

3-Mar-04 (1)

Modular Exponentiation using Parallel Multipliers

S.H. Tang, K.S. Tsui and P.H.W. Leong
Custom Computing Laboratory
The Chinese University of Hong Kong
<http://www.cse.cuhk.edu.hk/~phwl>

3-Mar-04 (2)

Overview

- Introduction
- Algorithms
- Architecture
- Results
- Conclusion

3-Mar-04 (3)

FPGAs for Cryptography

- Particularly suitable for cryptographic accelerators
- State of the art VLSI performance
 - Faster products with zero design effort
- Multiple algorithms on the same hardware
 - Change functionality by downloading different designs to the same device
- Faster design times
 - more sophisticated algorithms, lower design cost, shorter time to market
- No NRE costs
 - Cheaper for low volumes
 - Conversion to ASIC for high performance, high volume
- Only recently have become cost effective for cryptography

3-Mar-04 (4)

This work

- High radix implementation of modular exponentiation
 - Utilizes dedicated multipliers of Virtex II devices
 - Systolic design
 - High radix

3-Mar-04 (5)

Overview

- Introduction
- Algorithms
- Architecture
- Results
- Conclusion

3-Mar-04 (6)

RSA Algorithm

- Invented in 1977 by Rivest, Shamir and Adleman
- Credit card machines, bank ATMs, e-mail applications, web browsers and mobile phones
- Encryption, key exchange, authentication

$$\begin{array}{l}
 P \ \& \ Q \ \text{PRIME} \\
 N = PQ \\
 ED \equiv 1 \pmod{(P-1)(Q-1)} \\
 C = M^E \pmod{N} \\
 M = C^D \pmod{N} \\
 \text{RSA Algorithm}
 \end{array}$$

3-Mar-04 (7)

RSA Example

- $P=11, Q=7$ (secret)
- $N=pq=77$ & $E=13$ (public)
- Anyone can encrypt a message e.g. $M=4$
 - $4^{13} \bmod 77=53$
- Decryption
 - $(P-1)(Q-1)=60, 13 \times 37 \bmod 60=1$ ($D=37$)
 - $53^{37} \bmod 77 =$
628358038363668332248635694548393830494073
 $1973668146791149026213 \bmod 77 = 4$

3-Mar-04 (8)

Modular Exponentiation Algorithm ($P=C^E$)

- L-R binary exponentiation method
- On average takes 1.5h multiplications
- Multiplications are modulo M

```

P = 1;
for i = h-1 .. 0 do
{
  P = P x P;
  if  $e_i = 1$ 
    P = P x C;
}

```

3-Mar-04 (9)

The Montgomery Algorithm

- Used in most software and hardware implementations of modular exponentiation
- Performed in an R-residue where R is chosen as a power of 2
 - Divisions are shift operations
 - Reduction is interleaved with the multiplication

3-Mar-04 (10)

Montgomery Multiplication

- Input: A, B and M'
- Output: $S = ABR^{-1} \pmod{M}$
- Precompute M' s.t. $(-M m') \pmod{R} = 1$
- Note that R^{-1} term is unwanted

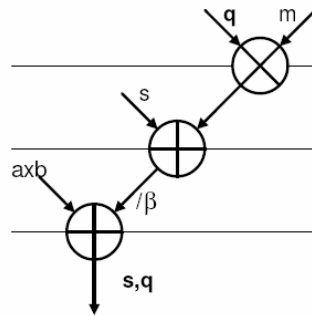
```

for i = 0 ... n do
{
  q = ((s mod β) × (M' mod β)) mod β
  s = (s + qM) / β + aiB
}

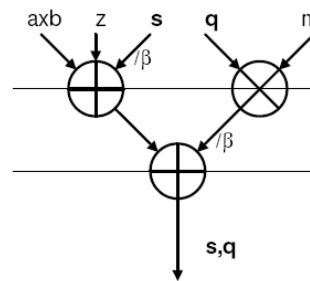
```

3-Mar-04 (11)

Orup's optimization simplifies calc of q and reduces logic levels



$$S = (S + qM)/\beta + a_i B$$



$$S = S/\beta + qM/\beta + z + a_i B$$

3-Mar-04 (12)

Optimized Montgomery Multiplication (Orup)

- Input: A , B and M'
- Output: $S = ABR^{-1} \pmod{M}$

```

 $\mathcal{M} = M (M' \bmod \beta)$ 
for i = 0 ... n do
{
    q = S mod  $\beta$ 
    z = 1 if q  $\neq$  0 else 0
    S =  $S/\beta + (q\mathcal{M})/\beta + z + a_i B$ 
}

```

3-Mar-04 (13)

Exponentiation using Montgomery Multiplication

- Extra pre and post processing steps are needed to remove unwanted R^{-1} term.

$C^2 <= (C)(C)$ $C^4 <= (C^2)(C^2)$ $C^5 <= (C^4)(C)$ $C^{10} <= (C^5)(C^5)$	$CR <= (C)(R^2)R^{-1}$ $C^2R <= (CR)(CR)R^{-1}$ $C^4R <= (C^2R)(C^2R)R^{-1}$ $C^5R <= (C^4R)(CR)R^{-1}$ $C^{10}R <= (C^5R)(C^5R)R^{-1}$ $C^{10} <= (C^{10}R)(1)R^{-1}$
--	--

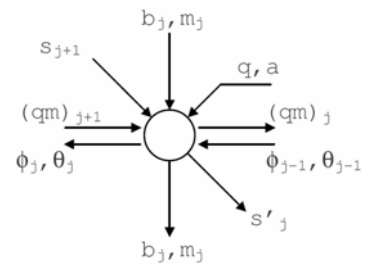
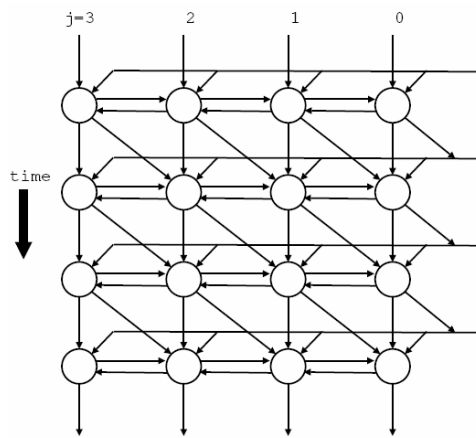
3-Mar-04 (14)

Overview

- Introduction
- Algorithms
- Architecture
- Results
- Conclusion

3-Mar-04 (15)

Systolic Design



- Multiplier is too large for qM and a_iB
- Use a systolic array structure
 - Project vertically (row)

3-Mar-04 (16)

Processing element

$$\theta_j = q \times m_j / \beta$$

$$\phi_j = a \times b_j / \beta$$

$$(qm)_j = \langle q \times m_j \rangle_\beta + \theta_{j-1}$$

$$(ab)_j = \langle a \times b_j \rangle_\beta + \phi_{j-1}$$

$$s_j = s_{j+1} + (qm)_{j+1} + (ab)_j$$

3-Mar-04 (17)

Carry Save Adder

- The carry propagates through all PEs
 - 1024b addition takes 70 ns
 - Fast carry broken into 4 segments
- Use carry save architecture
 - The carry is saved in individual PE and used in later cycles
 - Addition is 17 bits

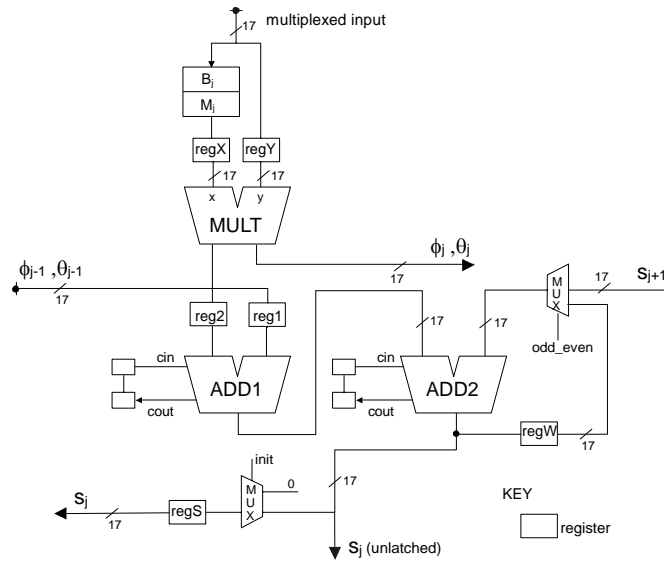
3-Mar-04 (18)

PE Design

- Single multiplier per cell (time multiplexed)
- Two clock cycles for each PE to compute S (pipelined)
 - ab on even cycles
 - qm on odd cycles
- Multiplex inputs to reduce wiring

3-Mar-04 (19)

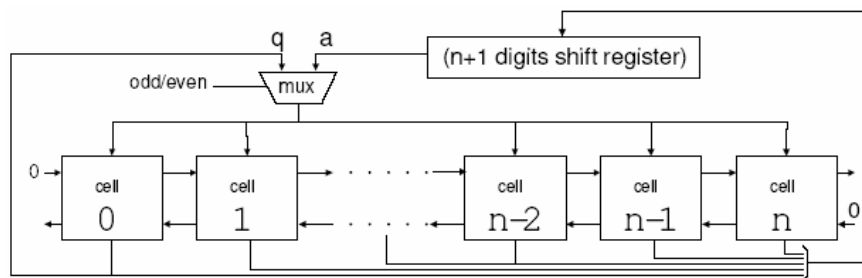
PE Block Diagram



3-Mar-04 (20)

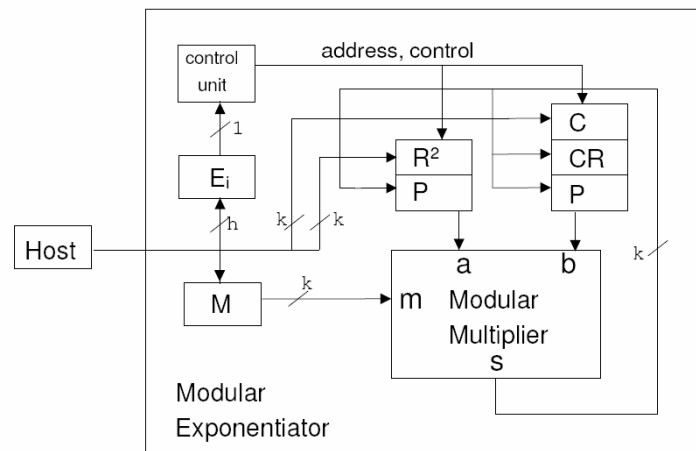
Semi-Systolic Array

- Has broadcast signals but reduces latency



3-Mar-04 (21)

Exponentiation



3-Mar-04 (22)

Overview

- Introduction
- Algorithms
- Architecture
- Results
- Conclusion

3-Mar-04 (23)

Results

- Platform: XC2V3000-6, XC2V4000-6
- Design description: VHDL
- Can work w/ or w/o CRT for RSA
- Designs of 512-bit and 1024-bit key are implemented
- All designs use a radix of 2^{17}

3-Mar-04 (24)

Clock Cycles

- Multiplication
 - $2n$ cycles (n = number of digits)
- Exponentiation
 - $3h/2$ cycles (h = number of exponent bits)
- Total
 - Approximately $3nh$ cycles (i.e. $O(h^2)$)

3-Mar-04 (25)

Clock Cycles

Operation	Generalized ¹	512b	1024b
Modular Multiplication			
MP × MP	$2(n + 5)$	74	134
Modular Exponentiation			
Montgomery pre/post-processing	$2(n + 5) \times 3$	222	402
Exponentiation	$2(n + 5) \times (b - h + \frac{3h}{2})$	59200	209844
Input / Output			
Input of R^2, M, A, E	$4 \times (b/64) + 1$	37	69
Output of $A^E \bmod M$	$(b/64) + 1$	10	18
Measured total clock cycles		59468	210333

¹ b – number of gross bits, 544 and 1054 for $r=17$ and $h=512$, 1024 respectively

3-Mar-04 (26)

Results

- Operating frequency: 90MHz
- Resources
 - 14334 of 14336 slices
 - 62 of 96 multipliers
- Using CRT, 1024-bit RSA can be computed in 0.66ms
 - i.e. 1.5 Mbps throughput
 - 10 times faster than an Intel P4 1.7GHz (OpenSSL with all optimizations)
- Software verification using OpenSSL library routines and random data

3-Mar-04 (27)

Performance Comparison

Author	Year	Technology	Speed (1024b decryption)
Orup et al	1991	ASIC 0.6u	(512b) 5.5 ms
Shand et al	1993	16x XC3090	6.1 ms
Itoh et al	1999	TMS320C6201 DSP	11.7 ms
Blum et al	2001	XC40250XV-09	3.1 ms
OpenSSL	2003	P4 1.7 GHz	6.9 ms
THIS WORK	2003	XC2V4000-6	0.7 ms

3-Mar-04 (28)

Conclusion

- Systolic array for modular exponentiation
- High clock frequency
 - Aggressive pipelining (systolic design)
 - Carry save adder
- Low number of clock cycles (approx 3nh)
 - Orup's improved Montgomery Multiplication algorithm
 - 17x17 multipliers used to achieve high radix ($\beta=2^{17}$)
 - Reduced latency
- Parallelism
 - Multipliers and adders are used every clock cycle
- Performance of 1.5 Mb/s achieved on an XC2V4000-6 device