Comparison of Parallelized Radix-2 and Radix-4 Scalable Montgomery Multipliers

Andrew Carter, Paula Ning, William Koven, David Money Harris,
Michael Braly, Nathan Jones, Julien Massas, Trevin Murakami, Alexandra Simoni
Harvey Mudd College
301 Platt Blvd., Claremont, CA, USA
Email: acarter@cs.hmc.edu, david_harris@hmc.edu

Sanu Mathew
Intel Corporation
Hillsboro, OR, US

***Abstract*—This paper compares $130nm$ custom silicon implementations of three scalable Montgomery multiplier architectures to previously published FPGA implementations of the same architectures. It investigates the delay, energy, and area tradeoffs of parallelized left-shifting radix-2, radix-4, and Booth-encoded radix-4 architectures. The radix-4 architecture is most efficient, performing 256×256-bit modular multiplication in $453ns$ while consuming $15.7nJ$ of energy and occupying an area of $0.141mm^2$. The radix-2 architecture is a close second, with an energy-delay product (EDP) 0.8% higher and an area-delay product (ADP) 3.1% higher. The Booth-encoded radix-4 architecture eliminates the need for an adder generating a $3\times$ multiple, but comes at a cost of 36% in EDP and 34% in ADP relative to the conventional radix-4 architecture. The relative efficiencies of the silicon implementations are consistent with the FPGA implementations.**

I. Introduction

Modular exponentiation is the basis for modern cryptographic algorithms such as RSA and digital signatures. These algorithms generally process 256 to 2048-bit numbers to ensure sufficient security. Modular multiplication is time-consuming because of the division step to find the remainder. However, Montgomery multiplication [1] allows the modulo operation to be done in a more convenient base, eliminating the division at the cost of extra multiplications and additions. Hence, Montgomery multiplication forms the basis of efficient modular exponentiation.

To accommodate arbitrary n-bit multiplications, *scalable* Montgomery multipliers repeatedly cycle chunks of the operands through smaller, fixed-size processing elements (PEs). A scalable, radix-2^v PE of width w processes v bits of the multiplier and w bits of the multiplicand per operation. A Montgomery multiplier contains a systolic array of p such PEs.

The critical path of standard Montgomery multipliers contains two dependent multiplication steps. Orup demonstrated a method of reordering the steps so that these multiplications take place in parallel in a $w \times w$ multiplier [2]. This method was later generalized to $w \times v$ multipliers by Kelley [3]. Another optimization to reduce latency is to left shift the operands rather than to right shift the result [4]. Jiang and Pinckney have applied this technique to three scalable parallelized left shifting architectures: radix-2, radix-4, and Booth-encoded radix-4 Montgomery multipliers [5][6][7]. They measured the delay and area of Virtex II FPGA implementations. Our goal in this study is to measure the delay, area, and energy of custom $130nm$ silicon implementations of each of these three architectures, and to determine if the relative merits of the architectures is consistent with the FPGA implementations.

Section II defines the Montgomery multiplication algorithm. Section III illustrates the three PE architectures under consideration. Section IV describes the test chip and Section V presents the results.

II. Background

Montgomery multiplication is defined as

$$Z \equiv XYR^{-1} \mod M \tag{1}$$

where

X :	n-bit multiplier
Y :	n-bit multiplicand
M :	n-bit odd modulus, typically prime
R :	2^n
R^{-1} :	modular multiplicative inverse of $R \mod M$ $RR^{-1} \equiv 1 \mod M$

The steps of Montgomery multiplication are shown in Figure 1. Because $R = 2^n$, dividing by R is equivalent to shifting right by n bits. Q has the property that the lower n bits of $[Z + Q \times M]$ are 0. Hence, no information is lost during the reduction step.

Multiply:	$Z = X \times Y$
Quotient:	$Q = Z \times M' \mod R$
Result:	$Z = [Z + Q \times M]/R$
Normalize:	If $Z \geq M$ then $Z = Z - M$

Fig. 1. Algorithm for Montgomery multiplication

Figure 2 shows the parallelized scalable Montgomery multiplication algorithm [3]. The variables are defined as follows:

$$\begin{aligned} n_1 &= n + v + 1 && \text{extended number of bits of } Y \\ n_2 &\geq n + v + 2 && \text{extended number of bits of } X \\ f &= n_2/v && \text{outer loop length} \\ e &= \lceil n_1/w \rceil && \text{inner loop length} \end{aligned}$$

M'	:	n_2-bit integer which is the modular inverse of $-M \mod 2^{n_2}$ such that $-MM' \equiv 1 \mod 2^{n_2}$
$\hat{M}$	:	n-bit integer $[(M' \mod 2^v) + 1]/2^v$
C	:	$v+1$-bit carry

```
$Z = 0$
for $i = 0$ to $f - 1$
    $Q^i = Z^0 \bmod 2^v$
    $C = 0$
    for $j = 0$ to $e - 1$
        $(C, Z^j) = (Z^{j+1}_{v-1:0}, Z^j_{w-1:v}) + C$
                $+\ Q^i \times \hat{M}^j + X^i \times Y^j$
```

Fig. 2. Parallelized radix-2^v scalable Montgomery Algorithm

The number of cycles required for a left-shifting Montgomery multiplication is [6]

$$c = \frac{f}{p} \max\left(e + 1, p + \frac{vp}{w} + 1\right) \tag{2}$$

Hence when $p \approx n/w$, the multiplier performance saturates and ceases to benefit from more PEs.

III. Processing Element Architectures

This section presents block diagrams for the three PE architectures under consideration and analyzes the number of cycles to compute a Montgomery multiply.

A. Radix-2 (R2)

Figure 3 shows a parallelized radix-2 left shifting Montgomery multiplier [5]. The two w-bit AND gates compute the partial products of one word of Y or M with one bit of X or Q. The partial products are then added to the running total Z in carry-save redundant form using the pair of $3:2$ carry-save adders (CSAs).

The critical path for the R2 involves one AND gate and two CSAs. In the custom circuit implementation, the late input is provided to the fastest input of the CSAs. The X and Q inputs see a fanout of w to drive all the AND gates.

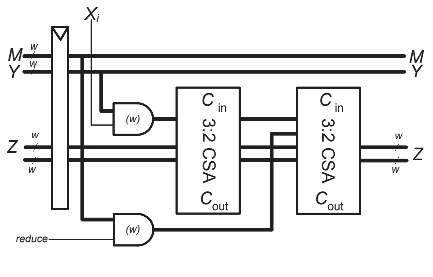

Fig. 3. R2 Block Diagram

B. Radix-4 (R4)

Figure 4 shows a parallelized radix-4 left shifting Montgomery multiplier [6]. The two w-bit MUX4 gates compute the partials products of one words of Y or M with two bits of X or Q. For example, the first MUX4 selects from $0, Y, 2Y$, or $3Y$. The partial products are added to the running total Z using the pair of CSAs. Because Q is derived from a previous Z and arrives in redundant form, an XOR gate is necessary to convert it back to non-redundant form to select the appropriate multiple of M. This architecture requires that the $3Y$ and $3M$ multiples be precomputed and passed through the pipeline.

The critical path for the R4 involves one MUX4 and two CSAs, which is slightly longer than the R2 . Again, the X and Q inputs have a fanout of w.

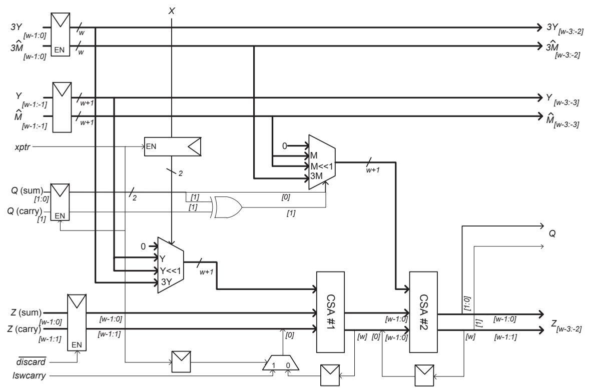

Fig. 4. R4 Block Diagram

C. Booth-Encoded Radix-4 (R4B)

Figure 5 shows a parallelized Booth-encoded radix-4 left shifting Montgomery multiplier [7]. The Booth-encoding eliminates the need for the $3Y$ and $3M$ multiples at the expense of more complex partial product selection logic. The two BOOTH SELECTORs each contain a w-bit MUX5 that chooses the partial product of one word of Y or M with two bits of X or Q. For example, the first MUX5 selects from $0, Y, -Y, 2Y$, or $-2Y$ based on the value of X. Because Q is derived from a previous Z and arrives in redundant form, a 2-bit carry-propagate adder (CPA) is necessary to convert it back to non-redundant form to select the appropriate multiple of M. The PE contains additional logic to handle corner cases related to Booth-encoding, which adds an AND gate and two MUX2s to the critical path.

The critical path for the R4B involves a BOOTH-ENCODER, a MUX5, an AND gate, and two CSA-MUX2 pairs. Again, the X and Q inputs have a fanout of w.

The R4B is the most complex of the architectures. However it is able to process two bits of X at a time without the need for the $3Y$ and $3M$ multiples.

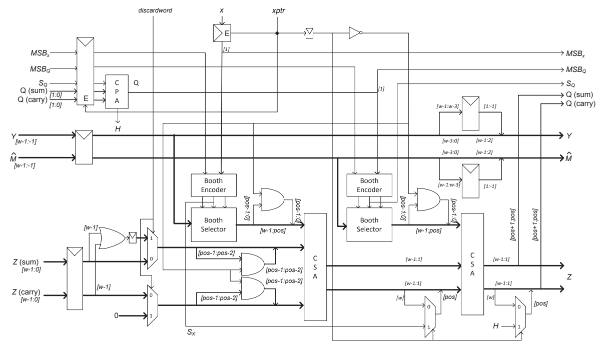
Fig. 5. R4B Block Diagram

IV. Silicon Implementation

The three 16−bit PEs were custom designed in Virtuoso and fabricated though MOSIS in a commercial 130nm process. Using a $1.2V$ supply, the process has an FO4 delay of $44ps$. A $4\times$ inverter consumes $3.7fJ$ per switch. The chip contains a testfixture to measure the delay, and energy of each processing element.

A. Testfixture Design

Figure 6 shows the block diagram of the test chip and Figure 7 shows an annotated photomicrograph. To perform high-speed testing using low-speed external inputs, the test chip uses a scan chain and a voltage controlled oscillator (VCO). Each PE receives its own power supply to monitor power consumption. The output of the the 13-stage VCO drives the PEs. To monitor frequency, the clock is divided by 64 and sent off chip.

Figure 8 shows the test circuitry for a PE. A pair of vectors is scanned into input registers. On-chip test circuitry alternates between these vectors at full speed. When an external trigger is applied, the result is synchronously captured by an output register. The result is then transferred to the scan chain and scanned out and compared against expectation to determine whether the PE operates correctly at the internal clock frequency.

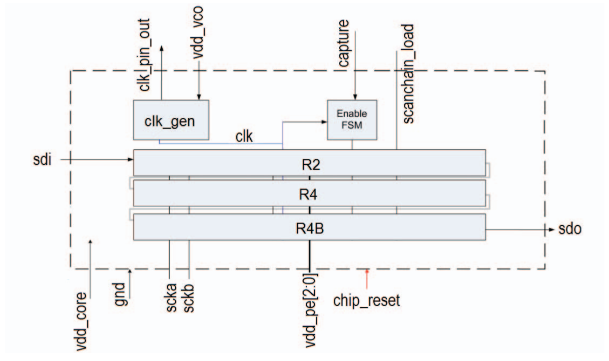

Fig. 6. Chip Block Diagram

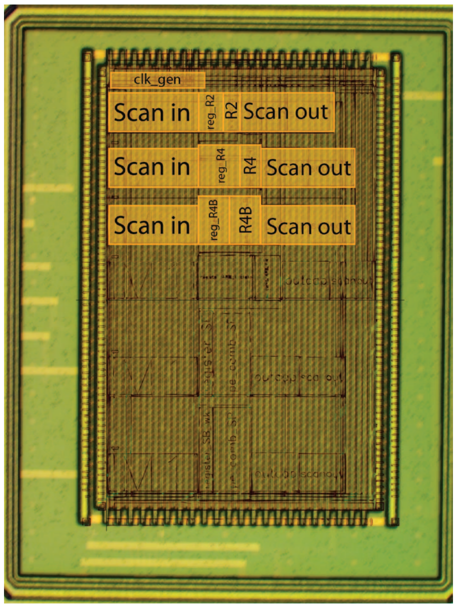

Fig. 7. Annotated Photomicrograph of Test Chip

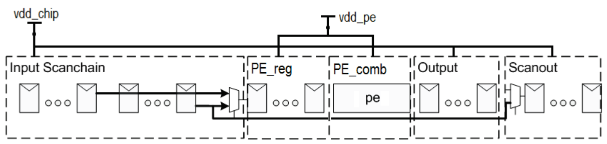

Fig. 8. PE Test Circuitry

B. Test Procedure

We extracted each PE layout from Virtuoso and simulated it in Nanosim to predict energy and maximum operating frequency.

To measure energy on the fabricated chip, a set of ten pseudorandom vector pairs were applied to each PE while the PE supply current was monitored. The measurements were performed at 94 and $440MHz$, and the energy results were consistent. However, the chip was unable to operate reliably above $440MHz$ because of an error in the clock distribution network.

V. Results

Table I lists the energy, delay, and area of each PE. The simulated and measured energy agree to within 11%, giving confidence in the layout extraction and simulation models. The minimum cycle time is only available from simulation. It should be noted that some wires in the R2 and R4B designs are under-driven, and it may be possible to reduce delay by up-sizing some of the registers driving the wires.

Table II shows the energy, delay, and area of a 256×256-bit multiply, using various numbers of PEs. Let $m = vwp$ be a measure of the amount of hardware dedicated to multiplication. The delay decreases and area increases with m. Energy increases only slightly with m because the total amount of computation remains constant. For $m \leq n$, the R4 has 19%

less energy and 20% less area than the R2, but the delay is 22% greater. However, the R4 can benefit from twice as much hardware before performance saturates. The R4B is never superior to the R4, although it eliminates the need for an external CPA to compute the 3× multiples. A pair of 16−bit CPAs has an area of approximately $4200\mu m^2$.

TABLE I
RESULTS OF HARDWARE AND SOFTWARE SIMULATION

	Energy pJ	Sim. Energy pJ	Sim. Delay ns	Area μm^2
R2	3.95	3.79	1.21	11118
R4	6.40	5.89	1.48	17670
R4B	7.50	8.33	1.72	20412

TABLE II
RESULTS WHEN APPLIED TO A 256 BIT × 256 BIT MULTIPLICATION.

PE	p	m	Cycles	Energy nJ	Delay ns	Area mm^2
R2	8	128	594	18.8	719	0.088
	16	256	306	19.3	370	0.177
	32	512	315	39.8	381	0.355
R4	4	128	594	15.2	879	0.070
	8	256	306	15.7	453	0.141
	16	512	171	17.5	253	0.283
R4B	4	128	594	17.8	1020	0.081
	8	256	306	18.4	526	0.163
	16	512	171	20.5	294	0.326

Figure 9 compares the Energy-Delay Products (EDP) of the different architectures. EDP improves with more hardware until saturation because the delay decreases while the energy remains constant. The R2 and R4 architectures have nearly identical EDP, except that the R4 can take advantage of twice as much hardware before delay saturates. The R4B is 36% worse than the R4.

Figure 10 compares the Area-Delay Products (ADP) of the different architectures. ADP is relatively constant with more hardware until saturation because the delay decreases while the area increases. The relative merits are nearly identical to EDP.

Table III compares our delay and ADP results with previously published FPGA implementations [7]. FPGA area is measured in terms of Lookup Tables (LUTs). Interestingly, the FPGA implementations have nearly the same relative frequencies and ADPs as the custom implementations.

TABLE III
COMPARISON BETWEEN FPGA AND SILICON FOR $m \leq n$.

PEs	FPGA Freq MHz	Silicon Freq MHz	LUT-DP ns	ADP mm^2–ns
R2	318	826	48×10^9	66
R4	248	676	47×10^9	64
R4B	186	581	63×10^9	86

VI. CONCLUSION

This paper has quantified the energy, delay, and area of custom silicon implementations of three parallelized left-shifting Montgomery multipliers. The Energy-Delay and Area-Delay Products of the R2 and R4 are comparable. The R2 is simpler and does not require an external carry-propagate adder. If the Montgomery multiplier can run at its own clock frequency, then the R2 is thus preferable. However, if the multiplier frequency is constrained by a fixed system clock, then the R4 delivers the same cycle count at lower energy and area. Moreover, the R4 can take advantage of twice as much hardware before the cycle count saturates.

Compared to the R4, the R4B avoids the need for an external carry-propagate adder at the cost of additional hardware. The extra hardware cost outweighs the benefits, so the R4B is never preferable.

These results are strikingly consistent with FPGA implementations. Thus, FPGAs appear to be a reasonable testbed for comparing these kinds of arithmetic circuits.

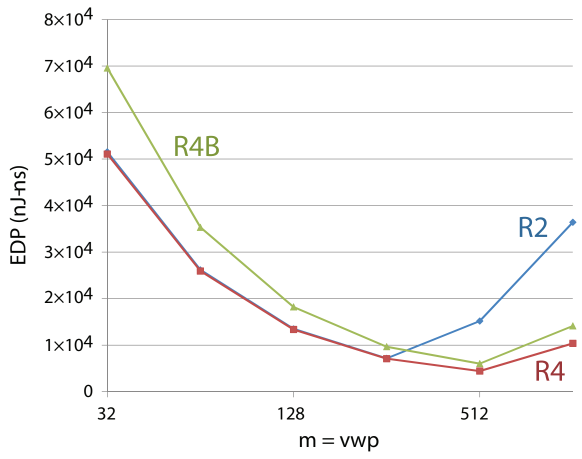

Fig. 9. Montgomery Multiplication EDP vs. Hardware

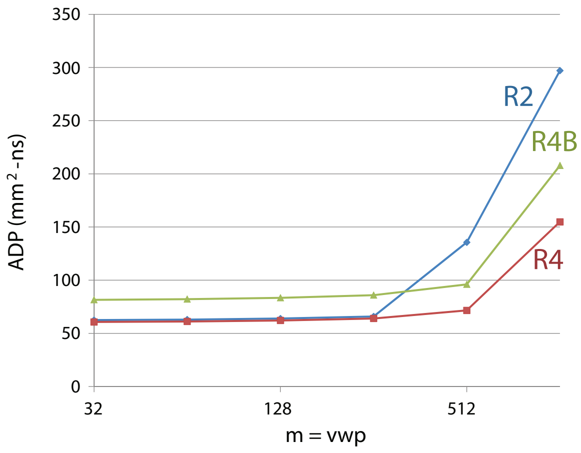

Fig. 10. Montgomery Multiplication ADP vs. Hardware

Acknowledgment

The authors would like to thank the Clay-Wolkin Family Foundation fellowship and Intel Circuit Research Lab for supporting this research.

References

[1] P. Montgomery, "Modular multiplication without trial division," *Math of Computation*, vol. 44, pp. 519–521, April 1985.

[2] H. Orup, "Simplified quotient determination in high-radix modular multiplication," *Proc. 12th IEEE Symp. Computer Arithmetic*, pp. 193–199, July 1995.

[3] K. Kelley and D. Harris, "Parallelized very high radix scalable Montgomery multipliers," *Proc. Asilomar Conf. Signals, Systems, and Computers*, pp. 1196–1200, November 2005.

[4] D. Harris, M. Krishnamurthy, M. Anders, S. Mathew, and S. Hsu, "An improved unified scalable radix-2 Montgomery multiplier," *Proc. 17th IEEE Symp. Computer Arithmetic*, pp. 172–178, 2005.

[5] N. Jiang and D. Harris, "Parallelized radix-2 scalable Montgomery multiplier," *IFIP Intl. Conf. on VLSI*, 2007.

[6] N. Pinckney and D. Harris, "Parallelized radix-4 scalable Montgomery multipliers," *J. Integrated Circuits and Systems*, vol. 3, no. 1, pp. 39–45, March 2008.

[7] N. Pinckney, A. Amberg, and D. Harris, "Parallelized booth-encoded radix-4 scalable Montgomery multipliers," *VLSI SOC*, 2008.