^k \vee \mathbf{P}_i^k \wedge \mathbf{G}_{k-1}^j$

$\mathbf{P}_i^j = \mathbf{P}_i^k \wedge \mathbf{P}_{k-1}^j$

by considering the values of $\mathbf{PG}_{k-1}^j$ and $\mathbf{PG}_i^k$ as primary inputs, we can compute $\mathbf{PG}_i^j$ as follow:

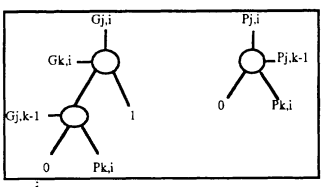

Figure 8 computation of $\mathbf{PG}_i^j$

5.2 Mapping

By replacing the Δ-cells by ROBDDs. we can easily evaluate the more suitable adder for a technology because we can control better the number of entries and the fan-out of the nodes of the BDDs.

5.2 Advantages of this method

- representation very closed to the Δ-cells.
- linear size of representation for the adder.
- easiness to build and modify the adders.
- two possibilities for the mapping: directly on the ROBDDs (Brace-Rudel-Bryant 1990) (by using greedy algorithms) or by turning ROBDDs into a function (mapping algorithms on trees, tree matching (Lisanke-Bergly-Kedem 1988), dynamic programmation (Kentzer 1988)).

6 EXPERIMENTAL RESULTS

The following results correspond to tests realised on Xilinx 4000. We gives the result after placement and routage performed by ppr (Xilinx placer and router).

Table 1 Von Neumann adders:

Size	Area (#Clbs)	Speed (ns)	level (#Clbs)
4	2	33,8	3
8	9	41,0	4
32	25	52,4	5
32	58	60,9	6

Table 2 Comparison between different structures for a 32 bits adder:

	2-levels carry-select adder	carry-ripple	Von Neumann
Area (#Clbs)	54	47	58
Speed (ns)	73,8	208,2	60,9
Level (#Clbs)	9	32	6

Table 3 Comparison between different structures for a 16 bits adder:

	2-levels carry-select adder	carry-ripple	Von Neumann
Area (#Clbs)	24	22	25
Speed (ns)	65,9	113,9	52,4
Level (#Clbs)	7	16	5

7 CONCLUSION

First, we have a description of the adders by a unified formalism, and the modelisation allows the adders good regularity and flexibility. Then we can control the granularity of the adders by collapsing or clustering (Touati-Sajov-Brayton 1991) to implement it on a given technology.

8 REFERENCES

R.P. Brent and H.T. Kung, *A regular layout for parallel adders*, IEEE Trans on Computers, Vol.C31, pages 260-264, Mars 1982.

Israel Koren, *Computer Arithmetic Algorithms*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

Alain Guyot, *Computer Arithmetic*, E.N.S.I.M.A.G. 1993

E. Detjens, G. Gannot, *Tehcnology Mapping in MIS*, in Proc. Int. Conference on Computer Aided Design (ICCAD 1987), pp 116-119, Santa Clara, USA, Nov 1987.

J. Sklansky, *Conditional Sum Addition Logic*, IRE Trans. EC-9(2) (June 1960), pp226-231.

R.E Bryant, *Graph Based Algorithms for Boolean Functions Manipulation*, IEEE Trans on Computers, Vol C35, pages 667-692, August 1986.

K. Brace, R.Rudell, R.Bryant, *Efficient Implementation of BDD Package*, 27th DAC, June 1990.

R. Lisanke, F. Bergly, G. Kedem, *McMap : A Fast Technology Mapping Procedure for Multi-Level Logic Synthesis*, ICCD 1988, pp 252-256, Oct. 1988.

K. Kentzer, *Timing Optimisation in a Logic Synthesis System*, International workshop on logic and architecture synthesis for silicon compiler, Grenoble 1988.

H.J. Touati, H.Sajov, R.Brayton, *Delay Optimisation of Combitional Logic Circuit by Clustering and Partial Collapsing*, in Proc. Int. Conference on Computer Aided Design ICCAD 1991).

K. Sakouti, *Synthèse et Optimisation Temporelle de Réseaux Booléens*, PhD, Institut National Plolytechnoque de Grenoble, Dec 1992.