An Optimized Hardware Architecture for Montgomery Multiplication Algorithm

Miaoqing Huang[1], Kris Gaj[2],
Soonhak Kwon[3], Tarek El-Ghazawi[1]

[1] The George Washington University, Washington, D.C., U.S.A.
[2] George Mason University, Fairfax, VA, U.S.A.
[3] Sungkyunkwan University, Suwon, Korea

Motivation

- Fast modular multiplication required in multiple cryptographic transformations
    - RSA, DSA, Diffie-Hellman
    - Elliptic Curve Cryptosystems
    - ECM, p-1, Pollard's rho methods of factoring, etc.

- Montgomery Multiplication invented by Peter L. Montgomery in 1985 is most frequently used to implement repetitive sequence of modular multiplications in both software and hardware

- Montgomery Multiplication in hardware replaces division by a sequence of simple logic operations, conditional additions and right shifts

Montgomery Modular Multiplication (1)

$Z = X \cdot Y \bmod M$ X, Y, M – n-bit numbers

Integer domain **Montgomery domain**

$X \longrightarrow X' = X \cdot 2^n \bmod M$

$Y \longrightarrow Y' = Y \cdot 2^n \bmod M$

$$Z' = MP(X', Y', M) =$$
$$= X' \cdot Y' \cdot 2^{-n} \bmod M =$$
$$= (X \cdot 2^n) \cdot (Y \cdot 2^n) \cdot 2^{-n} \bmod M =$$
$$= X \cdot Y \cdot 2^n \bmod M$$

$Z = X \cdot Y \bmod M \longleftarrow Z' = Z \cdot 2^n \bmod M$

Montgomery Modular Multiplication (2)

$X \longrightarrow X'$

$X' = MP(X, 2^{2n} \bmod M, M)$

$Z \longleftarrow Z'$

$Z = MP(Z', 1, M)$

Basic version of the Radix-2 Montgomery Multiplication Algorithm

Algorithm 1 Radix-2 Montgomery Multiplication

Require: odd $M, n = \lfloor \log_2 M \rfloor + 1, X = \sum_{i=0}^{n-1} x_i \cdot 2^i$, with $0 \leq X, Y < M$

Ensure: $Z = MP(X, Y, M) \equiv X \cdot Y \cdot 2^{-n} \pmod{M}, 0 \leq Z < M$

1: $S[0] = 0$
2: **for** $i = 0$ to $n - 1$ step 1 **do**
3: $\quad q_i = S[i] + x_i \cdot Y \pmod 2$
4: $\quad S[i+1] = (S[i] + x_i \cdot Y + q_i \cdot M) \text{ div } 2$
5: **end for**
6: **if** $(S[n] > M)$ **then**
7: $\quad S[n] = S[n] - M$
8: **end if**
9: return $Z = S[n]$

Classical Design by Tenca & Koc CHES 1999

Multiple Word Radix-2 Montgomery Multiplication algorithm (MWR2MM)

Main ideas:

Use of short precision words (w-bit each):

- Reduces broadcast problem in circuit implementation
- Word-oriented algorithm provides the support needed to develop scalable hardware units.

Operand Y(multiplicand) is scanned word-by-word,
operand X(multiplier) is scanned bit-by-bit.

Classical Design by Tenca & Koc CHES 1999

Each word has w bits

Each operand has

n bits

e words

$$e = \left\lceil \frac{n+1}{w} \right\rceil$$

$X = (x_{n-1}, \ldots, x_1, x_0)$

$Y = (Y^{(e-1)}, \ldots, Y^{(1)}, Y^{(0)})$

$M = (M^{(e-1)}, \ldots, M^{(1)}, M^{(0)})$

The bits are marked with subscripts, and the words are marked with superscripts.

MWR2MM
Multiple Word Radix-2 Montgomery Multiplication algorithm by Tenca and Koc

Algorithm 2 The Multiple-Word Radix-2 Montgomery Multiplication Algorithm

Require: odd $M, n = \lfloor \log_2 M \rfloor + 1$, word size w, $e = \lceil \frac{n+1}{w} \rceil$, $X = \sum_{i=0}^{n-1} x_i \cdot 2^i$, $Y = \sum_{j=0}^{e-1} Y^{(j)} \cdot 2^{w \cdot j}$, $M = \sum_{j=0}^{e-1} M^{(j)} \cdot 2^{w \cdot j}$, with $0 \leq X, Y < M$

Ensure: $Z = \sum_{j=0}^{e-1} S^{(j)} \cdot 2^{w \cdot j} = MP(X, Y, M) \equiv X \cdot Y \cdot 2^{-n} \pmod{M}, 0 \leq Z < 2M$

1: $S = 0$ — *initialize all words of S*
2: **for** $i = 0$ to $n - 1$ step 1 **do**
3: $q_i = (x_i \cdot Y_0^{(0)}) \oplus S_0^{(0)}$
4: $(C^{(1)}, S^{(0)}) = x_i \cdot Y^{(0)} + q_i \cdot M^{(0)} + S^{(0)}$

Task A

5: **for** $j = 1$ to $e - 1$ step 1 **do**
6: $(C^{(j+1)}, S^{(j)}) = C^{(j)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)} + S^{(j)}$
7: $S^{(j-1)} = (S_0^{(j)}, S_{w-1..1}^{(j-1)})$

Task B **e-1 times**

8: **end for**
9: $S^{(e-1)} = (C_0^{(e)}, S_{w-1..1}^{(e-1)})$

Task C

10: **end for**
11: return $Z = S$

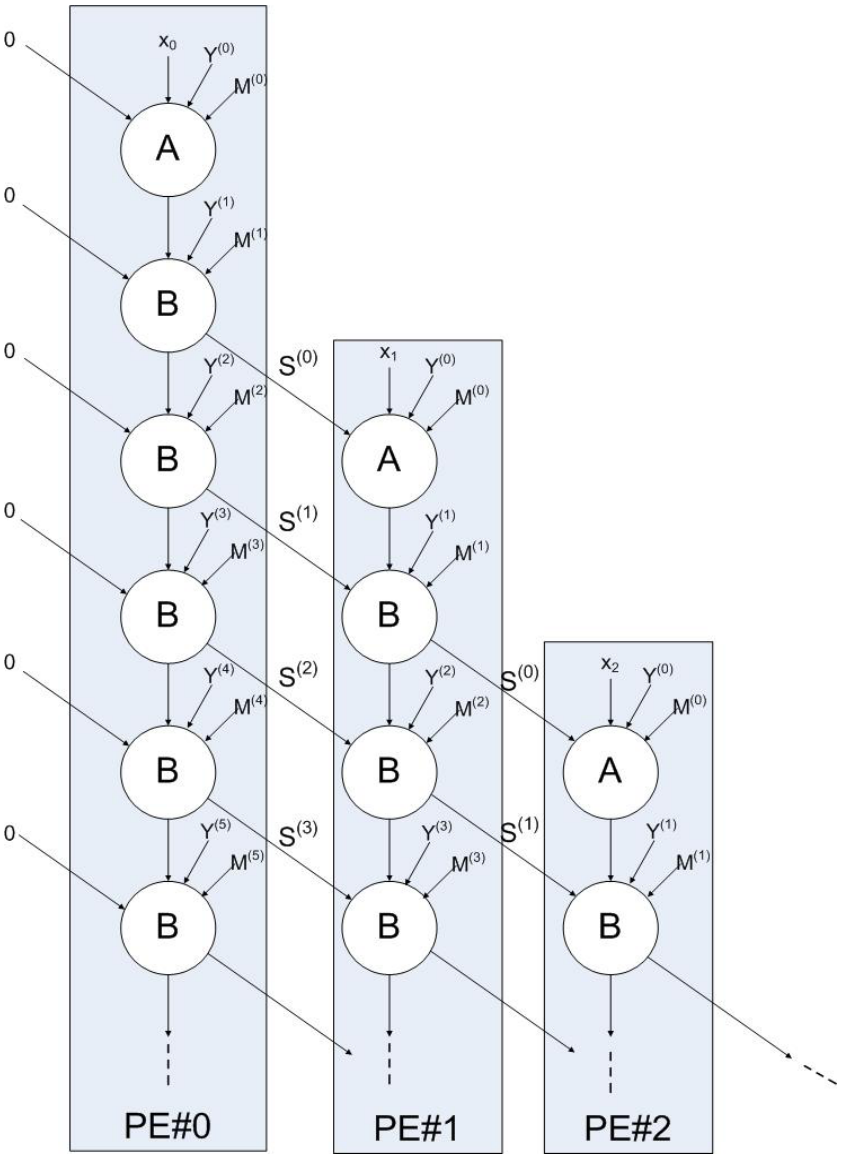
Data Dependency Graph for MWR2MM by Tenca & Koc

- One PE is in charge of the computation of one column that corresponds to the updating of *S* with respect to one single bit x_i.
- The delay between two adjacent PEs is 2 clock cycles.
- The minimum computation time is

  $2 \bullet n + e - 1$ clock cycles
- given

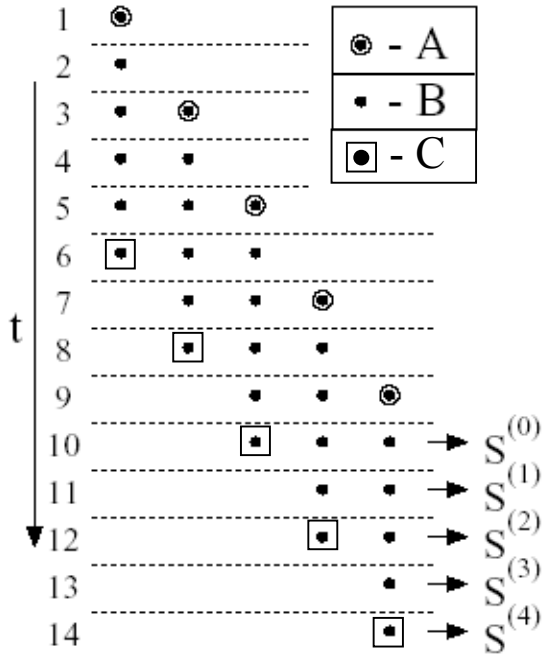
  $(e+1)/2$ PEs

  working in parallel.



Example of operation of the design by Tenca & Koc

Example of the computation executed for 5-bit operands with word-size w = 1 bit



n = 5

w = 1
e = 5

2n + e – 1 =
2·5 + 5 – 1 = 14 clock cycles

(e+1)/2 =
(5+1)/2 = 3 PEs
sufficient to perform all computations

Main idea of the new architecture

- In the architecture of Tenca & Koc
  - w-1 least significant bits of partial results $S^{(j)}$ are available one clock cycle before they are used
  - only one (most significant) bit is missing
- Let us compute a new partial result under two assumptions regarding the value of the most significant bit of $S^{(j)}$ and choose the correct value one clock cycle later

Pseudocode of the Main Processing Element

Algorithm 3 Pseudocode of the processing element PE#j of type E

Require: Inputs: q_i, x_i, $C^{(j)}$, $Y^{(j)}$, $M^{(j)}$, $S_0^{(j+1)}$

Ensure: Output: $C^{(j+1)}$, $S_0^{(j)}$

1: $(CO^{(j+1)}, SO_{w-1}^{(j)}, S_{w-2..0}^{(j)}) = (1, S_{w-1..1}^{(j)}) + C^{(j)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)}$

2: $(CE^{(j+1)}, SE_{w-1}^{(j)}, S_{w-2..0}^{(j)}) = (0, S_{w-1..1}^{(j)}) + C^{(j)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)}$

3: **if** $(S_0^{(j+1)} = 1)$ **then**

4: $\quad C^{(j+1)} = CO^{(j+1)}$

5: $\quad S^{(j)} = (SO_{w-1}^{(j)}, S_{w-2..0}^{(j)})$

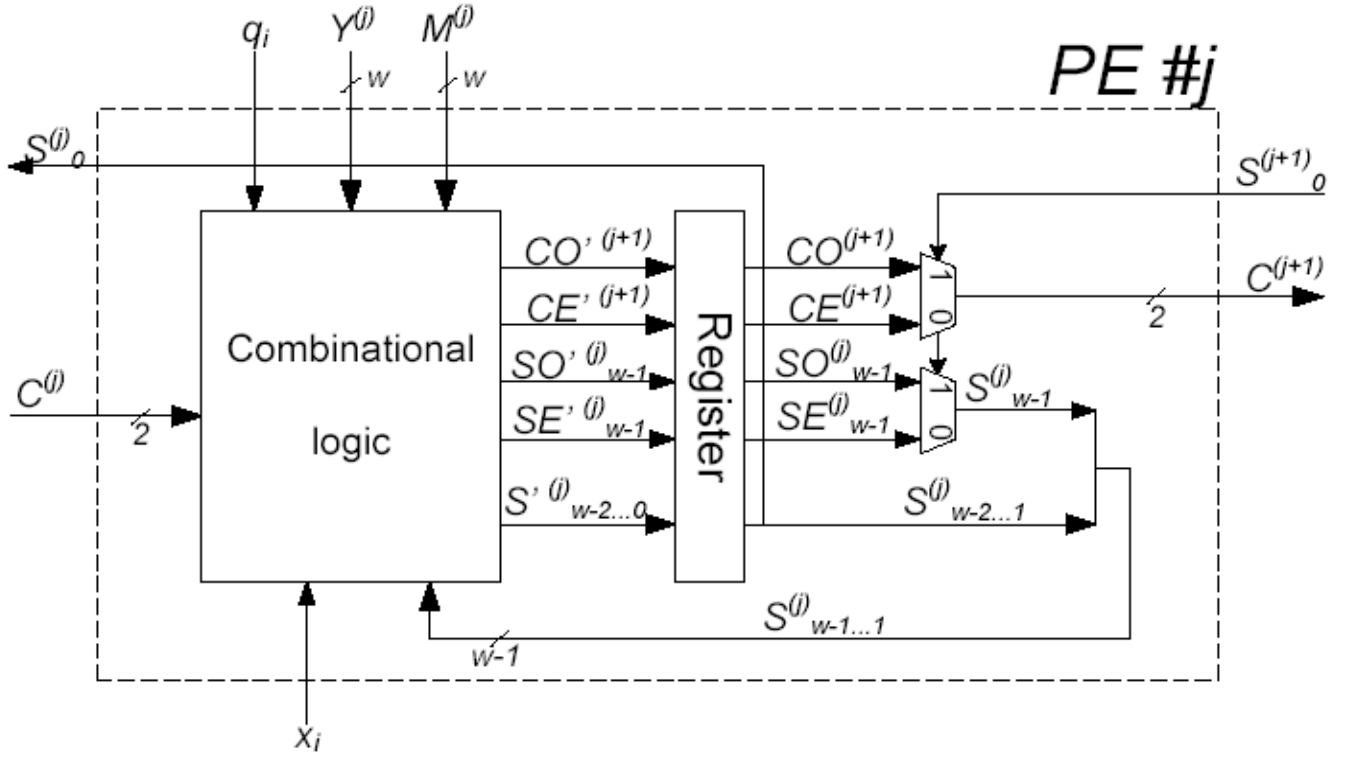
6: **else**

7: $\quad C^{(j+1)} = CE^{(j+1)}$

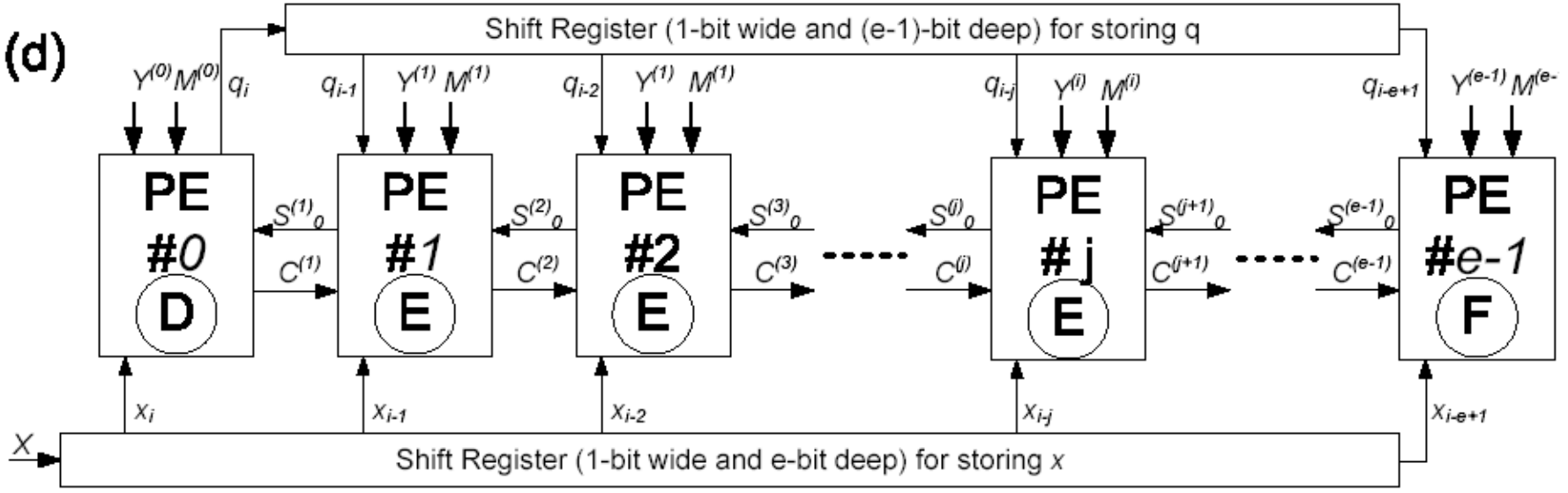
8: $\quad S^{(j)} = (SE_{w-1}^{(j)}, S_{w-2..0}^{(j)})$

9: **end if**

Main Processing Element
Type E

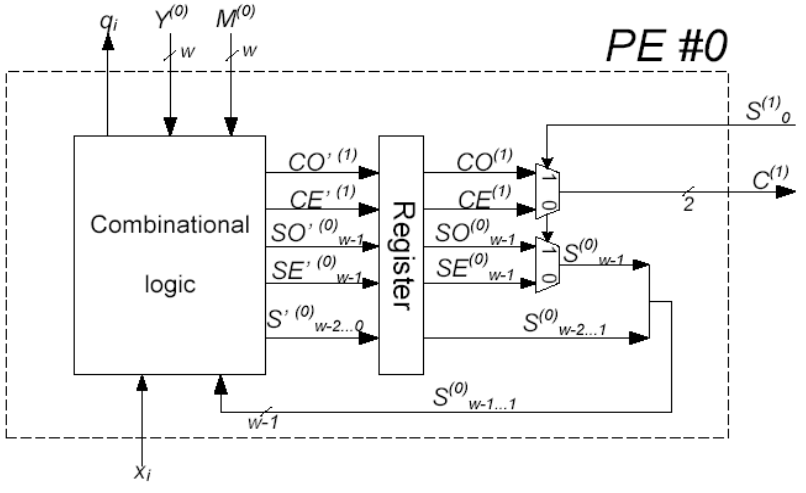


The Proposed Optimized Hardware Architecture

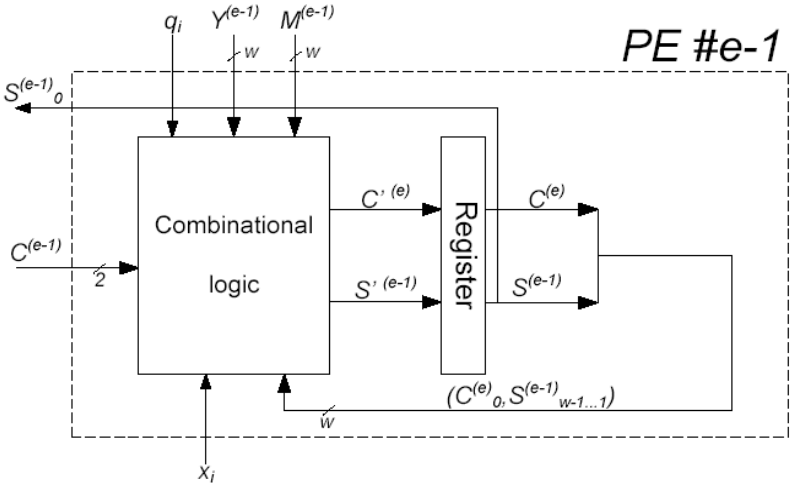


The First and the Last Processing Elements

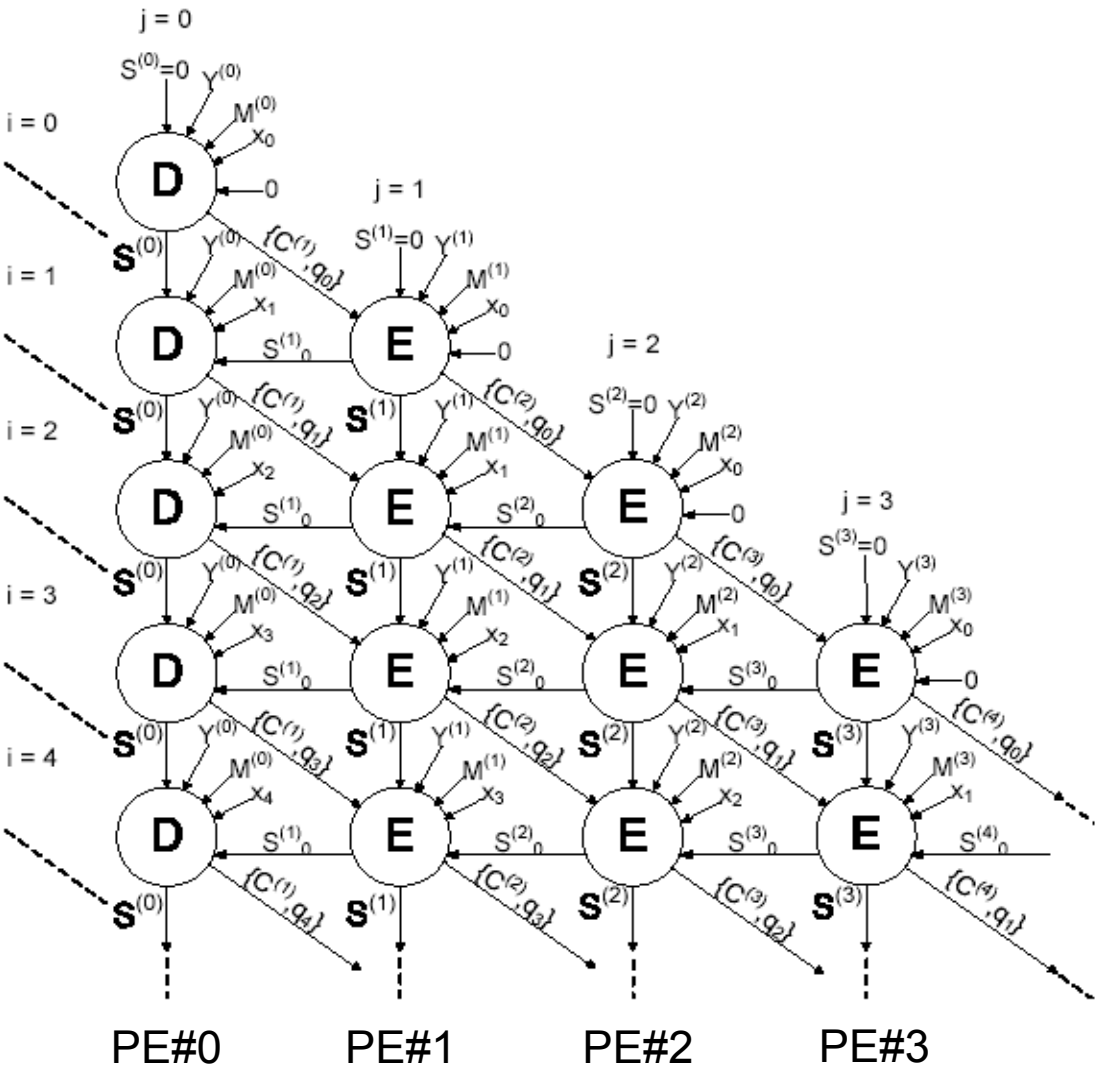
Type D



Type F



Data Dependency Graph of the Proposed New Architecture



Conceptual Comparison with Earlier Designs

	Tenca *et al.* [4–6]		Our Architecture		McIvor *et al.* [7]
	Radix-2	Radix-4	Radix-2	Radix-4	Radix-2
Minimum number of PEs minimizing circuit area	1	1	e	e	1
Optimal number of PEs minimizing circuit latency	$\lceil \frac{e+1}{2} \rceil$	$\lceil \frac{e+1}{2} \rceil$	e	e	1
Latency in clock cycles	$2 \cdot n + e - 1$	$n + e - 1$	$n + e - 1$	$\frac{n}{2} + e - 1$	$n + 1^*$

* *The result is in the Carry Save form.*

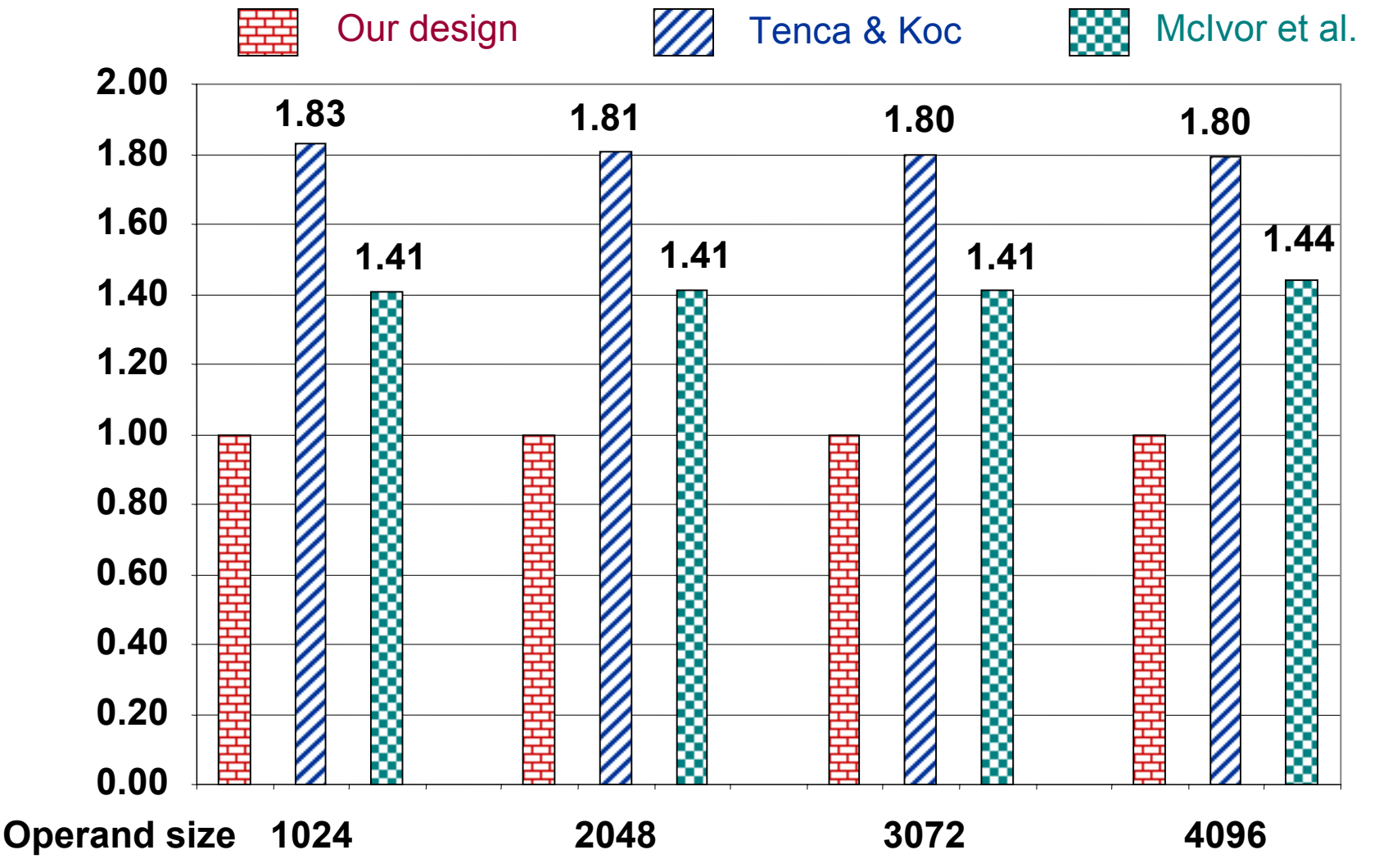
4. Tenca, A. and Koç, Ç. K.: A scalable architecture for Montgomery multiplication, *CHES 99, Lecture Notes in Computer Sciences*, **1717**:94–108, 1999
5. Tenca, A., Todorov, G., and Koç, Ç. K.: High-radix design of a scalable modular multiplier, *CHES 2001, Lecture Notes in Computer Sciences*, **2162**:185–201, 2001
6. Tenca, A. and Koç, Ç. K.: A scalable architecture for modular multiplication based on Montgomery's algorithm, *IEEE Trans. Computers*, **52(9)**:1215–1221, 2003
7. McIvor, C., McLoone, M. and McCanny, J.V.: Modified Montgomery Modular Multiplication and RSA Exponentiation Techniques *IEE Proceedings – C Computers & Digital Techniques*, **151(6)**:402-408, 2004

Quantitative Comparison for the Implementation Using Xilinx Virtex-II 6000 FF1517-4 FPGA

Target Clock Frequency 100 MHz
Experimental testing using SRC-6 Reconfigurable Computer

		1024-bit	2048-bit	3072-bit	4096-bit
Our Proposed Architecture	Slices Utilization	4,178(12%)	8,337(24%)	12,495(36%)	16,648(49%)
	Quantity of PEs	65	129	193	257
	Latency (clocks)	1088	2176	3264	4352
Architecture of Tenca & Koç [4]	Slices Utilization	3,937(11%)	7,756(22%)	11,576(34%)	15,393(45%)
	Quantity of PEs	33	65	97	129
	Latency (clocks)	2113	4225	6337	8449
Architecture of McIvor *et al.* [7]	Slices Utilization	6,241(18%)	12,490(36%)	18,728(55%)	25,474(75%)
	Latency (clocks)	1025	2049	3073	4097

Normalized Product Latency Times Area
New Architecture vs. Previous Architectures



Conclusions

- New optimized architecture for the word-based Montgomery Multiplier

Compared to the classical design by Tenca & Koc:

- Minimum latency smaller by a factor of almost 2, in terms of both clock cycles and absolute time units
- Comparable circuit area for minimum latency
- Improvement in terms of the product of latency times area by a factor of about 1.8
- Reduced scalability (fixed vs. variable number of processing elements required for the given operand size)

Compared to the newer design by McIvor et al.:

- Comparable latency
- Area smaller by at least 33%
- Improvement in terms of the product of latency times area by a factor of about 1.4
- Similar scalability