Low-Power Division: Comparison among implementations of radix 4, 8 and 16

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

Although division is less frequent than addition and multiplication, because of its longer latency it dissipates a substantial part of the energy in floating-point units. In this paper we explore the relation between the radix and the energy dissipated. Previous work has been done on radix-4 and radix-8 division. Here we extend this study to a radix-16 scheme with two overlapped radix-4 stages and compare the latency, area, and energy of the three implementations.

Results show that by applying the low-power techniques the energy dissipation is reduced from 30% to 40%, with respect to the standard implementation. An additional 20% reduction can be obtained using a dual voltage. Moreover, the energy dissipated to complete the division is roughly the same for the three radices. However, the power dissipation, proportional to the average current, increases with the radix. If reducing the energy is the priority, for the same latency radix-16 with dual voltage produces the smallest energy dissipation.

1 Introduction

Energy consumption is becoming more important every day because of the increased densities on chip and faster clocks. Although it is an infrequent operation, compared to addition and multiplication, the longer latency makes division dissipate a substantial portion of the energy in arithmetic units. For example, a rough evaluation of the energy dissipated in a floating-point unit, similar to the one presented in [8] shows that the energy consumption in the divider is comparable to that of the other components.

In this paper we explore the relation between division radix and energy dissipated. Previous work has been done in [5, 7, 6]. Here we extend the techniques used for radix-4 and radix-8 to a radix-16 scheme, obtained by overlapping two radix-4 stages. All implementations are for a 53-bit rounded quotient. A comparison of the three implementation is made with respect to the latency, energy consumption, and area. Results show that by applying the low-power techniques the energy dissipation is reduced from 30% to 40%, with respect to the standard implementation. An additional 20% reduction can be obtained using a dual voltage. Moreover, the energy dissipated to complete the division is roughly the same for the three radices. However, the power dissipation, proportional to the average current, increases with the radix. If reducing the energy is the priority, for the same latency radix-16 with dual voltage produces the smallest energy dissipation.

The units are implemented using the Passport $0.6\mu m$, $3.3V$, three-metal layers, CMOS standard cell library [2] and the layout is obtained by automatic floor-planning. The delay is computed by multiplying the number of cycles by the critical path obtained from simulations. The area is obtained from the layout and the energy dissipation is estimated using PET [4], a power evaluation tool which computes the power dissipated in a circuit from the netlist extracted from the layout (that includes the effect of interconnection capacitance), the standard cell library characteristics, and the results of a logic-level simulation. We report the energy dissipated in a complete division and the power, which is proportional to the average current.

2 Algorithm and Implementation for Radix-16 Divider

In this Section we review the division algorithm described in detail in [3]. The division is implemented by the residual recurrence

$$w[j+1] = 16w[j] - q_{j+1}d \qquad j = 0, 1, \ldots 13$$

with initial value $w[0] = x$, where x the dividend, d the divisor, and q_{j+1} the quotient digit at the j-th iteration, such

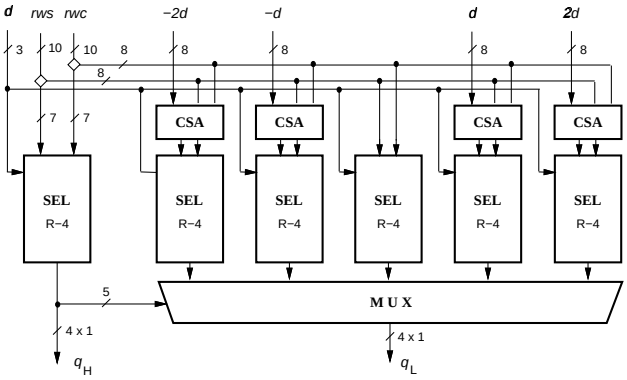

Figure 1. Selection function for radix-16

that the quotient is

$$q = \sum_{j=1}^{14} q_j 16^{-j} \tag{1}$$

where q_{j+1} is determined by the selection function described below. Two additional cycles, for initialization and rounding, are required to produce the quotient in conventional representation (53 bits for IEEE double-precision) for a total of 16 cycles. Both d and x are normalized in $[0.5, 1)$[1] and $x < d$.

The radix-16 division unit is obtained by overlapping the computation of two radix-4 digits [9]. Consequently, the quotient digit is split into two parts q_H and q_L such that $q_j = 4q_H + q_L$ with digit set $\{-2, -1, 0, 1, 2\}$ in each part, resulting in the digit-set $[-10, 10]$ for q_j ($a = 10$). The quotient digit is determined, at each iteration, by the selection function depicted in Figure 1. Once the digit q_H is chosen, its value is used to select among all the possible combinations of $q_H d$. The redundancy factor is $\rho = \frac{a}{r-1} = \frac{2}{3}$. The residual $w[j]$ is stored in carry-save representation (w_S and w_C). The signed-digit representation of the quotient is converted to conventional two's complement representation and rounded by the on-the-fly convert-and-round unit.

The implementation of the standard divider, optimized for shortest latency, is shown in Figure 2. The recurrence is implemented with the selection function (SEL), two multiple generators (MULT), two carry-save adders (CSA)[2] and two registers (REG) to store the carry-save representation of the residual. Because of the carry-save representation of the residual, the selection function (SEL R-4) in Figure 1 is composed by a 7-bit carry-propagate adder and a function implemented with logic gates. The converter performs the conversion and the rounding according to the sign of the final residual and the signal that detects if it is zero, which are produced by a sign-zero-detector (SZD).

[1] In IEEE standard floating-point numbers are normalized in $[1, 2)$. However this is equivalent to right-shift the operands one position.

[2] Each bit of the CSA is implemented with two half-adders (HA).

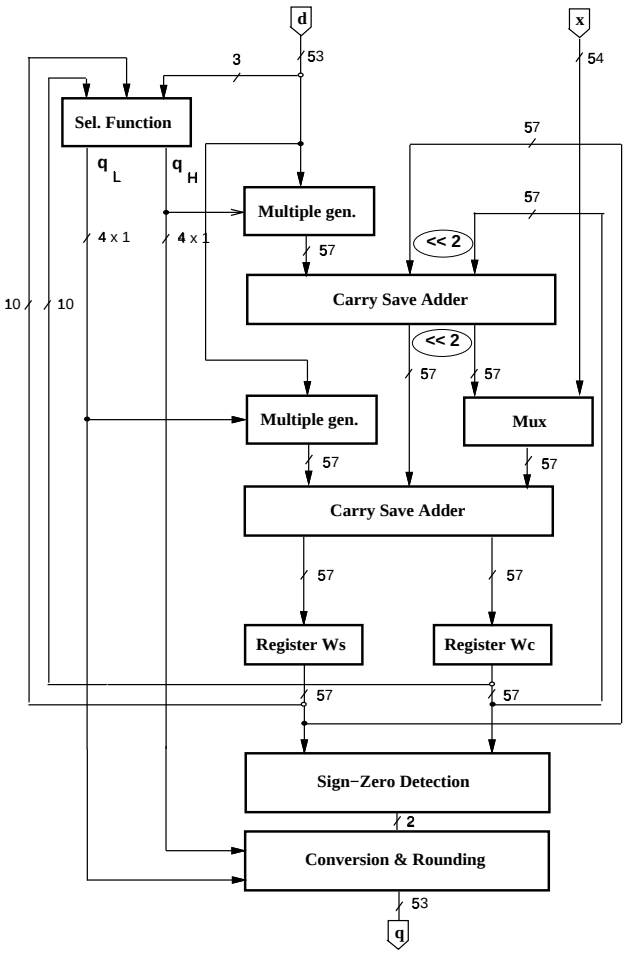

Figure 2. Implementation radix-16

Table 1 shows the delay through the two parts of SEL. Note that the larger delay of SELq_L is compensated by the additional CSA that exists in the path from SELq_H.

The critical path post-layout is 9.2 ns and 16 iterations are required to complete the operation, corresponding to a latency of 150 ns.

	path		delay [ns]
q_L	SELq_L - MULT - HA - REG		
	5.7 + 1.4 + 0.6 + 1.5	=	9.2
q_H	SELq_H - MULT - HA - FA - REG		
	4.0 + 1.4 + 0.6 + 1.1 + 1.5	=	8.6

Table 1. Critical path through q_L and q_H.

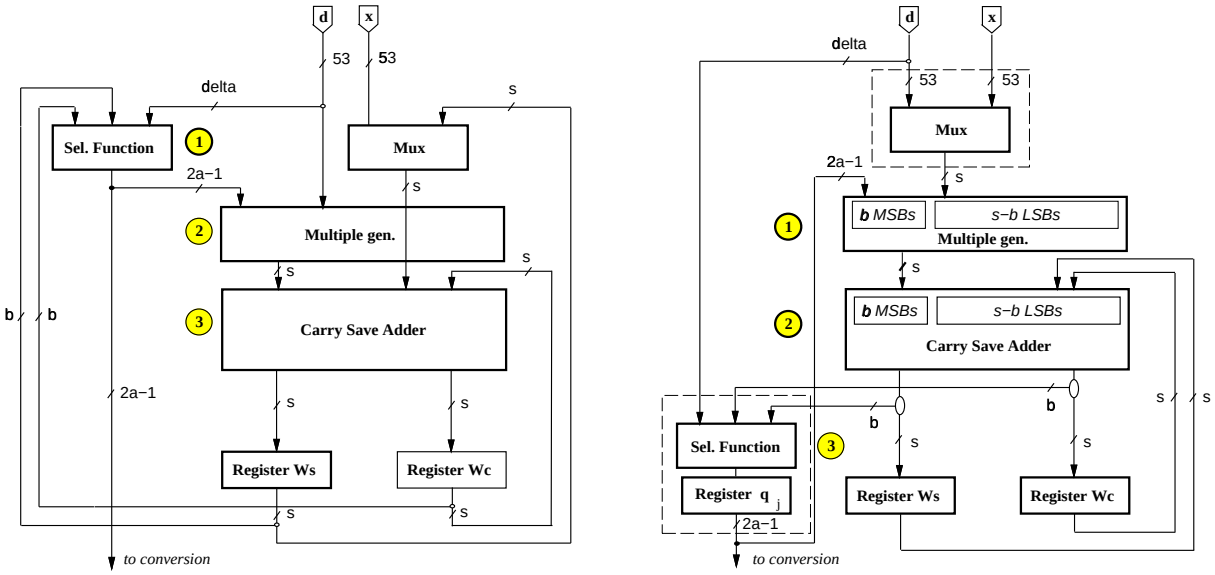

Figure 3. Retiming the recurrence

3 Low-Power Implementation

Low-power design techniques are applied to the standard implementation of the divider to reduce the energy consumption without penalizing the latency. The techniques, reviewed below, were originally developed for radix-4 [7] and are here adapted to the radix-16 case.

3.1 Retiming of the recurrence

The retiming of the recurrence is done by moving the selection function from the first part of the cycle to the last part of the previous cycle. This is shown in Figure 3 for a radix-r divider. A new register is needed to store the quotient digit.

This retiming has the advantage of limiting the critical path to the b most-significant bits of the recurrence ($b = 10$ for radix-16), so that the rest can be redesigned for low power. For instance, for the path through SELq_L, the two paths are

```
MSBs: MULT - HA - SEL QL   - REG
LSBs: MULT - HA -   CSA    - REG
```

On the other hand, the retiming requires a small extra register. Moreover, now the Mux is in the critical path, but because it changes once per division, its delay can be masked by earlier selection.[3]

As indicated in the previous section, and shown in Figure 4a, the larger delay of SELq_L is compensated by the additional carry-save adder in the path of SELq_H. This compensation is eliminated by the retiming, as shown in Figure 4b. Consequently, in order not to increase the cycle delay, the clock of register q_L is skewed, resulting in the paths of Figure 4c. The clock can be skewed by adding the appropriate delay (e.g. some buffers) in the clock distribution tree.

Since in the retimed implementation the selection function is placed after the second CSA, instead of directly af-

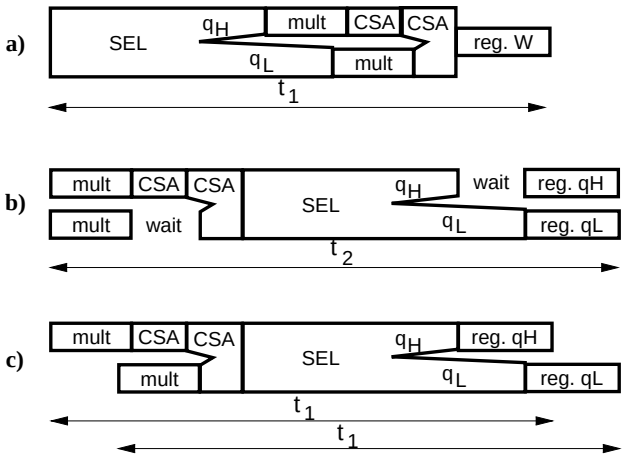

Figure 4. Retiming and critical path. a) before retiming, b) after retiming, c) after retiming and skewing the clock.

[3]The mux-select is the only signal sent from the controller to the recurrence and it can be skewed (anticipated) at the end of the first cycle masking the delay of the multiplexer.

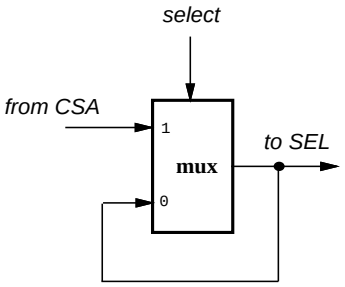

Figure 5. Multiplexers to filter glitches in selection function

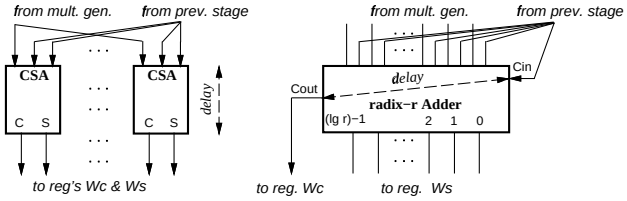

Figure 6. Replacing CSAs with radix-r CSAs

ter the registers, there is a large increase in the number of glitches, which are responsible for the increased dissipation of the selection function. One way to filter those glitches is to buffer the selection function with multiplexers acting as latches, as described in Figure 5. The select signal is driven by a different clock (same period, different phase) that enables the muxes to transmit the value from the CSA when it is stable, and hold the current value otherwise. However, in this case the delay of the mux affects the critical path. For radix-16 the energy dissipated in the selection function is halved, but the critical path is increased by about 5%. For this reason, the value is not included in Table 2, which summarizes the energy reductions.

3.2 Changing the redundant representation

As explained above, after the recurrence has been retimed the critical path is limited to the 10 most-significant bits and the rest can be redesigned for low power. One way of doing this is to replace the (radix-2) CSA for the least-significant bits by a radix-16 CSA (R16 CSA) that for each digit of the radix stores only one carry bit. This is shown in Figure 6 for a generic radix. This modification increases the delay of the CSA but reduces the number of flip-flops in the registers for the residual, resulting in a reduction of the energy (and of the area). This can be done for the 44 LSBs by implementing 11 radix-16 CSAs in each CSA, so that the number of flip-flops in register Wc is reduced from 57 to 25.

Moreover, an additional reduction in the energy dissipated is obtained by using low-drive cells for the LSBs in the recurrence.
In order not to increase the cycle time when using radix-16 CSA the two following paths should have the same delay:

```
MSBs: MULT - (  HA    -  SEL QL) - REG
LSBs: MULT - (R16 CSA - R16 CSA) - REG
```

Therefore, the condition to be satisfied is:

$$t_{R16\ CSA} \leq \frac{t_{HA} + t_{SEL_{q_L}}}{2} = 2.8 \quad ns$$

The easiest way to implement this R16 CSA is by using a 4-bit carry-ripple adder. The corresponding delay, in our library, is

$$t_{Sn} = t_{C_1} + 3t_{ripple} + t_{HA} = 0.8 + 3(0.35) + 0.5 \quad ns$$

which corresponds to about 2.0 ns.

Furthermore, since the most-significant bits of the residual are assimilated in the selection function, these assimilated bits could be stored in the register Ws, saving 7 additional flip-flops. However, in the radix-16 case, the assimilated value must be selected among the 5 possible alternatives (see Figure 1) and this requires an additional multiplexer driven by q_H that increases the load both on q_H and q_L. For this reason the 7 MSBs bits are stored in carry-save representation.

The combination of the retiming and changing the representation results in a reduction of about 10% in the energy dissipated in the divider.

3.3 Conversion and Rounding

The conversion and rounding is performed by the on-the-fly convert-and-round algorithm [3]. In the original algorithm the partial quotient is stored in two registers updated in each iteration and the final quotient is chosen among those registers during the rounding. The update of the registers is done by shift-and-load operations. The large amount of power 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.

In [7] a modified algorithm is described, which reduces the energy by changing the way the registers are loaded and their number. The same modifications can be applied to the radix-16 case. Because of the higher radix, the reduction in the number of the flip-flops is larger.

With the implementation of the modified algorithm the number of flip-flops in the convert-and-round unit is reduced from 108 to 69 and the power dissipated from $10.7nJ$ to $2.4nJ$, resulting in a reduction of about 20% in the whole divider.

In addition, the sign-zero-detection block (SZD), which is only used for the rounding, is switched off by forcing a

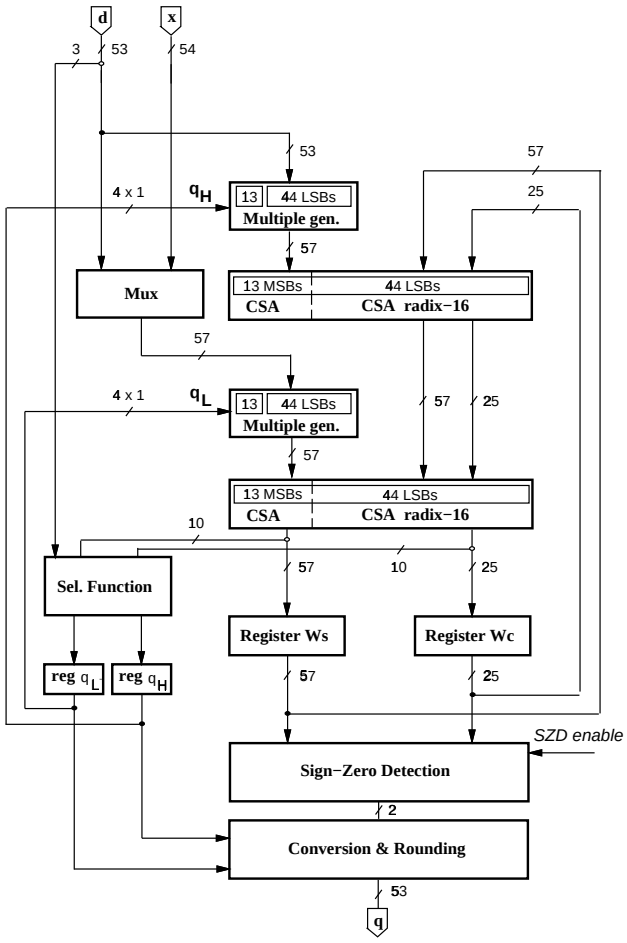

Figure 7. Low-power radix-16 divider

blocks	std nJ	l-p nJ
control	0.5	0.5
clk tree	0.5	0.5
mux	2.6	0.4
mul. gen. H	2.5	1.6
CSA H	3.3	3.3
mul. gen. L	2.7	1.8
CSA L	5.0	4.3
sel. func.	5.9	8.2
registers W	8.6	5.9
registers q	-	0.4
SZD	3.7	1.0
C&R unit	10.7	2.4
E_{div} $[nJ]$	46	30
Ratio	1.00	0.66
Area mm^2	2.2	1.8

Table 2. Energy-per-division for radix-16.

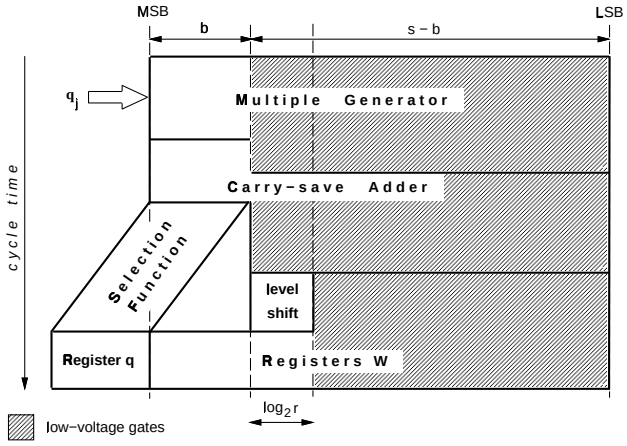

Figure 8. Low-voltage cells in the recurrence.

constant logic value at its inputs during the recurrence steps. The reduction in the energy dissipated is about 5% with respect to the standard implementation.

3.4 Results

Table 2 reports the average energy dissipation and area for the standard and low-power implementation, which is shown in Figure 7. In the Table, entry *std* refers to the standard implementation, optimized for speed, and entry *l-p* is the low-power implementation with the same delay. Some of the low-power techniques used, such as modified conversion algorithm and changed redundant representation, reduce the number of flip-flops in the registers and, consequently, the area.

4 Dual Voltage for Radix-4, 8 and 16

In this section we evaluate the impact of the use of dual voltage for units implementing radix-4, 8 and 16, division. 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. In the recurrence the $s - b$ least-significant bits can be redesigned for low voltage, as shown in Figure 8. By using two voltages we only need to level-shift when going from the lower to the higher voltage [10]. As a consequence, the voltage-level shifters are not needed until a specific digit moves into the b MSBs.

Our library was designed to operate with $V_{DD} = 3.3\ V$. To obtain the dependency of the delay with respect to V_{DD}, we performed SPICE simulations on a 4-bit carry-ripple adder and obtained Figure 9. The delay is normalized to the one for $V_{DD} = 3.3\ V$. The plot shows that for $V_{DD} = 2.0\ V$ the delay is doubled, and that for voltages

scheme	critical path	T_{cycle}	no. iter.	latency	speed-up
radix-4	SEL - MULT - HA - REG				
	3.9 + 1.4 + 0.6 + 1.1 =	7.0	30	210	1.0
radix-8	SELq_L - MULT - HA - REG				
	4.5 + 1.4 + 0.6 + 1.5 =	8.0	20	160	1.3
	SELq_H - MULT - HA - FA - REG				
	3.5 + 1.4 + 0.6 + 1.0 + 1.5 =	8.0			
radix-16	SELq_L - MULT - HA - REG				
	5.7 + 1.4 + 0.6 + 1.5 =	9.2	16	150	1.4

Table 3. Critical path, latency and speed-up for radix-4, 8 and 16.

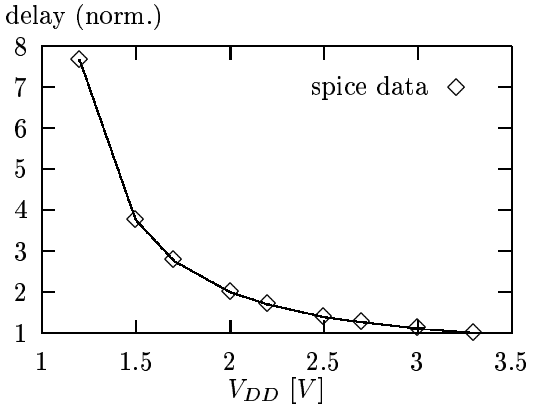

Figure 9. Delay with different V_{DD}

below $1.7\,V$ the delay increases in excess.

In order to evaluate the possible lower voltage V_2 to be used in a dual voltage implementation we need to determine the time slack available for the LSBs. The time slack is the difference between the delay in the paths through the MSBs and LSBs, and it gives the amount of time available for the delay of gates whose voltage is scaled to V_2. To compute the time slack, data on critical path and latency (ns) for the implementation of radix-4, 8 and 16 are reported in Table 3.

The delay of the least-significant portion depends on the type of CSA adder used, since the delay of the radix-r CSA is larger than that of the radix-2 CSA. For instance, for radix-4 the critical path is 7.0 ns. By implementing the LSBs of the recurrence with radix-2 CSAs, the delay in the LSBs is 3.1 ns, resulting in a time slack of 3.9 ns. In this case $V_2 = 2.0\,V$ without affecting the latency of the divider. On the other hand, by opting for the use of radix-4 CSAs, the time slack is reduced to 3.0 ns and, consequently, V_2 can be lowered to $2.2\,V$.

The energy consumption for dual voltage was estimated on a block basis, by using the following expression:

$$E_{d-v} = E_{l-p}\left[\frac{b}{s} + \left(\frac{V_2}{V_{DD}}\right)^2\left(1 - \frac{b}{s}\right)\right]$$

where E_{l-p} is the energy dissipated in the block in the *l-p* implementation. This assumes no variations in neither load capacitance nor activity. SPICE simulations showed that the values provided by the above expression are an overestimate of the actual power dissipation (less than 10%) for V_2 values from $3.3V$ to $2.0V$.

The estimated values are reported in Table 4 and include the contribution of voltage-level shifters.

Since the reduced voltage can be lower for radix-2 CSA, this might result in a reduction of the total energy. There is a tradeoff between the following:

- The voltage can be lower for radix-2 CSA
- There is a reduction in the number of flip-flops by using the radix-r CSA.

As seen in Table 4, a larger reduction is obtained for radix-2 CSA. For radix-4, the same estimated values are obtained so that the radix-4 CSA solution might be preferred because of the smaller area. We have also applied the dual-voltage technique to the convert-and-round unit.

	with radix-2 CSA			with radix-r CSA		
	t. slack $[ns]$	V_2 $[V]$	E_{div} $[nJ]$	t. slack $[ns]$	V_2 $[V]$	E_{div} $[nJ]$
radix-4	3.9	2.0	17	3.0	2.2	17
radix-8	3.5	2.0	20	1.2	3.0	26
radix-16	4.6	2.0	22	1.8	2.7	27

Table 4. Energy-per-division after voltage scaling.

5 Comparison between Radix-4, 8 and 16

Table 5 summarizes the energy dissipation and area for the implementations of radix-4, 8 and 16. It was not possible to implement dual voltage with our cell library so that entry *d-v* is an estimate of a possible implementation.

The results show that techniques for low-power design are effective for the three radices (Figure 10). However for

		E_{div} $[nJ]$			Area $[mm^2]$		Speed-up
		std	l-p	d-v	std	l-p	
radix-4		45	27	17	1.4	1.2	1.0
	ratio	1.0	0.59	0.37			
radix-8		48	29	20	2.2	1.8	1.3
	ratio	1.0	0.61	0.42			
radix-16		46	30	22	2.2	1.8	1.4
	ratio	1.0	0.66	0.48			

Table 5. Energy-per-division, area, speed-up.

the dual-voltage implementation the increased complexity in the recurrence datapath for the higher radices (double MULT and CSA and larger SEL) reduces the improvement. The longer critical path not only reflects in a longer T_{cycle}, but also limits the lower voltage to be used. In fact, voltage scaling is more effective for simpler datapaths, where the ratio $\frac{\text{delay MSBs}}{\text{delay LSBs}} > 2$.

The data from Table 3 and Table 5 are plotted in Figure 11. The target is to obtain an implementation that is as close as possible to the origin: minimum energy and latency. It is interesting to note that for the three radices the energy-per-operation is in a close range for each type of design (*std*, *l-p*, *d-v*).

Finally, we comment on the energy per cycle and the power. This is shown in Figure 12. As expected, the energy per cycle increases with the radix because of the larger complexity of the implementation, resulting in a larger number of transitions. In contrast to the latency, which is reduced for higher radices, the reduced number of iterations for higher radices cannot totally compensate the higher E_{pc} in $E_{div} = E_{pc} \times$ (no. cycles).

The power, which is

$$P = \frac{E_{pc}}{T_{cycle}} = V_{DD}\overline{I}$$

also increases with the radix. However, the increase is smaller than that of E_{pc} because of the increase in T_{cycle}.

In conclusion, for reducing the power (i.e. the average current) in a divider the best solution is to use a radix-4. On the other hand, if low energy is the priority, like for portable electronics where the life time of batteries depends on E_{div}, one might think of using a radix-16 with a lower voltage to keep the latency of a radix-4, but smaller E_{div}. For example, consider a divider with latency of 210 ns. This can be achieved with radix-4 and *l-p* implementation ($E_{div} = 27$nJ) or with radix-16 with a lower voltage of $V_{DD} = 2.5\,V$, dissipating only $E_{div} = 18$nJ.

Acknowledgments

This work was partially supported by NSF grant MIP 9314172 and by State of California and Sun Microsystems,

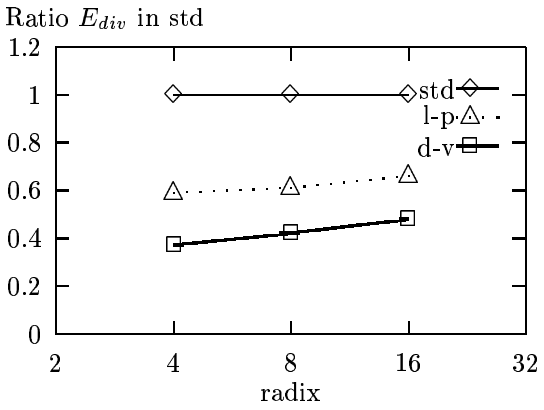

Figure 10. Energy-per-operation reductions

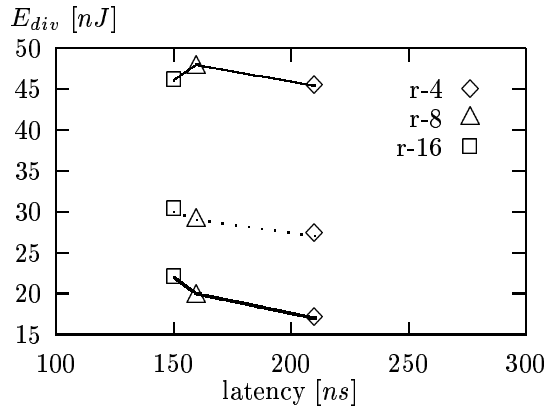

Figure 11. Energy-Latency diagram

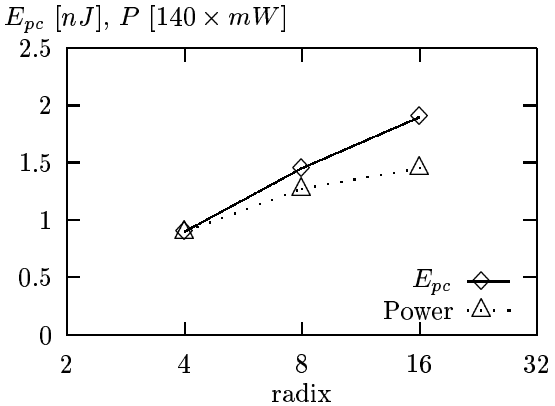

Figure 12. E_{pc} and power (scaled) for radix-4, 8, and 16

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. Ercegovac and T. Lang. *Division and Square Root: Digit-Recurrence Algorithms and Implementations.* Kluwer Academic Publisher, 1994.

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

[5] 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.

[6] A. Nannarelli and T. Lang. Low-Power Radix-8 Divider. *Proc. of International Conference on Computer Design (ICCD)*, pages 420–426, Oct. 1998.

[7] A. Nannarelli and T. Lang. Low-Power Divider. *IEEE Transactions on Computers*, pages 2–14, Jan. 1999.

[8] 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.

[9] G. Taylor. Radix-16 SRT dividers with overlapped quotient selection stages. *Proc. of 7th Symposium on Computer Arithmetic*, pages 64–71, 1985.

[10] 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.