ADDER SYNTHESIS

Martin Horauer, Dietmar Loy

University of Technology Vienna
Institute of Computer Technology
Gußhausstr. 27-29, A-1040 Vienna
{horauer, loy}@ict.tuwien.ac.at

Abstract

This paper describes the logic synthesis of several 91 bit wide adders. Their performance is compared before and after synthesis to tail out the power of a synthesis tool and to show the improvement gained due to a more sophisticated structure. Therefore a brief description of the investigated adder architectures is given.

Keywords: *Carry lookahead adder, carry select adder, high-speed CMOS binary adder, spanning tree carry lookahead adder*

Introduction

The project SynUTCSU[1] is devoted to the problem of how to establish a common notion of time that also relates within a few μs to an external time standard, in particular, *universal time coordinated* (UTC), in a distributed fault-tolerant (real-time) system. One mainstream related to this work is the development of an ASIC (UTCSU) supporting the clock synchronization by hardware. The synchronization algorithm benefits from a large adder (91 bit), which provides the time compatible to the network time protocol (NTP) format with a very fine resolution to allow corrections with high granularity. Hence several architectures were inspected to build an adder fast enough to provide a common notion of time to satisfy our goals.

Investigating adder architectures

Classical high speed adders include carry-lookahead [1], carry-skip [2], carry-select [3] and conditional sum [4] adders. Brent and Kung [5] developed a binary variant of the carry-lookahead adder, the binary carry-lookahead adder, which was improved and modified by Schwartzlander and Lynch to the so called spanning-tree carry-lookahead adder. Another variant, based on reformulating the carry equations was developed by Ling [6]. Other fast implementations are illustrated by Hwang and Fisher [8] in CMOS technology using multiple domino logic and by Flynn and Quach [9] by modifying the Ling adder.

Among the huge member of fast adders we wrote VHDL code for a carry-ripple, a carry-select, a carry-lookahead and a modified spanning-tree adder to have a widespread input into our synthesis tool. In the following subsections we give a brief description of the choosen architectures.

Carry-Ripple Adder

[1] Supported by the Austrian Science Foundation (FWF), grant no. P10244-ÖPH

The wellknown carry ripple adder, also denoted as ripple through carry adder, consists of n cascaded full adders for a n bit adder. The sum of each stage is calculated by $S_i=(A_i \oplus B_i) \oplus C_{i-1}$ and the carry by $C_i=S_i \vee (A_i \& B_i)$.

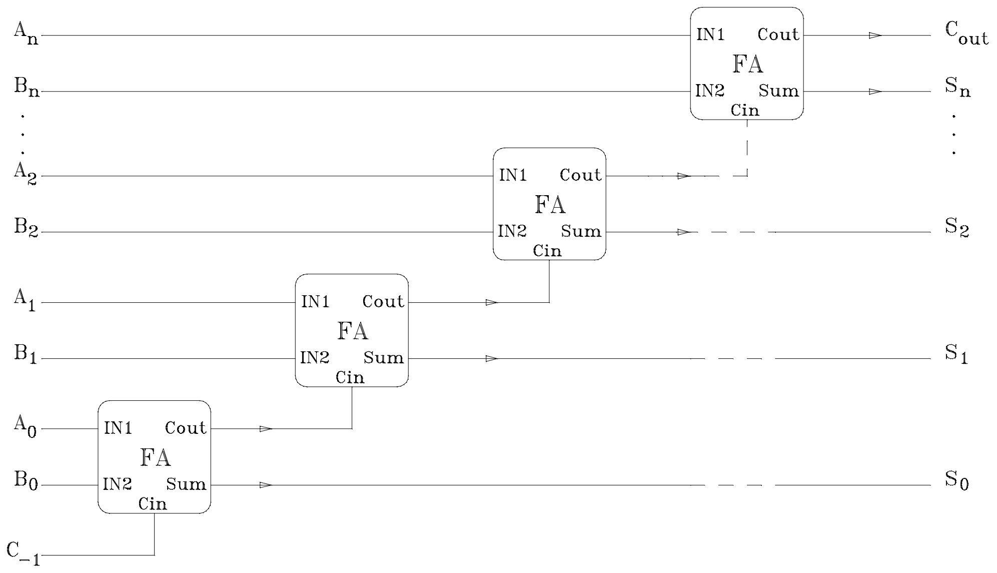

Figure 1: n-bit Carry-Ripple Adder

The theoretical value of area and delay are given by $\sim O(n)$.

Carry-Select Adder

A carry-select adder (CSA) is divided into multiple sections. Each of these performs two additions in parallel, one assuming a carry-in of zero and the other with a carry-in of one. Figure 2 illustrates a brief example of one section. In the first bit-position (a_i,b_i), two full-adders add the inputs assuming a carry-in of zero and one. The two sum signals are then multiplexed by the carry-out from the previous section k. Both carry-out signals of the two full-adders are fed to the next bit position, where they are used as new carry-in's. The carry-out signals in the last bit position are then mulitplexed to the carry-out of the current section l which serves as carry-in for section m.

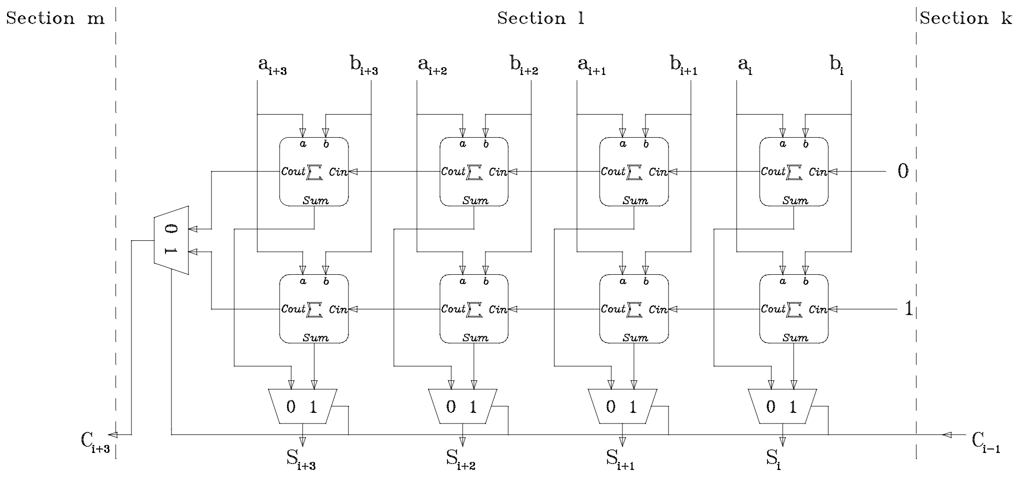

Figure 2: One section in a Carry-Select Adder structure

Carry Lookahead Adder

Carry-lookahead (CLA) technique is used to speed up the carry propagation in an adder complex. Hence the carries entering all the bit positions of a "parallel" adder are generated simultaneously by additional logic circuitry. This results in a constant addition delay independent of the length of the adder.

Let $A=A_{n-1}...A_1A_0$ and $B=B_{m-1}...B_1B_0$ be the augend and addend inputs to an n-bit adder. C_{i-1} is the carry input to the i^{th} position. The carry in to the least significant position is denoted as C_{-1}. Let *SUM* and C_i be the sum and carry outputs of the i^{th} stage. Then we can define auxiliary functions G_i, P_i and *SUM* as follows:

$$G_i=A_i \mathbin{\&} B_i$$

$$P_i=A_i \vee B_i$$

$$SUM=A_i \oplus B_i \oplus C_i$$

The functions G_i and P_i serve as inputs to the CLA blocks which compute the carry while *SUM* represents the result of the addition. The CLA blocks compute the carry and the functions GG_i and GP_i which serve as inputs for the next stage.

C_{-1} is the input to the last CLA block. This signal and the GP_i and GG_i functions are used to compute the carry-out of the adder C_{n-1} and the carry in C_i for the previous stage. The implied functions GP_i, GG_i and C_i are computed recursively as follows:

$$GP_i=GP_{i-1} \mathbin{\&} P_i$$

$$GG_i=(GG_{i-1} \mathbin{\&} P_i) \vee G_i$$

$$C_i=(C_{i-1} \mathbin{\&} P_{i-1}) \vee G_{i-1}$$

($i = 1,2,3, .. g$; $GP_0=P_0$ and $GG_0= G_0$; g ...number of PG blocks)
The number of CLA levels used depends on the size of the adder, the required speed and the available die size. In practice CLA does not pay-off for adders of length $n\leq 4$, because short ripple-carry adders would be equally fast. For adders of length, say $n\geq 16$, multilevel carry- lookahead may be needed. Figure 3 illustrates a n=16 bit adder with a two level CLA logic consisting of 4 bit CLA blocks.

The theoretical area is given by

$$6n+1+\frac{p+1}{p-1}q+\frac{p^2+2p-1}{p}\left[k\left| n+\frac{1}{p-1} \right| - \frac{pq}{(p-1)^2} \right|$$

and for the delay by

$$4+k(p+1)$$

in terms of required two-input-gates. Where *n* is the adder word length, *p* the carry span in the CLA adder and *k* a factor calculated with the following equation:

$$k=\log_p\left[1+n(p-1)\right]-1 .$$

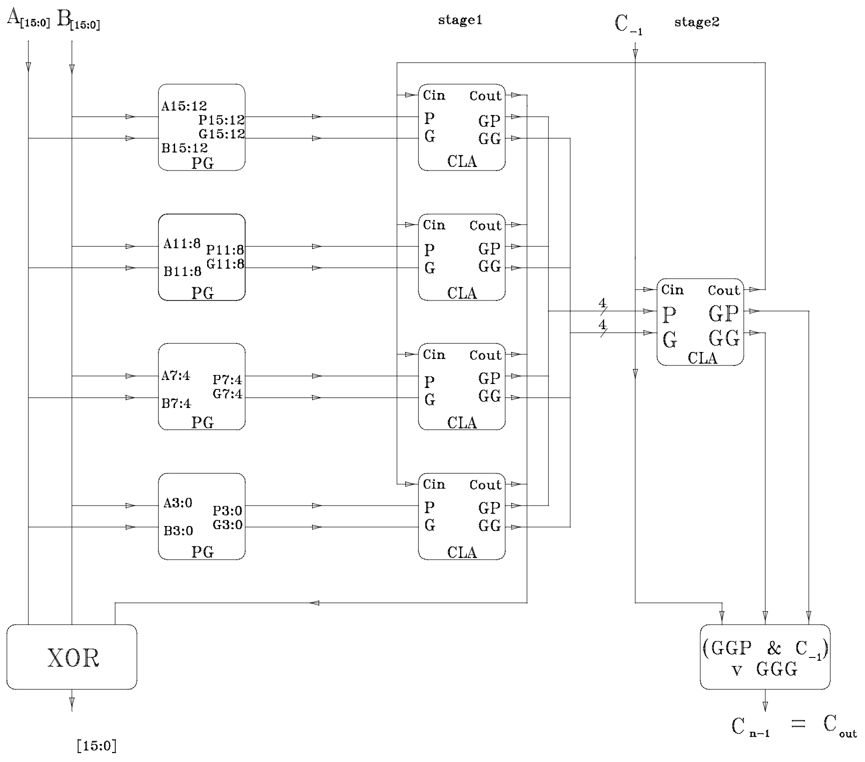

Figure 3: Carry-Lookahead Adder *(n=16, p=4, k=4)*

Spanning-Tree Carry-Lookahead Adder

The spanning-tree adder consists of a binary lookahead carry-tree and a carry-select section. The binary tree differs from the carry-lookahead tree because the carry-in logic in back propagation has been removed. The intermediate carries produced by the binary lookahead carry tree are fed to the multiplexers of the carry-select section. This is illustrated in figure 4.

Similar to the carry-lookahead adder, propagate P_i and generate G_i are computed first. Using these functions the first block generates GP_0,GG_0 and C^* using the following equations:

$$G_i=A_i \,\&\, B_i$$

$$P_j=A_i \vee B_i$$

Figure 4: *4n*-bit Spanning Tree Adder

Adder performance (delay,area)

The four adder architectures were synthesized with Synopsys V3.2a for 0,7µm ES2 standard cell design. Two different optimisation constraints are used. First the delay was denoted as critical and the second with mimimum area constraint. Table 1 gives an overview of the direct instantiated adders, table 2 shows the results after synthesis for minimum area and table 3 the results of synthesis for minimum delay, which was our major interest. The choosen architectures are primary optimized structures for speed and not for area.

Adder	Area [mm^2]	Delay [ns]	VHDL architecture
CRA	0.177	118.8	structural
CSA	0.454	54.93	structural

Table 1: Direct instantiated adder

Adder	Area [mm^2]	Delay [ns]	VHDL architecture
CRA	0.177	118.8	structural
CSA	0.281	158.5	structural
CLA-4	0.299	138.9	behavioral
CLA-7	0.276	193.0	behavioral
SPTA			behavioral

Table 2: Optimized adder with constraint min. area

Adder	Area [mm^2]	Delay [ns]	VHDL architecture
CRA	0.589	12.89	structural
CSA	0.607	12.49	structural
CLA-4	0.547	12.05	behavioral
CLA-7	0.676	11.66	behavioral
SPTA			behavioral

Table 3: Optimized adder with constraint min. delay

Summary and Conclusion

In this paper four different adders were described. They were implemented with VHDL code and compared against each other before and after synthesis. The results show that nowadays synthesis nearly overrules improved descriptions of fast adders. Although synthesis can be very satisfactory careful implementation and use of the synthesis tools is nessesary.

References

[1] Hwang, Kai: "Computer Arithmetic - Principles, Architecture and Design", John Wiley and Sons Inc. USA,1979

[2] Majerski, S: "On Determination of Optimal Distributions of Carry Skips in Adders", IEEE Trans. EC-16 No.1, Feb. 1967

[3] Bedrij, O.J.: "Carry-Select Adders", IRE Trans. EC-11 No. 3, June 1962

[4] Sklansky,J: "Conditional Sum Addition Logic", IRE Trans. EC-9 No. 2, June 1960

[5] R.P. Brent and H.T. Kung: "A regular layout for parallel adders", IEEE Trans. Comput. C-31, 1982

[6] H. Ling: "High-speed binary adder", IBM J. Res.Develop., vol. 25, May 1981

[7] T. Lynch and E. Swartzlander jr.: "A Spanning Tree Carry Lookahead Adder", IEEE Trans. Comput., vol. 41 No. 8, Aug. 1992

[8] I.S. Hwang and A.L. Fisher: "A 3.2 ns 32-bit CMOS adder in multiple output domino logic", IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers, 1988

[9] N.T. Quach and M.J. Flynn: "High Speed Addition in CMOS", IEEE Trans. Comput., vol. 41 No. 12, Dec. 1992