LOW-POWER RADIX-4 DIVIDER

Alberto Nannarelli and Tomas Lang

Department of Electrical & Computer Engineering
University of California, Irvine, California 92717

e-mail : alberto@ece.uci.edu, tomas@ece.uci.edu

Abstract

The general objective of our work is to develop methods to reduce the power consumption of arithmetic modules, while maintaining the delay unchanged and keeping the increase in the area to a minimum. Here we illustrate some techniques for a radix-4 divider realized in $0.6\mu m$ CMOS technology. Using techniques such as switching-off not active blocks, retiming the recurrence, equalizing the paths to reduce glitches, using gates with lower drive capability, and changing the redundant representation, we obtained a power consumption reduction of 35% with respect to the standard implementation. The techniques used here should be applicable to a variety of arithmetic modules which have similar characteristics.

1 Introduction

The general objective of our work is to develop methods to reduce the power consumption of arithmetic modules. We attempt to reduce the power while maintaining the delay unchanged and keeping the increase in the area to a minimum. Since the dynamic power dissipation in CMOS cells is proportional to the switching frequency and to the output load [1], we reduce the number of transitions and the capacitance (by using lower-drive cells when available).

The implementations of the radix-4 divider were done using the COMPASS design environment [2] and the Passport $0.6\mu m$ standard cell library [3]. The structural model was obtained by manually decomposing the behavioral model into functional blocks. Some of those blocks were synthesized by COMPASS from behavioral models to gate networks, others were manually implemented with gates. The power estimation has been carried out with PET [4], a power evaluation tool which computes the power dissipated in a circuit from the netlist extracted from the layout, the standard cell library characteristics, and the results of a logic-level simulation run on a suitable number of test vectors.

Using techniques such as switching-off not active blocks, retiming the recurrence, equalizing the paths to reduce glitches, using gates with lower drive capability, and changing the redundant representation, we obtained a power consumption reduction of 35% with respect to the standard implementation.

ISLPED 96 - International Symposium on Low Power Electronics and Design, Monterey, California.
August 12-14, 1996

2 Basic Design

In this paper we consider a double precision radix-4 division unit [5], which is typical of those found in many floating-point processors. The recurrence is

$$w[j+1] = 4w[j] - q_{j+1}d \qquad j = 0, 1, \ldots 28$$

with the initial value $w[0] = x$ and with the quotient-digit selection

$$q_{j+1} = SEL(w[j]_7, d_3) \qquad q_j = \{-2, -1, 0, 1, 2\}$$

where x the dividend, d the divisor, q_{j+1} the quotient digit at the j-th iteration and d_3 and $w[j]_7$ are the divisor and the residual (carry-save representation) truncated after the 3rd and 7th fractional bit respectively.

We begin with the standard block diagram shown in Figure 1. The recurrence is implemented with the selection function, the multiple generator, the carry-save adder and two registers to store the carry-save representation of the residual W. The conversion block, whose low-power optimization is not treated in this paper, performs the conversion from the signed-digit quotient and performs the rounding using the sign and zero from the carry propagate adder (CPA). The datapath shown in Figure 1 is completed by a controller and by a tree to distribute the clock signal (not depicted in the figure). The corresponding implementation, optimized for minimum delay, has the power dissipation

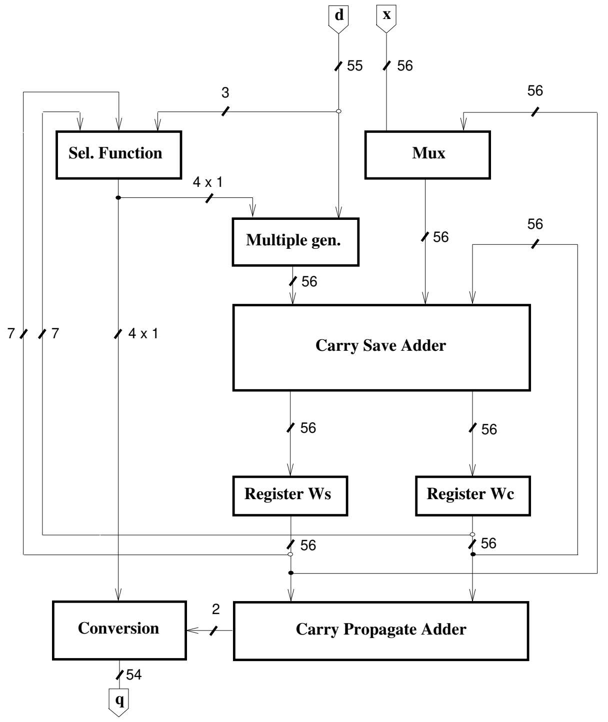

Figure 1: Block diagram of *div0* unit

characteristics shown for *div0* in Table 3 in the last section.

3 Power Consumption Reduction

Starting from the above implementation, we reduced the power dissipated in the unit using the following techniques:

3.1 Switching-off not active blocks

The first technique consists in switching-off blocks which are not active during several cycles. This is the case for the CPA, which is only used at the end to determine the sign of the last residual. This results in implementation *div1*. The power reduction is 10%.

3.2 Retiming the recurrence

By retiming the recurrence we reduce the number of spurious transitions and take out components from the critical path. This is achieved in the divider case by moving the selection function from the first part of the cycle to the last part of the previous cycle (Figure 2). The resulting block diagram for implementation *div2* is shown in Figure 3 (modifications are marked with

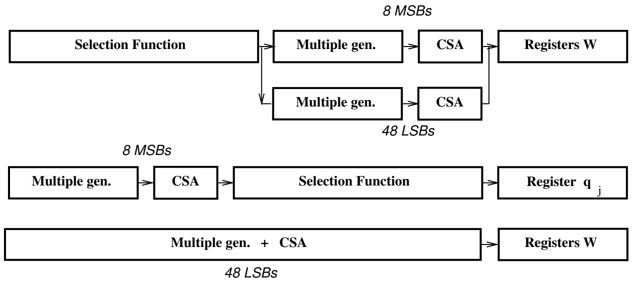

Figure 2: Retiming of recurrence: critical path of original (above) and retimed (below) scheme

dotted lines). In this modified divider, the controller selects the input x of the multiplexer during the first cycle and the input d during the rest. In the first cycle, the controller also resets register q_j to 1, allowing the input x to be stored in registers W as the first residual $w[0]$.

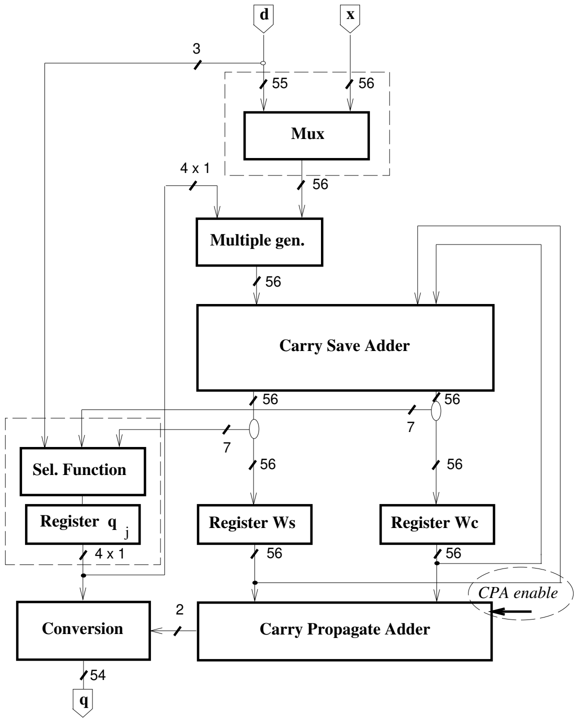

Figure 3: Block diagram of *div2* unit

Since now the quotient digit is stored in a register, this has the effect of reducing the glitches in the multiple generator and in the carry-save adder. Also the multiplexer is now out of the recurrence and this reduces the power dissipated in it. The reduction in the number of transitions is 22% and the reduction in

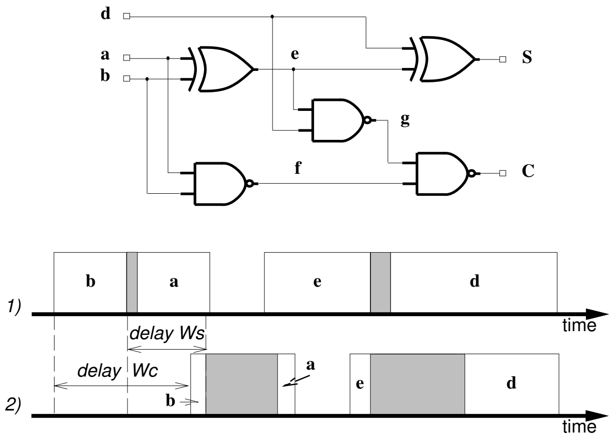

Figure 4: Equalizing paths in CSA

power is 16% (with respect to *div1*).

Moreover, now the critical path is limited to the eight most significant bits, so that the forty-eight least significant can be redesigned for lower power dissipation.

3.3 Equalizing the paths to reduce glitches

Equalizing the paths of the input signals of the blocks we reduce the generation of glitches. Because of different delays, both gate and interconnection delay, the input signals to the carry-save adder arrive at different times, creating spurious transitions inside the full adders. Figure 4 shows, in the upper part, the implementation of one of the full-adders composing the carry-save adder. If the input signals a and b arrive at different times a glitch might be produced in e and f. Also, if there is a difference between the arrival times of d and e, glitches might be produced in S, g and C.

Time diagram 1) in Figure 4 shows the distribution of the arrival times for signal a, b, d, and e. The boxes indicates the range of the arrival times for the 56 bit-slices. This range is produced by the delay in the interconnections; moreover, notice that although a and b both come directly from the registers, a arrives later than b, again because of interconnection delays. In order to eliminate the spurious transitions, we delay the clock to the Ws and Wc registers (which produce a and b) so that the ranges of a and b as well as those of e and d overlap, as shown in time diagram 2) in Figure 4. Anyway, because the regions don't overlap completely, it is impossible to eliminate all the glitches. Moreover, due to the different delays of the XOR and NAND gates, signals at nodes f and g always arrive at different times, and spurious transitions eventually occurring at C, cannot be eliminated either.

Table 1 shows the number of transitions per cycle in the carry-save adder for the two implementations *div2* and *div3* (where the equalization of paths is applied). With *zero-delay* transitions we mean the number of transitions in the stationary state, simulating the circuit without delays. Note that in this circuit the number of spurious transitions is a relatively small fraction of the total. The reduction in the number of spurious transitions is 20%.

	zero-delay	actual	spurious	reduction
div2	140	190	50	-
div3	140	180	40	10

Table 1: Transitions per cycle in CSA

3.4 Using gates with lower drive capability

Another reduction in the power dissipation is achieved by minimizing the power in the gates not in the critical path by using cells with lower drive capability. In the retimed recurrence, this can be done for the forty-eight least-significant bits of the multiple generator and the carry-save adder. The combination of the two last two techniques results in implementation *div3*. The reduction in power consumption is 5% and it is very dependent on the characteristics of the library of standard cells used. We estimate that, if lower-drive gates were available for all the cells not in the critical path, the power reduction would have been 9%.

3.5 Changing the redundant representation to reduce the number of flip-flops

Since the contribution of flip-flops to both power dissipation and area is significant, we changed the redundant representation of W to reduce the number of flip-flops in the registers. We use a radix-4 carry-save representation with two sum and one carry flip-flops for each two bits (Figure 5). Since this requires a redesign of the carry-save adder to propagate the carry of the even bit-slice to the next bit-slice, in order not to increase the critical path this is done only in the 48 least-significant bits of W, which in the retimed recurrence are not in the critical path. Moreover, we equalized again the paths of the signals arriving to the carry-save adder because of the difference introduced by the radix-4 representation for odd and even bits. This results in implementation *div4* and in the block diagram of Figure 6. The additional power reduction is 9%, and the total reduction with respect to the basic implementation is 35%.

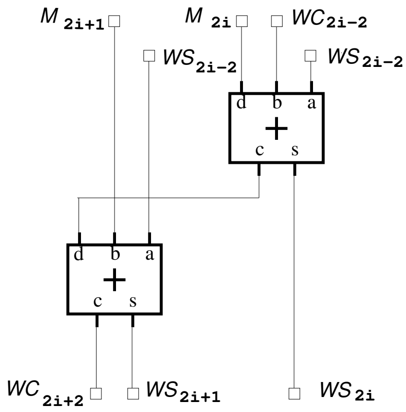

Figure 5: Radix-4 implementation in the carry-save adder

4 Summary of reductions

Table 2 shows the reduction in the number of transitions (thousands) for the execution of a division (30 clock cycles) in the different implementations. It also shows the number of transistors (thousands) and area of the dividers.

	div0	div1	div2	div3	div4
0-delay transitions	17.1	13.6	11.9	11.9	9.1
actual transitions	26.4	21.6	16.8	16.6	13.9
no. of transistors	21.4	22.1	22.1	22.0	20.1
Area (mm^2)	1.58	1.66	1.68	1.63	1.51

Table 2: Transitions and Area

Table 3 summarizes the result obtained in the low-power optimization of the divider. Each column represents a different implementation.

blocks	div0	div1	div2	div3	div4
control	0.6	0.6	0.6	0.6	0.6
clk tree	1.9	1.9	1.9	1.9	2.1
mux	0.7	0.7	0.1	0.1	0.1
mul. gen.	2.9	2.9	1.7	1.5	1.5
CSA	5.2	5.2	3.5	2.8	2.9
sel. func.	0.7	0.7	1.3	1.3	1.3
register ws	2.6	2.6	2.6	2.6	2.6
register wc	2.3	2.3	2.3	2.3	1.0
register q	-	-	0.1	0.1	0.1
CPA	2.5	0.6	0.6	0.6	0.5
Total	19.4	17.5	14.7	13.8	12.7
Ratio	1.00	0.90	0.76	0.72	0.65

Table 3: Power dissipation in mW @ 10 MHz

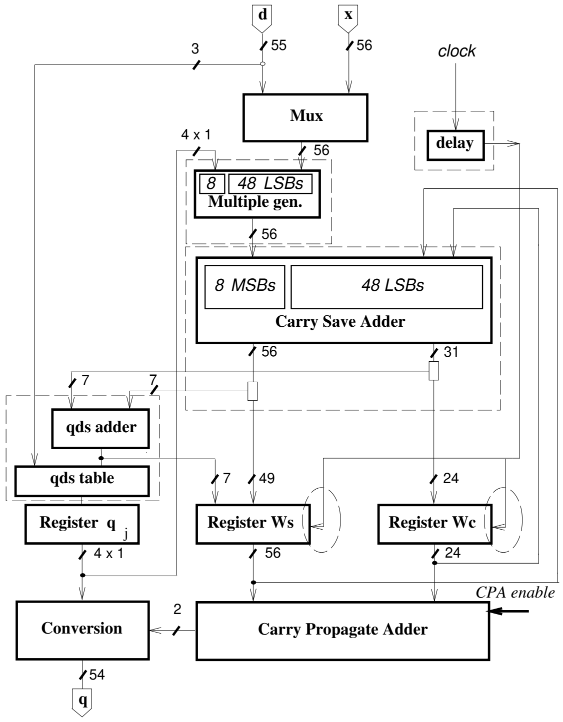

Figure 6: Block diagram of *div4* unit

From Table 3 we conclude that the above mentioned transformations result in an overall power reduction of 35%. The techniques used here should be applicable to a variety of arithmetic modules which have similar characteristics.

References

[1] N. H. E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design*. Addison-Wesley Publishing Company, 2nd edition, 1993.

[2] Compass Design Automation. *User Manuals for COMPASS VLSI*. Compass Design Automation, Inc., 1992.

[3] Compass Design Automation. *Passport - 0.6-Micron, 3-Volt, High-Performance Standard Cell Library*. Compass Design Automation, Inc., 1994.

[4] A. Nannarelli. Power evaluation tool: modeling and implementation. *COMPASS User's Group Conference*, Apr 1996.

[5] M.D. Ergegovac and T. Lang. *Division and Square Root: Digit-Recurrence Algorithms and Implementations*. Kluwer Academic Publisher, 1994.