Montgomery Multiplication

David Harris and Kyle Kelley

Harvey Mudd College

Claremont, CA 91711

{David_Harris, Kyle_Kelley}@hmc.edu

Outline

- Cryptography Overview
- Finite Field Mathematics
- Montgomery Multiplication
- Tenca-Koç Montgomery Multiplier
- Improved Montgomery Multiplier
- Very High Radix
- Implementation Results
- Summary

Cryptography Overview

- Encryption has become essential
  - E-commerce (SSL)
  - Communications / network processors
  - Smart cards / digital cash
  - Military
- Two major classes of algorithms
  - *Symmetric cryptosystems* (e.g. DES)
  - *Public key cryptosystems* (e.g. RSA)

Cryptographic Protocols

- Alice and Bob would like to communicate securely. Eve wants to listen in.
  - Symmetric key:
    - Alice and Bob must share a key for encryption and decryption.
    - If Eve hears it, she can read the messages.
  - Public key:
    - Alice publishes her public key to the world.
    - Bob encrypts with Alice's public key.
    - Alice can decrypt only with her private key.
    - Eve can't decrypt with the public key.

Digital Signatures

- Alice wants to sign a contract in a way that only she can do.
  - Alice publishes her public key and keeps the private key secret.
  - Encrypt the document with her secret key.
  - Anyone can decrypt the document with her public key
  - But nobody can forge her signature.

Key Exchange

- Public key encryption is slow
- Use it to share a symmetric key
  - Use symmetric key to encrypt large blocks of data

RSA Encryption

- Most widely used public key system.
  - Good for encryption and signatures.
  - Invented by Rivest, Shamir, Adleman (1978)
- Public *e* and private *d* keys are long #s
  - n = 256-2048^{+} bits
  - Satisfy $x^{de} \bmod M = 1$ for all x
  - Finding *d* from *e* is as hard as factoring *M*
- Encryption: $B = A^{e} \bmod M$
- Decryption: $C = B^{d} \bmod M = A^{ed} = A$

Modular Exponentiation

- Critical operation in RSA and for
  - Digital signature algorithm
  - Diffie-Hellman key exchange
  - Elliptic curve cryptosystems
- Done with 2*n* modular multiplications
  - Ex: $A^{27} = (((((A^2) * A)^2)^2) * A)^2) * A$
  - Division required after each multiplication to compute modulo

Finite Field Mathematics

- +, * modulo prime *p* form a finite field
  - *p* elements
  - Additive identity: 0
  - Multiplicitive identity: 1
  - Each nonzero number has a unique inverse x^{-1}
  - Named GF(*p*)
    - For Evariste Galois, a 19th century number theorist killed in a duel at age 20

Binary Extension Fields

- Building blocks are polynomials in x
  - Operations performed modulo some irreducible polynomial $f(x)$ of degree n
  - Arithmetic done modulo 2
  - Called GF(2^n)
- Example: GF(2^3)
- Computation is the same as GF(p)
  - Except that no carries are propagated

Element	Code
0	000
1	001
x	010
$x+1$	011
x^2	100
x^2 +1	101
x^2+x	110
x^2+x+1	111

Montgomery's Algorithm

Multiply: $Z = X \times Y$

Reduce: $q = Z \times M' \bmod R$

$Z = [Z + q \times M] \,/\, R$

Normalize: if $Z \geq M$ then $Z = Z - M$

- M' satisfies $RR^{-1} - MM' = 1$
  - Drives LSBs to 0

Montgomery Multiplication

- Faster way to do modular exponentation
  - Operate on *Montgomery residues*
  - Division becomes a simple shift
  - Requires conversion to and from residues only once per exponentiation

Montgomery Residues

- Let the modulus *M* be an odd *n*-bit integer

  $$2^{n-1} < M < 2^n$$

- Define $r = 2^n$
- Define the M-residue of an integer $a < M$ as

  $$\bar{a} = ar \bmod M$$

- There is a one-to-one correspondence between integers and M-residues for

  $0 < a < M-1$

M-Residue Examples

- $M = 11$, $r = 16$

$$\overline{0} = 0*16 \bmod 11 = 0$$
$$\overline{1} = 1*16 \bmod 11 = 5$$
$$\overline{2} = 2*16 \bmod 11 = 10$$
$$\overline{3} = 3*16 \bmod 11 = 4$$
$$\overline{4} = 4*16 \bmod 11 = 9$$
$$\overline{5} = 5*16 \bmod 11 = 3$$
$$\overline{6} = 6*16 \bmod 11 = 8$$
$$\overline{7} = 7*16 \bmod 11 = 2$$
$$\overline{8} = 8*16 \bmod 11 = 7$$
$$\overline{9} = 9*16 \bmod 11 = 1$$
$$\overline{10} = 10*16 \bmod 11 = 6$$

Montgomery Multiplicaton

- Define

$$\bar{z} = MM(\bar{x}, \bar{y}) = \bar{x}\bar{y}r^{-1} \bmod M$$

- Where r^{1} is the inverse of r mod M:
  - $r^{1}r = 1$ (mod M)
- This gives the Montgomery residue of
  - $z = xy$ mod M

$$\begin{aligned}\bar{z} &= \bar{x}\bar{y}r^{-1} \bmod M \\ &= (xr)(yr)r^{-1} \bmod M \\ &= xyr \bmod M \\ &= zr \bmod M\end{aligned}$$

Mont. Multiplication Example

$$r^{-1} = 9 \quad (16 * 9 \mod 11 = 1)$$

$$MM(5,7) = 5 * 7 * 9 \mod 11 = 7$$

- It may not be obvious that this is easier to do than regular modular multiplication.

Montgomery Multiplier

- **MM is an easier operation that requires no hard division, just shifting**
- **In radix 2,**

```
Z = 0
for i = 0 to n-1
  Z = Z + x_i•Y
  if Z is odd then Z = Z + M
  Z = Z/2
if Z ≥ M then Z = Z – M
```

Example

- $X = 7 = 0111$
- $Y = 5 = 0101$
- $M = 11 = 1011$

```
Z = 0
for i = 0 to n-1
      Z = Z + xi•Y
      if Z is odd then Z = Z + M
      Z = Z/2
if Z ≥ M then Z = Z - M
```

- Z initially 0
  - Z = (0 + 5 + 11) / 2 = 8
  - Z = (8 + 5 + 11) / 2 = 12
  - Z = (12 + 5 + 11) / 2 = 14
  - Z = (14 + 0) / 2 = 7 (final result)

Conversion

- Conversion of integers to/from Montgomery residues takes one MM operation (if r^2 mod M is precomputed and saved):

$$\bar{x} = MM(x, r^2) = xr^2 r^{-1} \bmod M = xr \bmod M$$

$$x = MM(\bar{x}, 1) = \bar{x} 1 r^{-1} \bmod M = xr 1 r^{-1} \bmod M = x$$

- Modular exponentiation takes two conversion steps and 2*n* multiplication steps.

Cryptography Accelerators

- Hardware accelerators offer more speed at less power than software
  - Via announced x86 C5J core Montgomery Multiply opcode (May 04)

3COM Router 5000 Series Encryption Accelerator

IBM PCI SSL Cryptography Accelerator

Break

Break

Break

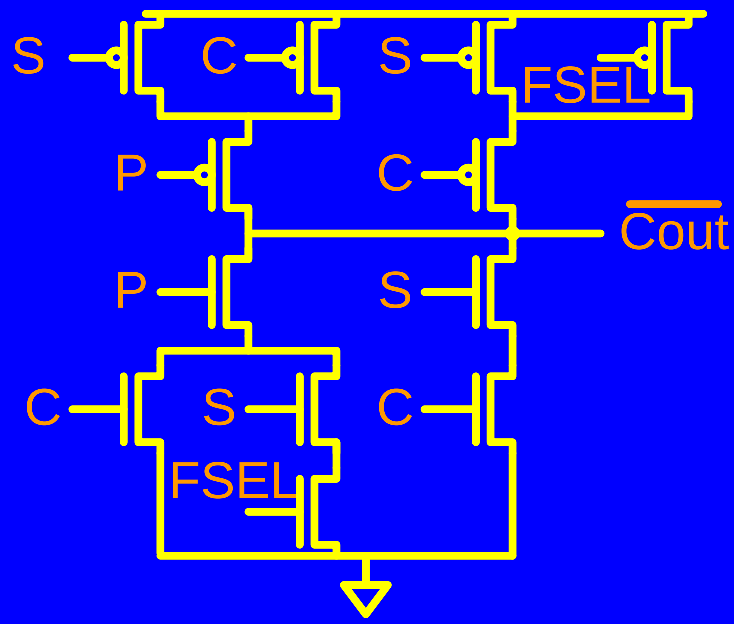

ALABAMA
Montgomery
© 1992 MAGELLAN Geographix, Santa Barbara, CA

Reconfigurable Hardware

- Building hardwired n-bit unit is limiting
  - Slow for large n
  - Not scalable to different n
- Better to design for w-bit words
  - Break n-bit operand into e w-bit words
  - This is called *scalable*
- Also handle both GF(p) and GF(2^n)
  - Requires conditionally killing carries
  - Called *unified*

Unified Carry Gate

- Full adder modified for dual-field ops
  - fsel = 1: normal operation GF(p)
  - fsel = 0: kill carry GF(2^n)
- Only changes majority gate
- Sum remains XOR



Tenca-Koç Montgomery Multiplier

$Z = 0$

for $i = 0$ to $n-1$

  $(C^A, Z_{w-1:0}) = Z_{w-1:0} + X_i \times Y_{w-1:0}$

  $reduce = Z_0$

  $(C^B, Z_{w-1:0}) = Z_{w-1:0} + reduce \times M_{w-1:0}$

  for $j = 1$ to $e+1$

    $(C^A, Z_{(j+1)w-1:jw}) = Z_{(j+1)w-1:jw} + X_i \times Y_{(j+1)w-1:jw} + C^A$

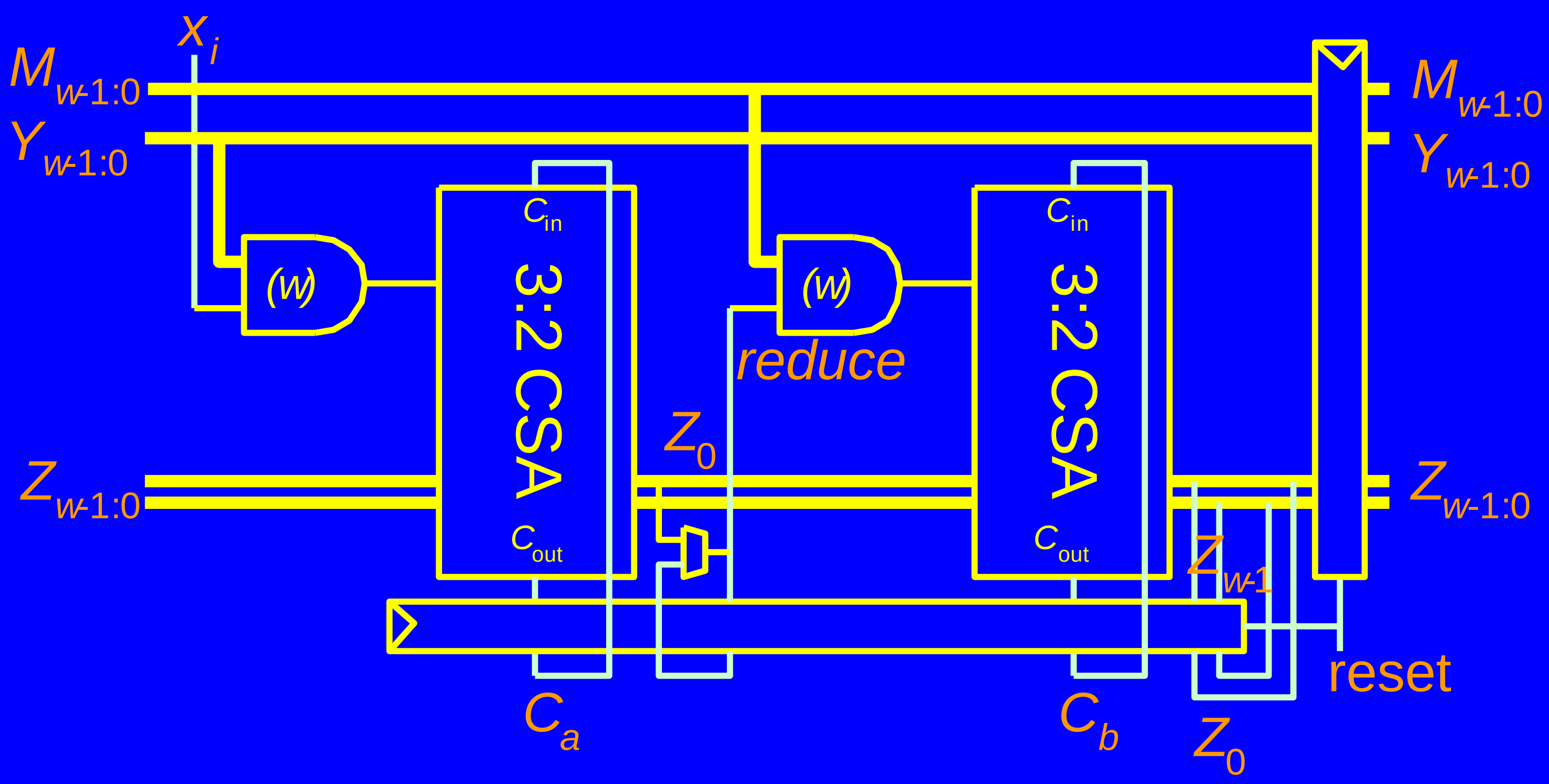
    $(C^B, Z_{(j+1)w-1:jw}) = Z_{(j+1)w-1:jw} + reduce \times M_{(j+1)w-1:jw} + C^B$

    $Z_{jw-1:(j-1)w} = (Z_{jw}, Z_{jw-1:(j-1)w+1})$

IEEE Transactions on Computers, Sept. 2003

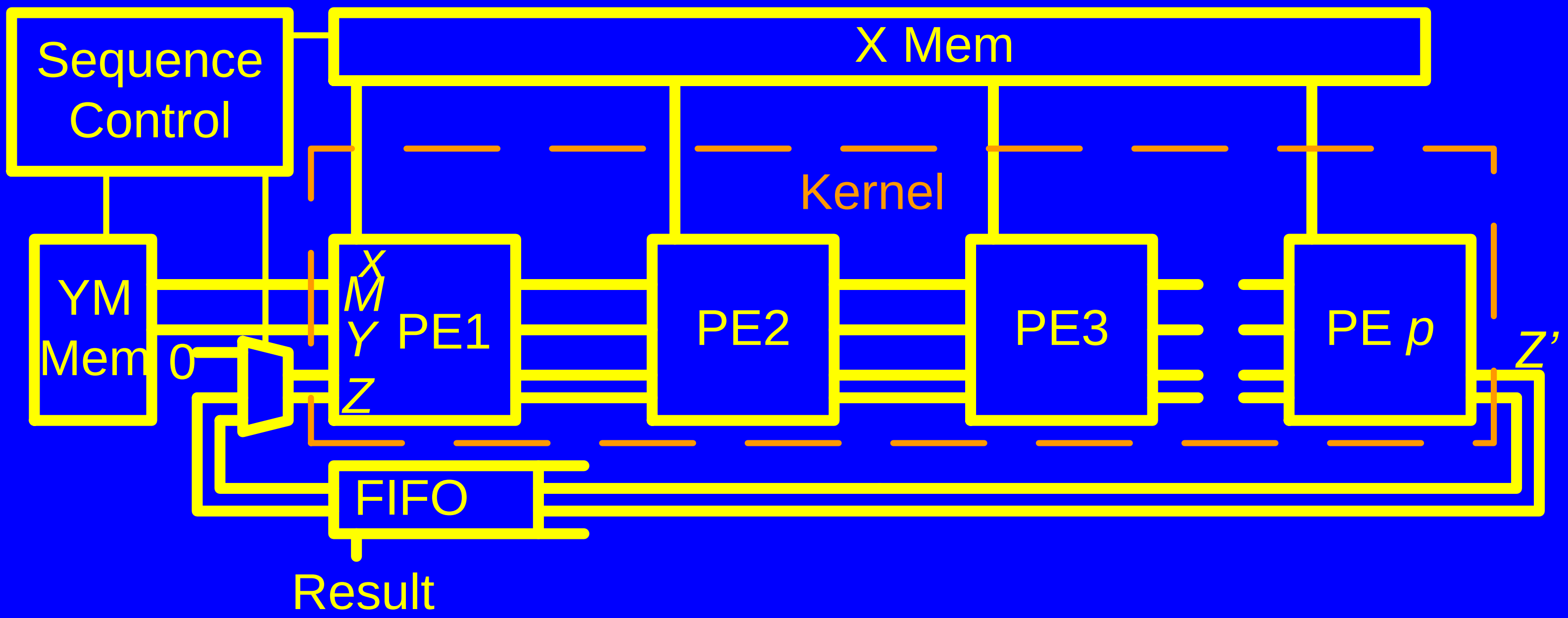
Processing Elements

- Keep *Z* in carry-save redundant form
- Simple processing element (PE)

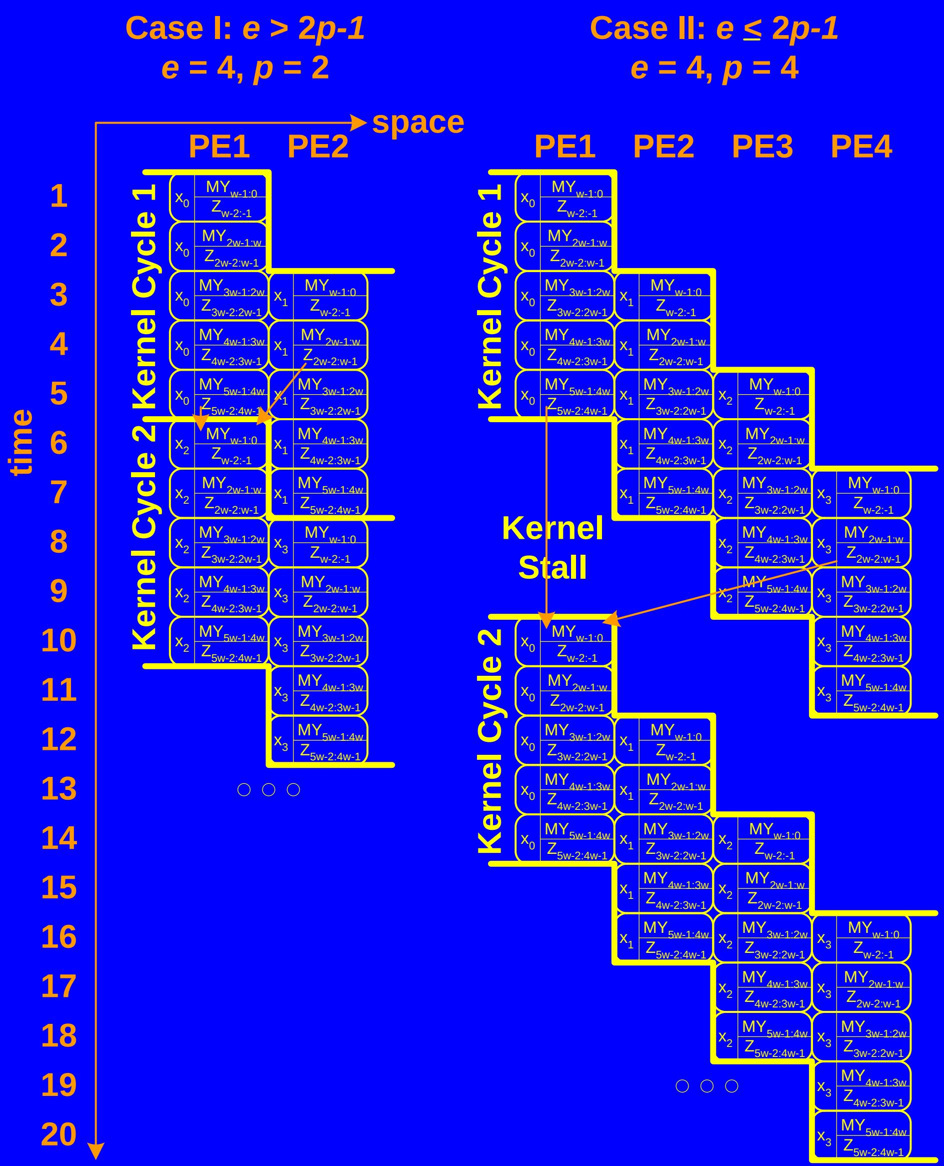


Parallelism

- **Two dimensions of parallelism:**
  - **Width of processing element *w***
  - **Number of pipelined PEs *p***
    - **Multiply takes $k = n/p$ kernel cycles**

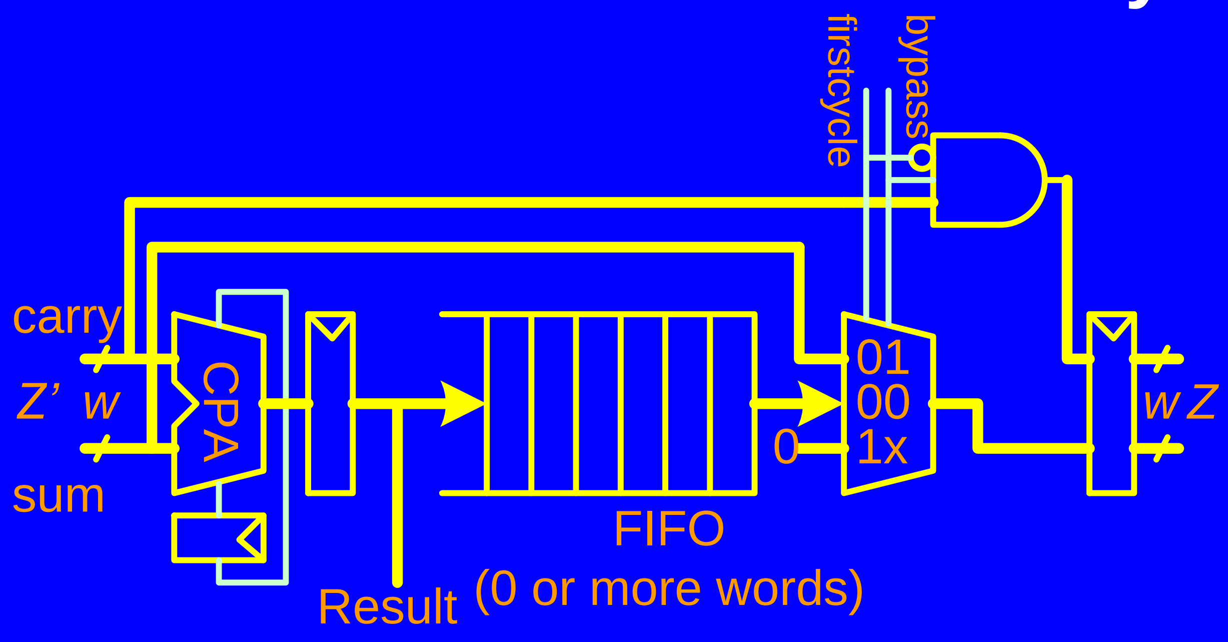


Pipeline Timing



Queue

- If full PEs cause stall, queue results
- Convert back to nonredundant form
  - Saves queue space
  - CPA needed for final result anyway

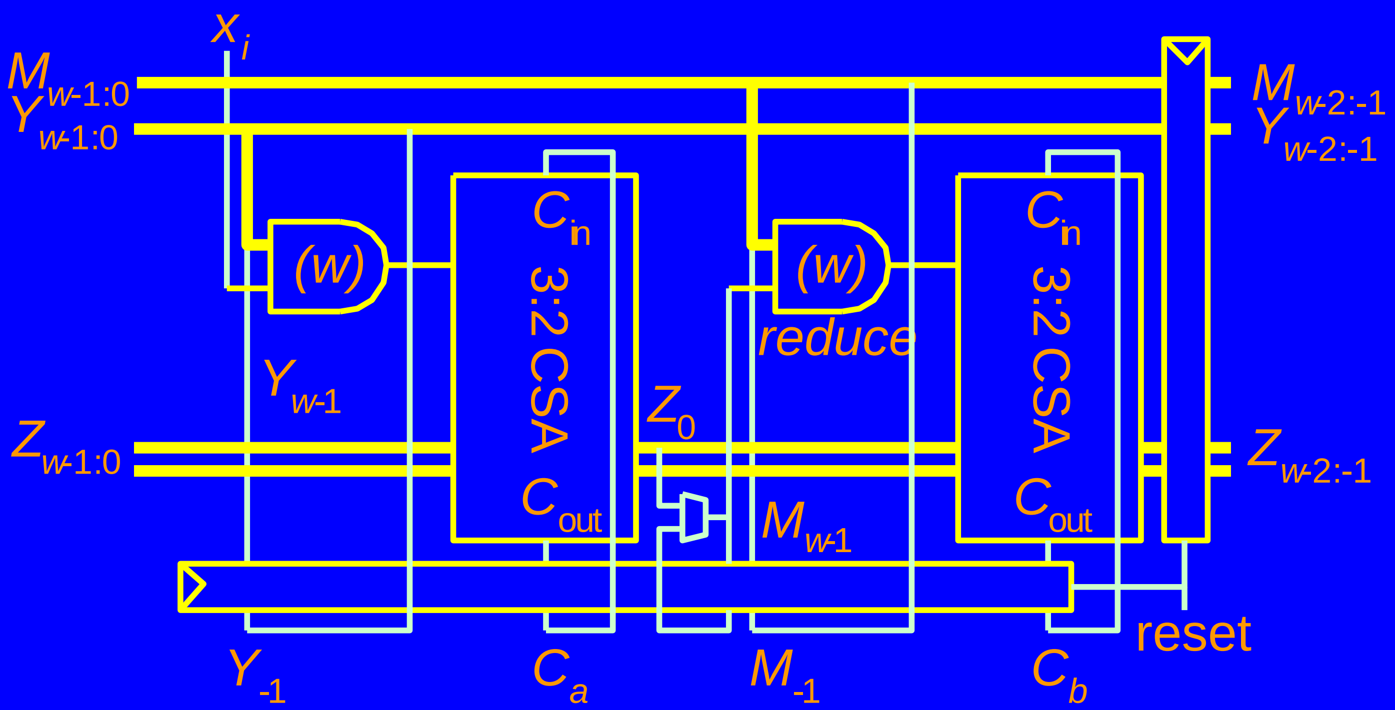


Improved Design

- Don't wait two cycles for MSB
- Kick off dependent operation right away on the available bits
- Take extra cycle(s) at the end to handle the extra bits
- For p processing elements, cycle count reduces from $2p$ to $p + (p/w)$

Improved PE

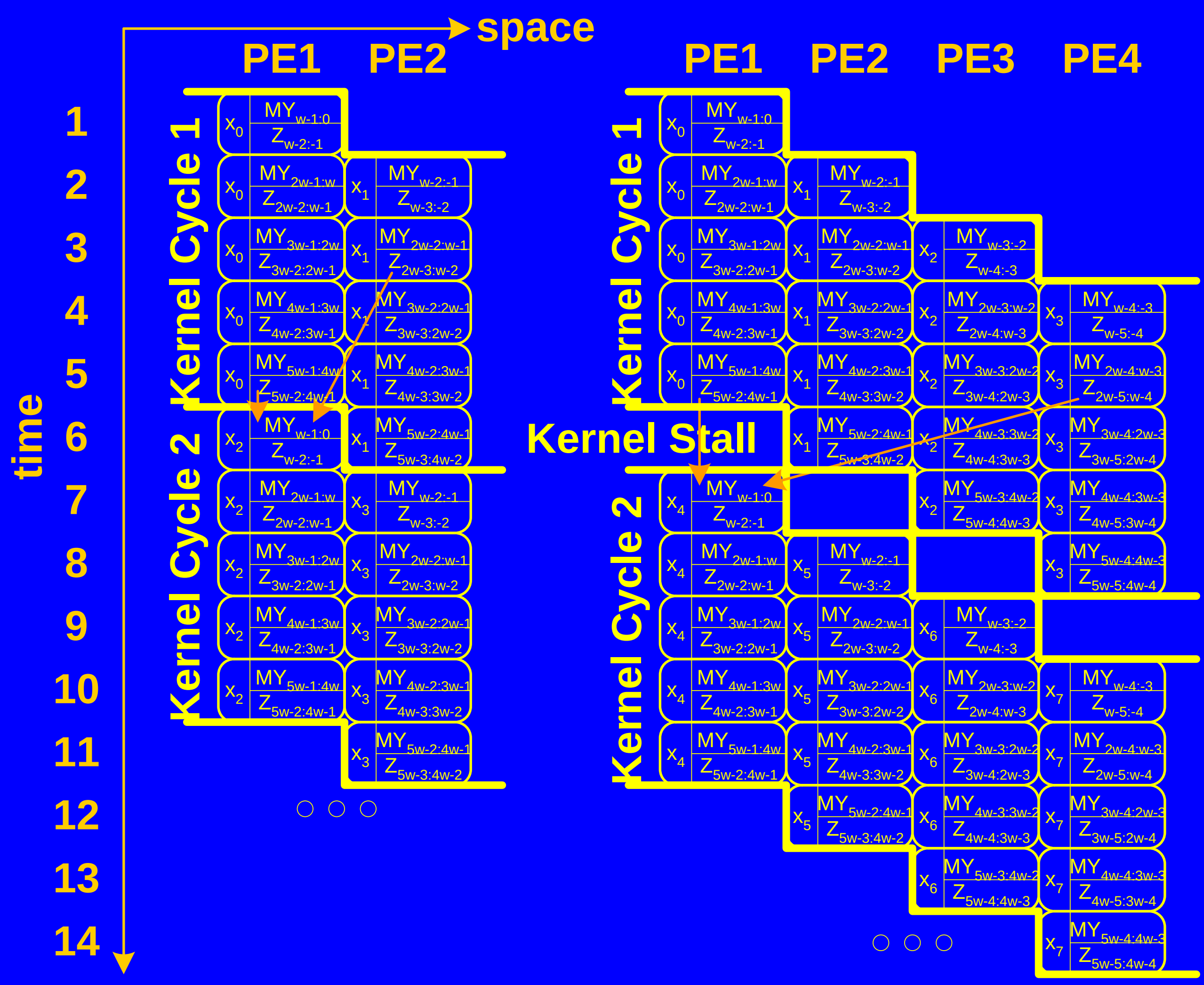
- Left-shift *M* and *Y* rather than right-shifting *Z*
- Same amount of hardware



Pipeline Timing

Case I: *e* > *p*+1
***e* = 4, *p* = 2**

Case II: *e* ≤ *p*+1
***e* = 4, *p* = 4**



Latency

- **Tenca-Koç**

$$
\begin{array}{lll}
k(e+1)+2(p-1) & e > 2p-1 & \text{(Case I)} \\
k\left(2p+1\right)+e-2 & e \le 2p-1 & \text{(Case II)}
\end{array}
$$

- **Improved Design**

$$
\begin{array}{lll}
\left(k+1\right)e+p-1 & e > p+1 & \text{(Case I)} \\
k\left(p+1\right)+2e-2 & e \le p+1 & \text{(Case II)}
\end{array}
$$

Very High Radix

- These designs are Radix-2
  - 1 bit of x per PE
- Higher radix designs reduce latency
  - Process more bits of x per PE
  - Require integer multiplication instead of AND gates

Montgomery's Algorithm

Multiply: $Z = X \times Y$

Reduce: $q = Z \times M' \bmod R$

$Z = [Z + q \times M] / R$

Normalize: if $Z \geq M$ then $Z = Z - M$

- M' satisfies $RR^{-1} - MM' = 1$
  - Drives LSBs to 0

Scalable Very High Radix Algorithm

w-bit words of M and Y $\quad e = n/w$

v-bit digits of X $\quad f = n/v \quad$ radix $= 2^v$

$Z = 0$

for $i = 0$ to $f-1$

$\quad (C^A, Z_{w-1:0}) = Z_{w-1:0} + X_{(i+1)v-1:iv} \times Y_{w-1:0}$

$\quad reduce = (M'_{v-1:0} \times Z_{w-1:0})_{v-1:0}$

$\quad (C^B, Z_{w-1:0}) = Z_{w-1:0} + reduce \times M_{w-1:0}$

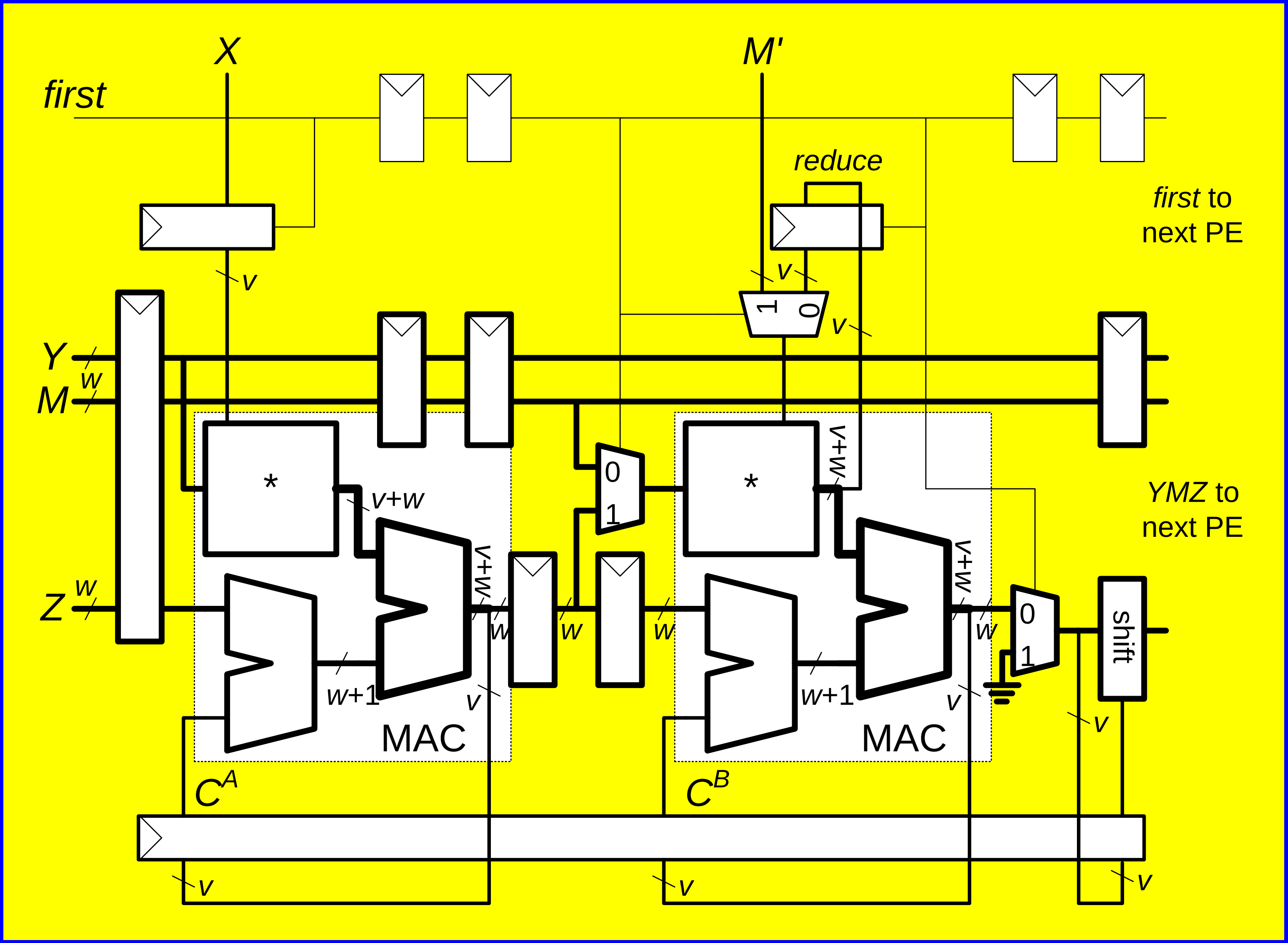
$\quad$ for $j = 1$ to $e+1$

$\quad\quad (C^A, Z_{(j+1)w-1:jw}) = Z_{(j+1)w-1:jw} + X_{(i+1)v-1:iv} \times Y_{(j+1)w-1:jw} + C^A$

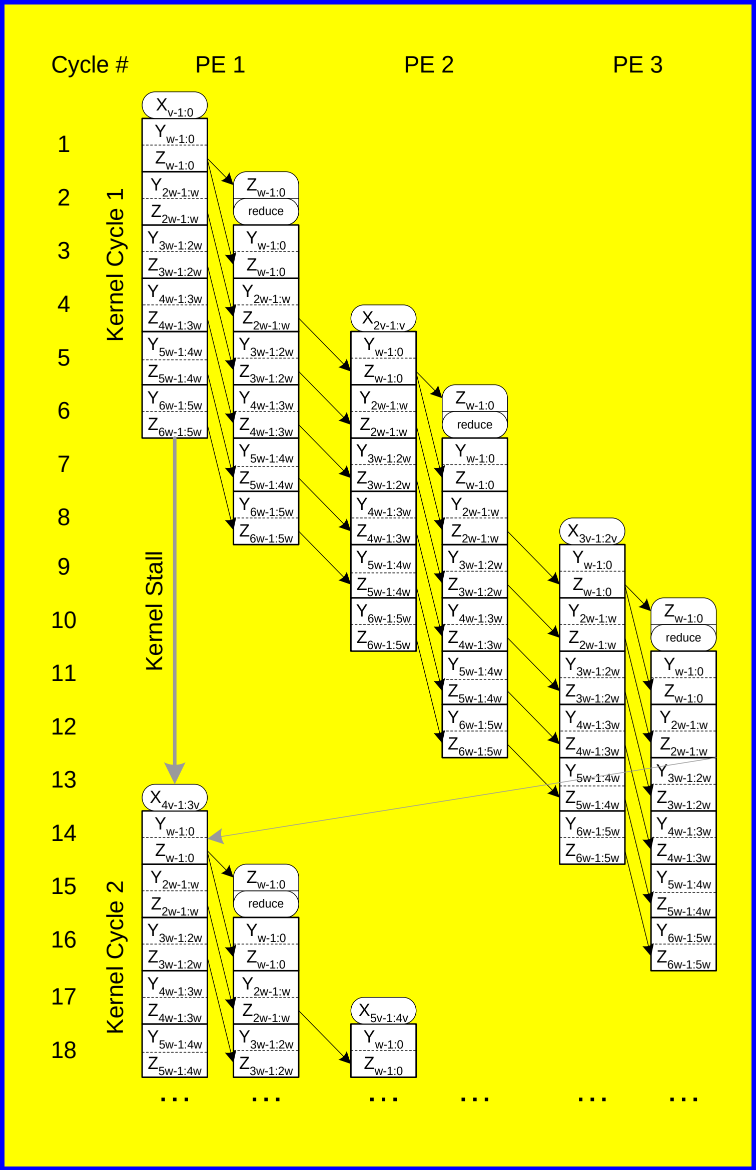
$\quad\quad (C^B, Z_{(j+1)w-1:jw}) = Z_{(j+1)w-1:jw} + reduce \times M_{(j+1)w-1:jw} + C^B$

$\quad\quad Z_{jw-1:(j-1)w} = (Z_{jw+v-1:jw}, Z_{jw-1:(j-1)w+v})$

Very High Radix PE



Very High Radix Pipeline Timing



Latency

- **Tenca-Koç:** ***k* = *n*/*p***

$$\begin{array}{lll} k(e+1)+2(p-1) & e>2p-1 & \text{(Case I)} \\ k\left(2p+1\right)+e-2 & e\le 2p-1 & \text{(Case II)} \end{array}$$

- **Very High Radix:** ***k* = *n*/*pv***

$$\begin{array}{lll} k\left(e+3\right)+4\left(p-1\right)+2 & e>4p-2 & \text{(Case I)} \\ k\left(4p+1\right)+e-1 & e\le 4p-2 & \text{(Case II)} \end{array}$$

Implementation

- C and Verilog reference models
  - Parameterized by *w*, *p*, and *v*
  - Extensive testing up to *n* = 1024
- Synthesized Verilog onto FPGA
  - Xilinx Virtex II Pro XC2V2000-6

Results

Description	Technology	Hardware	Clock Speed (MHz)	Scalable	256-bit time (ms)	1024-bit time (ms)
T-K $p = 40$ $w=8$	0.5 μm CMOS synthesized	28 Kgates	80	Yes	3.8	88
Improved $p = 16$ $w = 16$	Xilinx Virtex II	1514 LUTs + ~5n RAM	144	Yes	1.1	59
Improved $p = 64$ $w = 16$	Xilinx Virtex II	5598 LUTs + ~5n RAM	144	Yes	1.0	16
$p = 4$ $w = 16$ $v = 16$ very high radix	Xilinx Virtex II	780 LUTs + 8 mults + ~5n RAM	102	Yes	0.45	22
$p = 16$ $w = 16$ $v = 16$ very high radix	Xilinx Virtex II	2847 LUTs + 32 mults + ~5n RAM	102	Yes	0.40	6.6

Summary

- Modular exponentiation is key operation in cryptography
- Hardware accelerators getting popular
  - Reconfigurable in key length & field
- Developed improved MM
  - Half the latency for $n \leq w*p$
  - Half the queue size
- Higher radix looks even better
  - Well-suited to FPGAs