IMPLEMENTATION OF LOW POWER DIVIDER TECHNIQUES USING RADIX

Rakesh Jain

Research Scholar M. Tech. VLSI, Mewar University, ***rk_patni10@yahoo.com***

Abstract

This work describes the design of a divder technique Low-power techniques are applied in the design of the unit, and energy-delay tradeoffs considered. The energy dissipation in the divider can be reduced by up to 70% with respect to a standard implementation not optimized for energy, without penalizing the latency. In this dividing technique we compare the radix-8 divider is compared with one obtained by overlapping three radix-2 stages and with a radix-4 divider. Results show that the latency of our divider is similar to that of the divider with overlapped stages, but the area is smaller. The speed-up of the radix-8 over the radix-4 is about 23.58% and the energy dissipated to complete a division is almost the same, although the area of the radix-8 is 67.58% larger.

----------***----------

1. INTRODUCTION

Division is the most complex of the four basic arithmetic operations and, in general, does not produce an exact answer, since the dividend is not necessarily a multiple of the divisor. Therefore, the correct quotient and remainder are usually obtained through performing a sequence of iterations until the desired precision is reached. This procedure is called sequential division and serves as the basic principle for many practical implementations [1, 3, 4].

Step	HW	LW	Explanation
initialize	0000	0111	set HW to 0, set LW to dividend
shift-left	0000	111x	
subtract	−0011		HW − divisor = −0011
	−0011	1110	result is < 0, set LSB of LW to 0, restore HW
restore	0000	1110	HW + divisor = 0000
shift-left	0001	110x	
subtract	−0011		HW − divisor = −0010
	−0010	1100	result is < 0, set LSB of LW to 0, restore HW
restore	0001	1100	HW + divisor = 0001
shift-left	0011	100x	
subtract	−0011		HW − divisor = 0000
	0000	1001	result is ≥ 0, set LSB of LW to 1, no restoring
shift-left	0001	001x	
subtract	−0011		HW − divisor = −0010
	−0010	0010	result is < 0, set LSB of LW to 0, restore HW
restore	0001	0010	HW + divisor = 0001

This work describes the design of a double-precision radix-8 divider. The unit is designed to reduce the energy dissipation without penalizing the latency.

The algorithm used is the digit-recurrence division algorithm described in detail in [3]. Digit-recurrence algorithms retire in each iteration a fixed number of result bits, determined by the radix. Higher radices reduce the number of iterations to complete the operation, but increase the cycle time and the complexity of the circuit. There are not many implementations of high-radix dividers, and the purpose of this paper is to determine the performance and the energy dissipation of a radix-8 unit. An evaluation of the area and performance of a radix-8 divider was done in [3] without an actual implementation. In [4] an algorithm for radix-8 division and square root with shared hardware is implemented. In [11] the implementation of a divider with three radix-2 overlapped stages is presented. In [5] it is commented that stages with radices larger than four are not convenient because of the increased complexity of the digit-selection function. We present here our implementation of a low-power radix-8 divider, compare its performance with the implementation presented in [11] and evaluate the speed-up and the energy consumption with respect to a more common radix-4 unit. Moreover, some energy-delay tradeoffs are considered.

The primary objective of the design is to perform the operation in the shortest time. Then low-power design techniques are applied in order to reduce the energy dissipated in the unit. Area is not minimized, but some energy reduction techniques reduce the area as well.

The implementation of the divider was done using the Passport 0.56 □ m standard-cell library [2]. The structural model was obtained by both manual design and synthesis of the functional blocks, and was laid out by using automatic floor-planning and routing. The latency of the division is reduced by choosing appropriate parameters in the algorithm that affect both the critical path and the number of iterations,

as described in Section 2. Section 3 describes the design for low-power

In Section 4 the divider is compared with a radix-8 obtained by overlapping three radix-2 stages, using a scheme similar to that implemented in the Sun *UltraSPARC* processor [11], and with a radix-4 divider [8].

The results show that the energy dissipation in the radix-8 unit can be reduced by up to 70% with appropriate low-power design techniques

2.ALGORITHM AND IMPLEMENTATION

The radix-8 division algorithm, described in detail in [3], is implemented by the residual recurrence

$$w[j+1] = 8w[j] - qj+1d$$

$$j = 0,1\ldots\ldots m$$

with initial value w[0] = x , where x is the dividend, d the divisor, and qj+1 the quotient digit at the j-th iteration. Both d and x are normalized in [0.5, 1). The quotient digit is in signed-digit representation {-a,…..,-1,0,1, ……, a} with redundancy factor p = a/7. The residual w[j] is stored in carry-save representation (wS and wC). The quotient digit is determined, at each iteration, by the selection function

$$qi=SEL (ds,y)$$

where d□ is d truncated after the □-th fractional bit and [y] = 8wS + 8wC truncated after t fractional bits.

The recurrence is implemented with a selection function (SEL), a multiple generator, a carry-save adder (CSA) and two registers to store the carry-save representation of the residual.

In order to avoid the implementation of a complicated multiple generator, the quotient digit is split into two parts qH with weight 4 and qL with weight 1 (qj = 4qH + qL) and the digit set of each part is reduced to {-2,-1,0,1,2}. Four signals (M2, M1, P1, P2) are used to represent these five values in a one-out-of-four code (zero is coded as (0,0,0,0)). This representation makes the multiple generator simple.

Since the selection function (SEL) is in the critical path, to have the minimum latency we have to minimize its delay. We explored the implementation of three possible values of a: 6, 7, and 10 (the maximum value possible with the above mentioned representation). Table 1 shows a summary of the results. The gate-level implementation was obtained by synthesizing the VHDL description of the selection function with Compass ASICSynthesizer. This includes both the assimilation of the carry-save representation of [^y] and the actual digit-selection function.

Table 1: Summary selection function

From Table 1, we can see that SEL for a = 7 is as fast as for a = 6, but its area is smaller. Surprisingly, the delay for the over-redundant case a = 10 is larger. Therefore, the SEL for a = 7 is chosen, which results in a redundancy factor p = 1.

A first implementation of the divider is shown in Figure 1. The scheme is completed by a controller (not depicted in the figure). The conversion block performs the conversion and the rounding. The quotient is rounded in the last iteration according to the sign of the final residual and the signal that detects if it is zero, which are produced by the sign-zero-detection block (SZD).

To have the divider compliant with IEEE standard for double-precision (53-bit significand normalized in [1,2)) while operating with fractional values, 1-bit shifts are performed on the operands. Moreover, to have a bound residual in the first iteration (w[0] = x □ □ d), when x□d we shift x one bit to the right obtaining a fractional quotient. To compute the 53 bits of the quotient and an additional bit to perform rounding, 54/3 = 18 iterations are required. An additional cycle is required to load the value x as first residual w[0]. However, for the proposed architecture and selection function, the simplest way to accomplish this is to do as follows:

- Clear the registers for w (this is done at the end of the previous division). With the selection function we have implemented, this produces a $q_1 = 1$.
- Compute w[1] = x - d using the hardware for the recurrence. This requires a multiplexer, which is not on the critical path.
- For q_1 to be 1, we shift the dividend three bits to the right. As a consequence, it is necessary to shift the final quotient accordingly,

In conclusion, the load cycle is substituted by an extra iteration for a total of 20 iterations: 19 to compute the digits and one for the rounding. Finally, the quotient is normalized in [1,2) by shifting it four positions to the left. Note that all shifts are done by wiring and do not affect the latency of the operation. In the recurrence (w[j]) we need 54 fractional bits and 2 integer bits: one to hold the sign and the other to avoid the overflow in the CS-representation (being p = 1).
There are two possible critical paths, one going through qH and the other through qL. Since the delay of qH is smaller than that of qL, but the number of adders to traverse is larger, a good design tries to equalize the delays of both paths. The resulting critical paths, pre-layout, are

The addition of the delay due to the clock distribution tree and the interconnection capacitance results in a post-layout critical path of 10.5 ns.

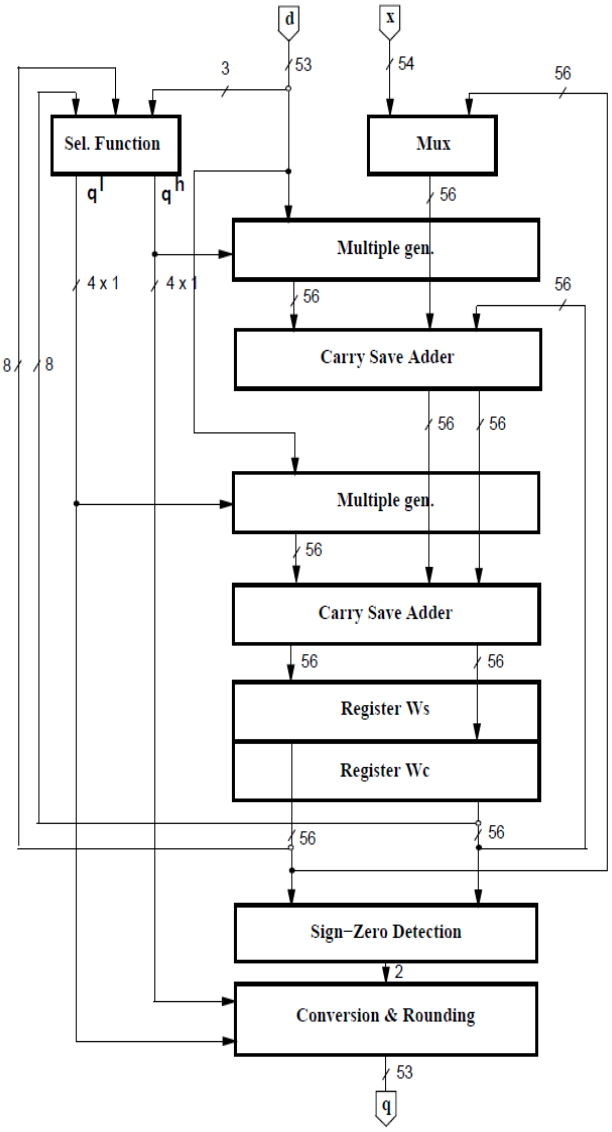

Figure 4.10: Implementation of the radix-8 divider.

Figure 1: Radix-8 divider implementation.

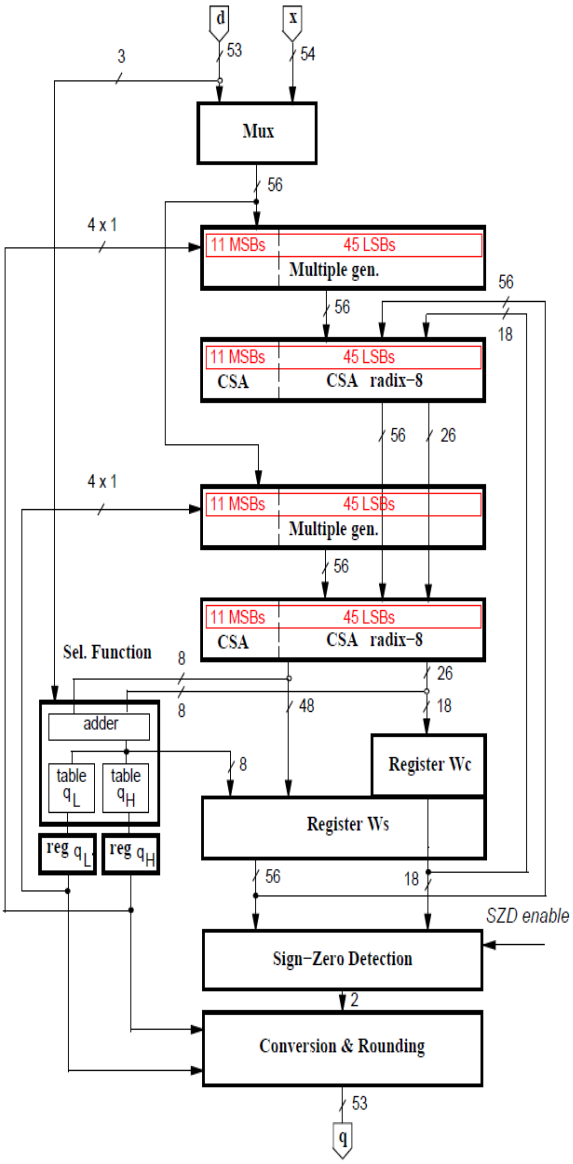

Figure 2: Low-power implementation.

3. LOW POWER IMPLEMENTATION

In this Section we apply techniques for the reduction of the energy dissipation in the unit of Figure 1. Some of the techniques applied here are described in [8] for the case of a radix-4 divider. These are adapted for radix-8 in Sections 3.1-3.4. In addition, techniques specific to radix-8 are introduced: Section 3.5 describes the partitioning of the selection function and Section 3.6 extends the modified convert-and-round algorithm (refer to [8] for a complete description) to the case p = 1. Finally, in Section 3.7 an evaluation of the impact of dual voltage on the low-power implementation is given.

Figure 2 shows the implementation of the low-power radix-8 divider and Table 2 summarizes the results obtained by applying the low-power techniques described in Sections 3.1-3.7. In the Table, entry *std* refers to the standard implementation, optimized for speed, and entry *l-p* is the low-power implementation. It was not possible to implement dual voltage with our cell library so that entry *d-v* is an estimate of a possible implementation.

3.1. Switching- off not active blocks

The sign-zero-detection block (SZD), which is only used in the rounding step, is switched off by forcing a constant logic value at its inputs during the recurrence steps. The reduction is about 8%.

3.2 Retiming the recurrence

The position of the registers in a sequential system affects the energy dissipation. Retiming is the transformation that consists in re-positioning the registers in a sequential circuit without modifying its external behavior [6].

For the recurrence, the retiming is done by moving the selection function of Figure 1 from the first part of the cycle to the last part of the previous cycle (see Figure 3.a and Figure 3.b). Two new registers (qH and qL) are needed to store the quotient digit.

By retiming the recurrence we reduce the switching activity in the multiple generators and in the CSAs, and change the critical path that is now limited to the eight most-significant bits. This allows the rest of the bits in the recurrence to be redesigned for low power.

3.3 Reducing transitions in multiplexer

The multiplexer in Figure 1 is used to select either x in the first iteration or the residual in the others. The number of transitions in the mux can be reduced by moving it out of the recurrence, as shown in Figure 2. Consequently, the operations in the first cycle are modified by resetting registers qH and qL to 0 and -1 respectively and by storing x in w[0] = 0 - (-x).

3.4Changing the redundant representation

By using a radix-8 carry-save representation with three sum bits and one carry bit for each digit in the recurrence, as shown in Figure 4, we only need to store one carry bit for each digit, instead of three. This can be done for the 45 LSBs that, after the retiming, are not on the critical path.
Furthermore, after the retiming, the eight MSBs, assimilated in the adder inside the selection function block (Figure 2), can be stored in w_S eliminating another eight flip-flops in w_C.

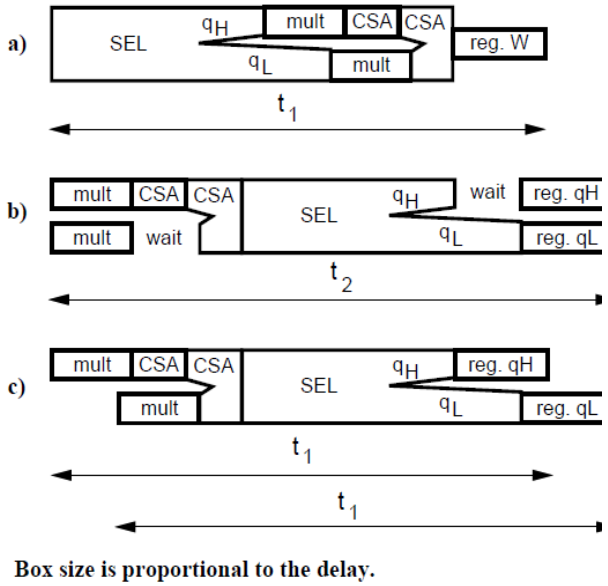

Figure 3: Retiming and critical path. a) before retiming, b) after retiming, c) after retiming and skewing the clock.

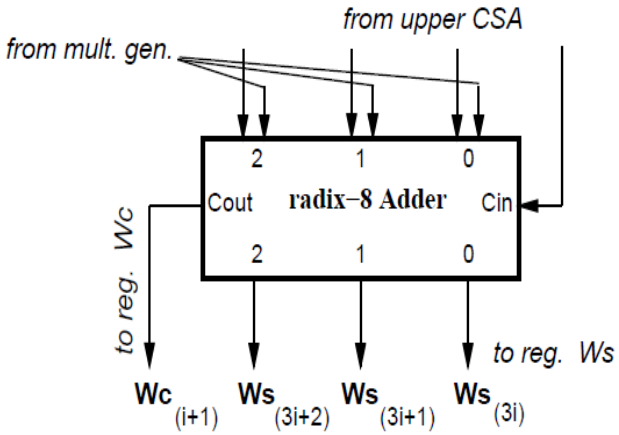

Figure 4: Radix-8 carry-save adder (lower).

By retiming, moving the mux, and changing the representation, the reduction in *l-p* with respect to *std* is about 14%.

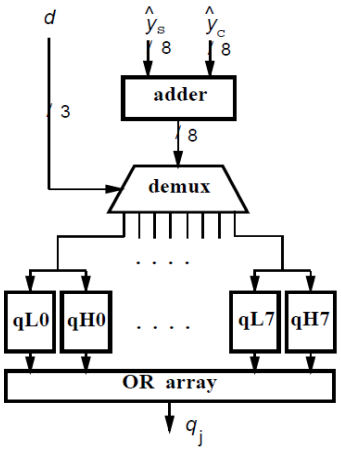

Figure 5: partitioned selection function.

3.5 Partitioning and disabling selection function

The quotient-digit selection is a function of three bits of the divisor and eight of the residual.

In the radix-8 case, Figure 5 shows the partitioning in eight parts (all the possible values of d_3) for both the higher and lower parts. The demultiplexer transmits the assimilated value of [^y] to the selected pair of selection tables and forces to zero the output of the others. Finally, an array of OR gates combines the partial values.

Experimental results showed that the partitioned selection function dissipates less energy, but the critical path increased. The smaller selection functions are faster than the implementation in one piece, but the delays in the demultiplexer and the OR gates offset the improvement.

3.6. Modifications in conversion And Rounding

The on-the-fly convert-and-round algorithm [3] performs the conversion from the signed-digit representation to the conventional representation in 2's complement. The conversion is done as the digits are produced and does not require a carry-propagate adder. The algorithm, in its original formulation, did not consider the energy dissipation, resulting in about 30% of the whole divider.

For p = 1, the partial quotient is stored in three registers (Q, QM, QP) updated in each iteration by shift-and-load operations, and the final quotient is chosen among those registers during the rounding. The large amount of energy dissipated in the unit is mainly due to the shifting during each iteration and to the number of flip-flops, used to implement the registers.

As a first step to reduce energy dissipation, we load each digit in its final position [9]. In this way, we avoid to shift digits along the registers. To determine the load position we use an 18-bit ring counter C, one bit for each digit to load.

3.7. Using dual voltage

The power dissipated in a cell depends on the square of the voltage supply (VDD) so that significant amount of energy can be saved by reducing this voltage [1]. However, by lowering the voltage the delay increases, so that to maintain the performance this technique is applied only to cells not in the critical path. Different power supply voltages require level-shifting circuitry that dissipate energy. However, by using two voltages we only need to level-shift when going from the lower to the higher voltage [13]. In our case, the 45 least-significant bits in the recurrence can be redesigned for low voltage, as shown in Figure 7. The voltage-level shifters are not needed until a specific digit moves towards the eleven MSBs, by shifting across iterations and into the critical path. By placing the voltage-level shifters (a total of three) in the digit immediately before the eleven MSBs the cycle time is not increased and the energy dissipated in the level-shifting circuitry is small.

We can apply the dual-voltage technique also to the convert-and-round unit which is not in the critical path. The number of level-shifters required is 53, as many as the double-precision significand representation, but because of the new algorithm, each bit switches at most twice and the energy dissipation in the level-shifters does not offset the reduction due to the lower voltage.

We estimated a reduction of about 50% in *d-v* with respect to *l-p* if low-voltage gates were available.

4. COMPARISON WITH RADIX-4 IMPLEMENTATION

To evaluate the tradeoffs between energy and delay, we compare the radix-8 divider with the radix-4 unit previously presented in [8]. We chose a radix-4 divider for the comparison because the algorithm is the same with the only variation of the radix, and the techniques to reduce the energy are similar.

The radix-4 divider has the disadvantage of requiring more cycles to compute the quotient (30 cycles) but the advantage of a shorter iteration cycle (faster clock) and smaller area.

The performance metric used is the time elapsed per operation which is *tdiv* = *Tcycle* ×*(no. of cycles*). The energy measure is the energy per division Ediv and we also include the energy per cycle Epc. Table 5 summarizes the characteristics of the two dividers.

Table 5: Radix-4 vs. radix-8 divider

	radix-4	radix-8	[unit]
T_{cycle}	9.3	11.5	ns
t_{div}	260	215	ns
E_{div}	26.0	26.6	nJ
E_{pc}	1.9	2.4	nJ
Area	2.2	2.8	mm^2

The speed-up for the radix-8 over the radix-4 is about 20%, while the increase in the energy-per-division is less than 2%. On the other hand, the radix-8 has an energy-per-cycle which is 50% larger. Our design shows that the increase of area (about 50%) does not reflect on the energy dissipated to complete an operation, which is almost the same because of the reduction in the number of cycles. The radix-4 divider is smaller, but it is slower and consumes almost the same amount of energy per operation.

REFERENCES

[1]. A. P. Chandrakasan and R. W. Brodersen. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.

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

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

[4] J. Fandrianto. Algorithm for high-speed shared radix-8 division and radix-8 square root. *Proc. of 9th Symposium on Computer Arithmetic*, pages 68-75, Sept. 1989.

[5]. D. Harris, S. Oberman, and M. Horowitz. SRT division architectures and implementations. *Proc. of 13th Symposium on Computer Arithmetic*, pages 18-25, July 1997.

[6]J. Monteiro, S. Devadas, and A. Ghosh. Retiming sequential circuits for low power. *Proc. of 1993 International Conference on Computer-Aided Design (ICCAD)*, pages 398-402, Nov. 1993.

[7]A. Nannarelli. PET: Power evaluation tool, Aug. 1996. http://www.eng.uci.edu/numlab/PET/.

[8]A. Nannarelli and T. Lang. Low-power radix-4 divider. *Proc. of International Symposium on Low Power Electronics and Design*, pages 205-208, Aug. 1996.

[9]A. Nannarelli and T. Lang. Low-Power Convert-and-Round Unit. *Technical Report*, Jan 1997. Available on the WWW at http://www.eng.uci.edu/numlab/archive/pub/nl97p02/.

[10]W. Nebel and J. Mermet editors. *Low Power Design in Deep Submicron Electronics*. Kluwer Academic Publishers, 1997

[11]A. Prabhu and G. Zyner. 167 MHz radix-8 divide and square root using overlapped radix-2 stages. *Proc. of 12th Symposium on Computer Arithmetic*, pages 155-162, July 1995.

[12]. J. M. Rabaey, M. Pedram, et al. *Low Power Design Methodologies*. Kluwer Academic Publishers, 1996.

[13]K. Usami and M. Horowitz. Clustered voltage scaling technique for low-power design. *Proc. of International Symposium on Low Power Design*, pages 3-8, Apr. 1995.