LOW-POWER RADIX-8 DIVIDER

Alberto Nannarelli and Tomas Lang
Department of Electrical & Computer Engineering
University of California, Irvine, California 92697
e-mail : alberto@ece.uci.edu, tlang@uci.edu

Abstract

This work describes the design of a double-precision radix-8 divider. 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. 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 20% and the energy dissipated to complete a division is almost the same, although the area of the radix-8 is 50% larger.

1. Introduction

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 $0.6\mu m$ 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. Several techniques are applied to reduce the energy consumption in the unit. Some of these techniques are applicable to a variety of CMOS circuits [10, 12, 1], while others are specific for the algorithm in question [8, 9]. The power estimation has been carried out with PET (Power Evaluation Tool) [7], 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.

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]. For these comparisons we have used the same standard-cell library and synthesis tools.

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. The performance of the radix-8 divider is similar to that of the divider with overlapped radix-2 stages, but the area is significantly smaller. Moreover, the speed-up of the radix-8 over the radix-4 is about 20% while the energy dissipated to complete a division is almost the same.

2. Algorithm and Implementation

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

$$w[j+1] = 8w[j] - q_{j+1}d \qquad j = 0, 1, \ldots m$$

with initial value $w[0] = x$, where x is the dividend, d the divisor, and q_{j+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 $\{-a, \ldots, -1, 0, 1, \ldots, a\}$ with redundancy factor $\rho = a/7$. The residual $w[j]$ is stored in carry-save representation (w_S and w_C). The quotient digit is determined, at each iteration, by the selection function

$$q_j = SEL(d_\delta, \hat{y})$$

where d_δ is d truncated after the δ-th fractional bit and $\hat{y} = 8w_S + 8w_C$ 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 q_H with weight 4 and q_L with weight 1 ($q_j = 4q_H + q_L$) 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 $\hat{y}$ and the actual digit-selection function.

	bits of		delay $[ns]$		
a	d	$\hat{y}$	q_L	q_H	gates
10	3	6	4.0	3.0	325
7	3	8	3.8	3.0	370
6	5	7	3.8	2.7	420

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 $\rho = 1$.

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 $x \geq d$ 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 $\rho = 1$).

There are two possible critical paths, one going through q_H and the other through q_L. Since the delay of q_H is smaller than that of q_L, 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

SEL(q_L)	+ mult	+ HA	+ reg	
3.8	+ 1.2	+ 0.5	+ 1.3	= 6.8 ns.

SEL(q_H)	+ mult	+ HA	+ FA	+ reg	
3.0	+ 1.2	+ 0.5	+ 0.8	+ 1.3	= 6.8 ns.

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.

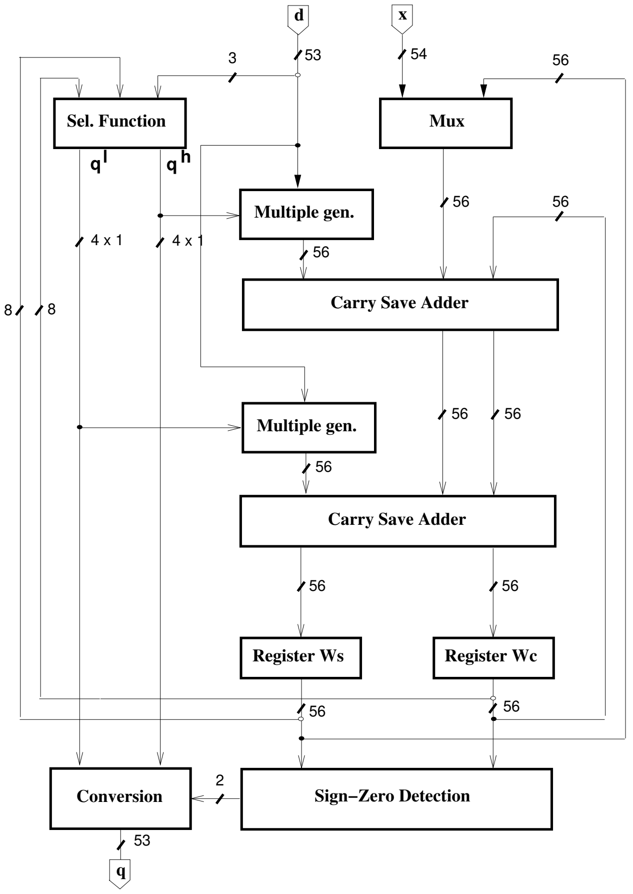


Figure 1. Radix-8 divider implementation.

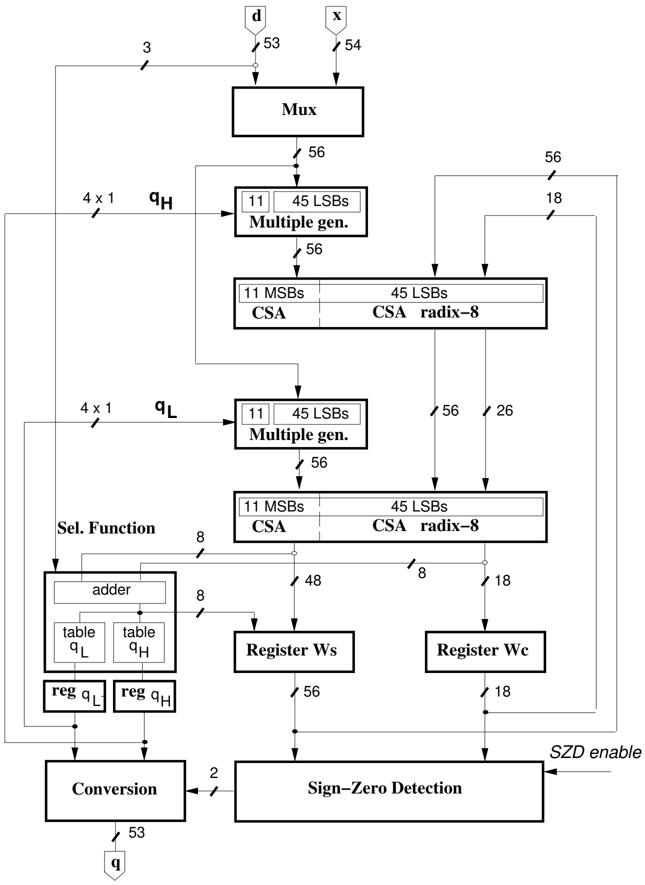


Figure 2. Low-power implementation.

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 $\rho = 1$. 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 (q_H and q_L) 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 recur-

blocks	std	l-p	d-v
control	0.6	0.6	0.6
clk tree	0.4	0.4	0.4
mux	1.4	0.2	0.1
mul. gen. H	3.1	1.8	0.7
CSA H	4.4	4.2	1.5
mul. gen. L	2.6	1.7	0.6
CSA L	6.0	5.3	1.9
SEL	3.6	4.0	4.0
register wc	4.2	1.2	0.4
register ws	4.2	4.0	*1.4
register qL	-	0.2	0.2
register qH	-	0.2	0.2
SZD	3.8	1.0	1.0
C&R unit	13.4	2.8	*1.0
Total $[nJ]$	47.7	27.6	14.0
Ratio	1.00	0.59	0.30
Area $[mm^2]$	2.2	1.8	-

Values marked * include level shifters.

Table 2. Energy-per-division for radix-8.

rence to be redesigned for low power.

However the retiming alters the critical path because the two paths through q_H and q_L have different delays, and now the delay of q_L is added to the delay of the two CSAs (see Figure 3.b). To reduce the critical path to the previous value we skew the clock of register q_L by the delay difference of the two paths, which, in this case, corresponds to the delay of the CSA, as shown in Figure 3.c.

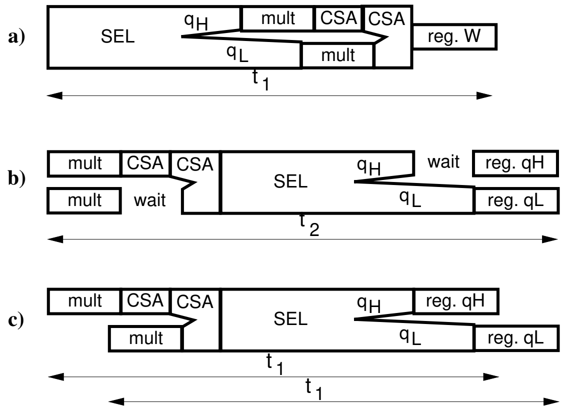
3.3. Reducing transitions in mux

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 q_H and q_L to 0 and -1 respectively and by storing x in $w[0] = 0 - (-x)$.

3.4. Changing 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.



Box size is proportional to the delay.

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

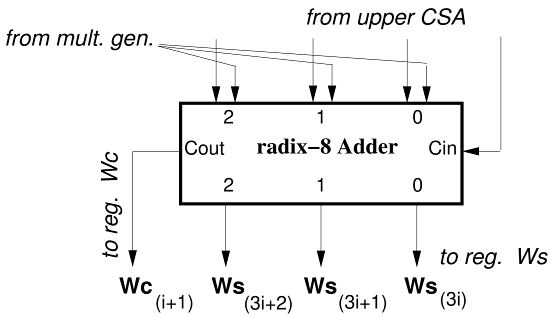


Figure 4. Radix-8 carry-save adder (lower).

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

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. Since the divisor is fixed for the whole division operation, from the point of view of energy consumption it is convenient to decompose the function into subfunctions and to enable only the subfunction corresponding to the actual value of the divisor. This is specially convenient for higher radices, because the quotient-digit selection is more complex and therefore is responsible for a significant portion of the energy.

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 $\hat{y}$ to the selected pair of selection tables and forces to zero the output of the others. Finally, an array of

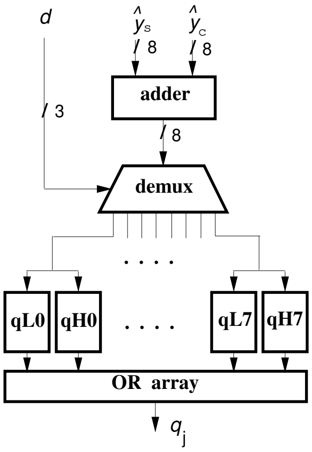


Figure 5. partitioned selection function.

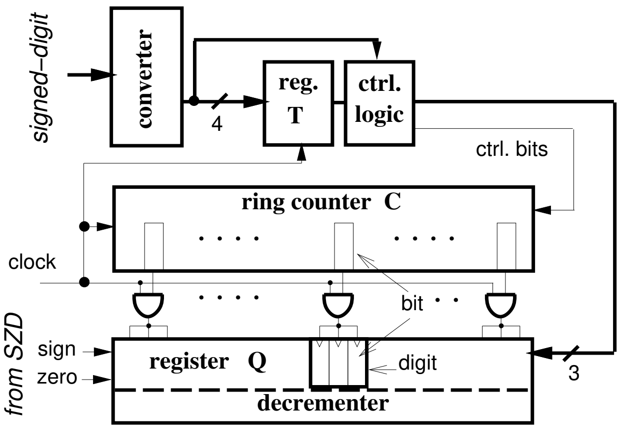


Figure 6. Convert-and-Round unit.

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.

By partitioning the selection function, we could obtain a reduction of 40% in the energy dissipated in SEL, but at the expense of a larger clock cycle (about 10%) and area. For this reason, the value is not included in Table 2.

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 $\rho = 1$, 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. As a second step, we reduce the partial-quotient registers from three to one, as follows:

- Register QM is needed when the converted digit is negative. In this case, instead of propagating the borrow to the previous digit, the value stored in QM is used. In the modified algorithm [9], we eliminate register QM and include in Q a digit decrementer. A digit is decremented only if the corresponding bit in the ring counter C is marked.

- Register QP is used in the rounding step when it is necessary to add 1 to a digit that is already $r - 1 = 7$ (max. value representable in three bits) and propagate the carry. In the modified algorithm, the register QP is eliminated by recoding the quotient digit into the digit set $\{-7, \ldots, -1, 0, 1, \ldots, 6\}$. More specifically, the value 7 is recoded into -1 and the previous digit incremented by one. This recoding requires to store the current quotient digit in a temporary register T (3 bits + 1 sign bit) as sketched in Figure 6. No additional cycle is needed since the conversion of the last digit is done together with the rounding. Table 3 shows an example of conversion using this recoding.

q_j	T	Q	C
4	4	xxxxxxxx	10000000
7	-1	5xxxxxxx	01000000
-1	-1	47xxxxxx	00100000
7	-1	470xxxxx	00110000
0	0	4677xxxx	00001000
7	-1	46771xxx	00000100
7	-1	467710xx	00000110
7	-1	4677100x	00000111

Final rounding			
add 1	x	46771000	00000111
add 0	x	46770777	00000111

Table 3. Example of radix-8 recoding.

Summarizing, the algorithm is modified by eliminating the shifting of the digits previously loaded and two partial quotient registers, and by recoding. Two additional registers are introduced: the ring counter to keep track of the digits to update and the temporary register T used in the recoding. In this way, we reduce the number of flip-flops in the convert-and-round unit from 162 (registers Q, QM, QP) to 76 (registers Q, C, T).

As a further step to reduce the energy dissipation in the convert-and-round unit, we switch off the clock signal and disable the trees that propagate signals to flip-flops that don't have to be updated [9].

The reduction in *l-p* with respect to *std* is about 20%.

3.7. Using dual voltage

The power dissipated in a cell depends on the square of the voltage supply (V_{DD}) 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.

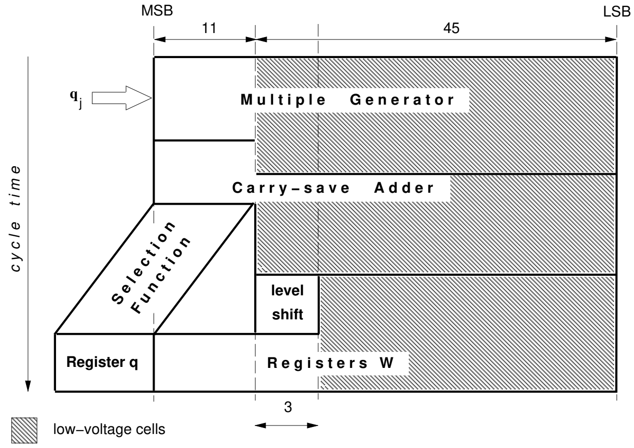


Figure 7. Low-voltage cells in the recurrence.

4. Comparison with Other Implementations

4.1. Comparison with scheme with overlapped radix-2 stages

In [11] a radix-8 divider is implemented by overlapping three radix-2 stages and computing the quotient digits in parallel. Moreover, the next partial remainder (w_s and w_c) is calculated speculatively for each possible quotient digit. This scheme, indicated here as *r8overlap*, is implemented in the Sun *UltraSPARC* FP-unit. As described in [11], the critical path is: $1 \times SEL + 2 \times CSA + 3 \times MUX$

In order to compare the *r8overlap* division unit with our radix-8 divider, we made the following assumptions:

- The CSA has the same delay (0.8 ns) in both implementations.
- The multiple generator is equivalent to the 3:1 MUX of *r8overlap*.
- We implemented the radix-2 selection function of [11] with our library and obtained a delay of 1.9 ns.
- Buffering of one of the MUXes is required.

With these assumptions, we can reasonably estimate the pre-layout critical path of *r8overlap* as:

SEL	+ 2 CSA	+ 3 MUX	+ buff.	+ reg.
1.9	+ 1.6	+ 1.5	+ 0.7	+ 1.3 = 7.0 ns

This is similar to the 6.8 ns of the radix-8 unit described here.

As for the area, Table 4 shows a comparison of the number of the wider (bitwise) blocks. Register QN in *r8overlap* can be eliminated by introducing register C. However it is not possible to change the representation of w_C (e.g. reducing the size of register Wc) without penalizing the performance. Moreover, the resulting selection function of the overlapped implementation is about twice as large as the radix-8 SEL. In conclusion, it is reasonable to assume that the area of our divider is significantly smaller.

We don't have data available on the energy consumption of the *r8overlap* division unit, but considering the larger area (larger current drawn) and roughly the same operation latency, we conclude that the energy dissipation is smaller in our implementation. Furthermore, the energy reduction

	r8overlap		radix-8	
no. CSAs	6		2	
no. mux/mult	6		2	
no. registers	4	(Ws, Wc, Q, QN)	2.7	(Ws, Wc/3, Q, C)

Table 4. Area comparison.

techniques applied in the radix-8 divider might not be effective in the *r8overlap* scheme.

4.2. Comparison with a Radix-4 Divider

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 $t_{div} = T_{cycle} \times (\textit{no. of cycles})$. The energy measure is the energy per division E_{div} and we also include the energy per cycle E_{pc}. Table 5 summarizes the characteristics of the two dividers.

	radix-4	radix-8	[unit]
T_{cycle}	8.3	10.5	ns
t_{div}	250	210	ns
E_{div}	27.0	27.6	nJ
E_{pc}	0.9	1.4	nJ
Area	1.2	1.8	mm^2

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

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.

5. Conclusions

In this work we presented the implementation of a low-power radix-8 divider. The energy dissipated in the unit was reduced by about 40% and we estimated a reduction of up to 70% by implementing the unit with dual voltage. The radix-8 divider latency is similar to that of the *r8overlap* division unit but its area is significantly smaller. The radix-8 also showed a speed-up of about 20% over a low-power radix-4 divider while the energy dissipated to complete a division was almost the same.

Acknowledgment

This work was partially supported by NSF grant MIP 9314172 and by State of California and Sun Microsystems, through UC MICRO 97-084.

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.