FPGA Implementation of High Throughput Circuit for Trial Division by Small Primes

Gabriel Southern

Chris Mason

Lalitha Chikkam

Patrick Baier

Kris Gaj

George Mason University

Co-authors

Chris Mason

Lalitha Chikkam

Patrick Baier

Kris Gaj

Objectives

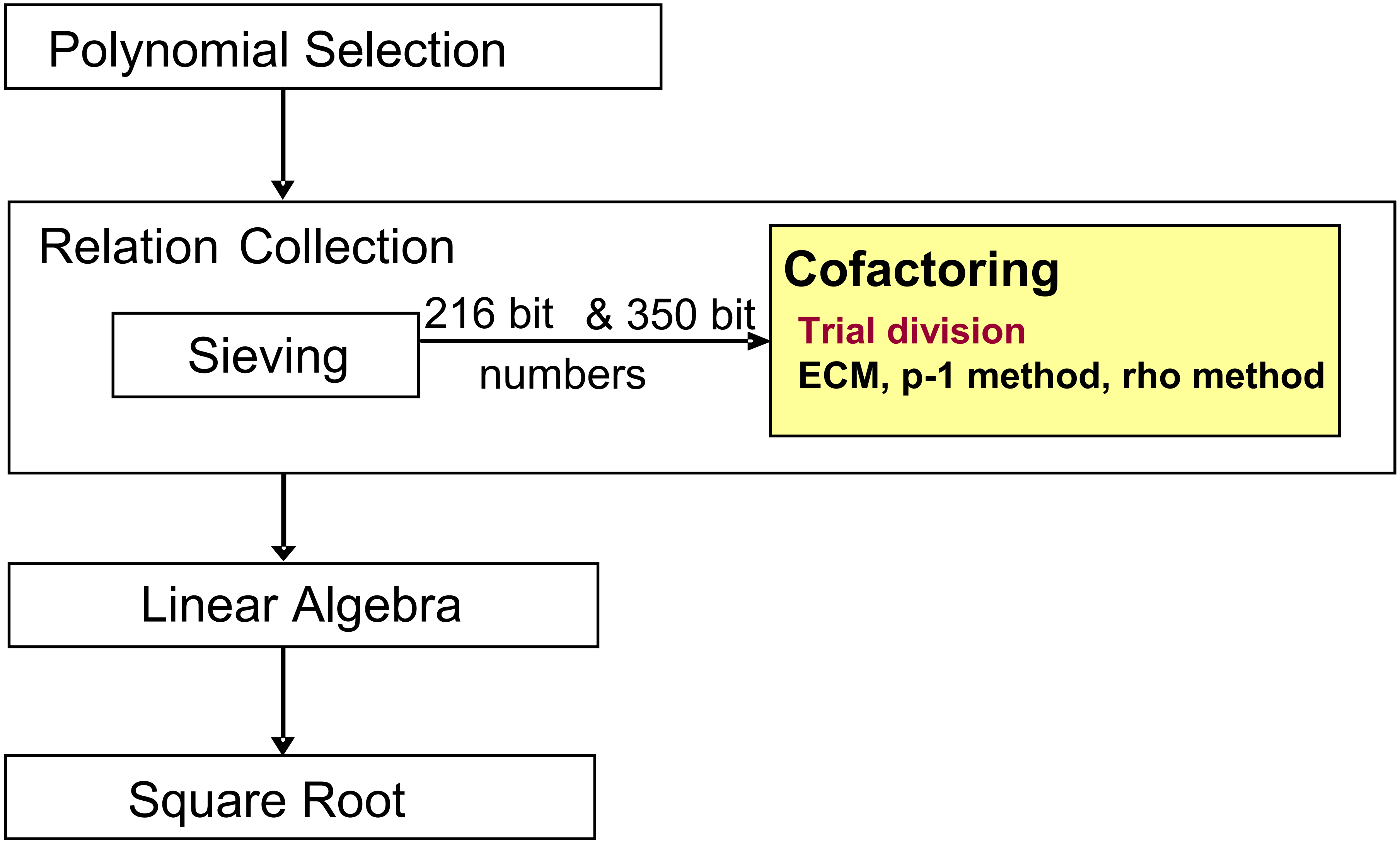
- Optimize Trial Division algorithm for use in the hardware implementations of the Number Field Sieve and/or Quadratic Sieve
- Develop efficient FPGA architecture optimized for maximum throughput
- Compare performance of FPGA implementation with software implementation

Outline

- Trial Division in Number Field Sieve (NFS)
- Design Decision
- Hardware Architecture
- Synthesis results
- Comparison with software implementation
- Conclusion

Trial Division in Number Field Sieve (NFS)

Factoring 1024-bit RSA keys using Number Field Sieve (NFS)



Trial Division

Given:

Inputs:

Variables:

1. Integers N_1, N_2, N_3, ... each of the size of k-bits

Constants:

2. Factor base =

   set of all primes smaller smaller than a certain bound B

   $= \{ p_1=2, p_2=3, p_3=5, \ldots , p_t \leq B \}$

Parameters of interest:

$k = 216, 350, 512$

$B = 100\,000$

Trial Division (cont)

<u>Outputs:</u>

For each integer N_i:
A list of primes from the factor base that divides N_i,
the number of times each prime divides N_i, and
the remainder M_i after factoring out small primes

For example if

$$N_i = p_1^{e1} \cdot p_2^{e2} \cdot p_3^{e3} \cdot M_i,$$

where M_i is not divisible by any prime belonging to
a factor base, then
the output is

$$\{p_1, e1\}, \{p_2, e2\}, \{p_3, e3\}, \{M_i\}$$

Classical Division Algorithms

- Division is typically performed as a series of subtract and compare operations
- Sequential divider
  - uses shift/subtract division algorithm
  - requires one clock cycle per quotient bit
  - small but slow
- Array divider
  - pipelined version of sequential divider
  - circuit area quadratically dependent on dividend size
  - fast but large

Design Decisions

Trial Division by Small Primes

- Large dividend (216, 350, or 512 bits)
- Many small divisors
  - Divisor 17 bits
  - 9592 divisors per dividend
- Relatively few small primes divide a randomly chosen large integer N

Expected Number of Small Prime Factors for 512-bit Number

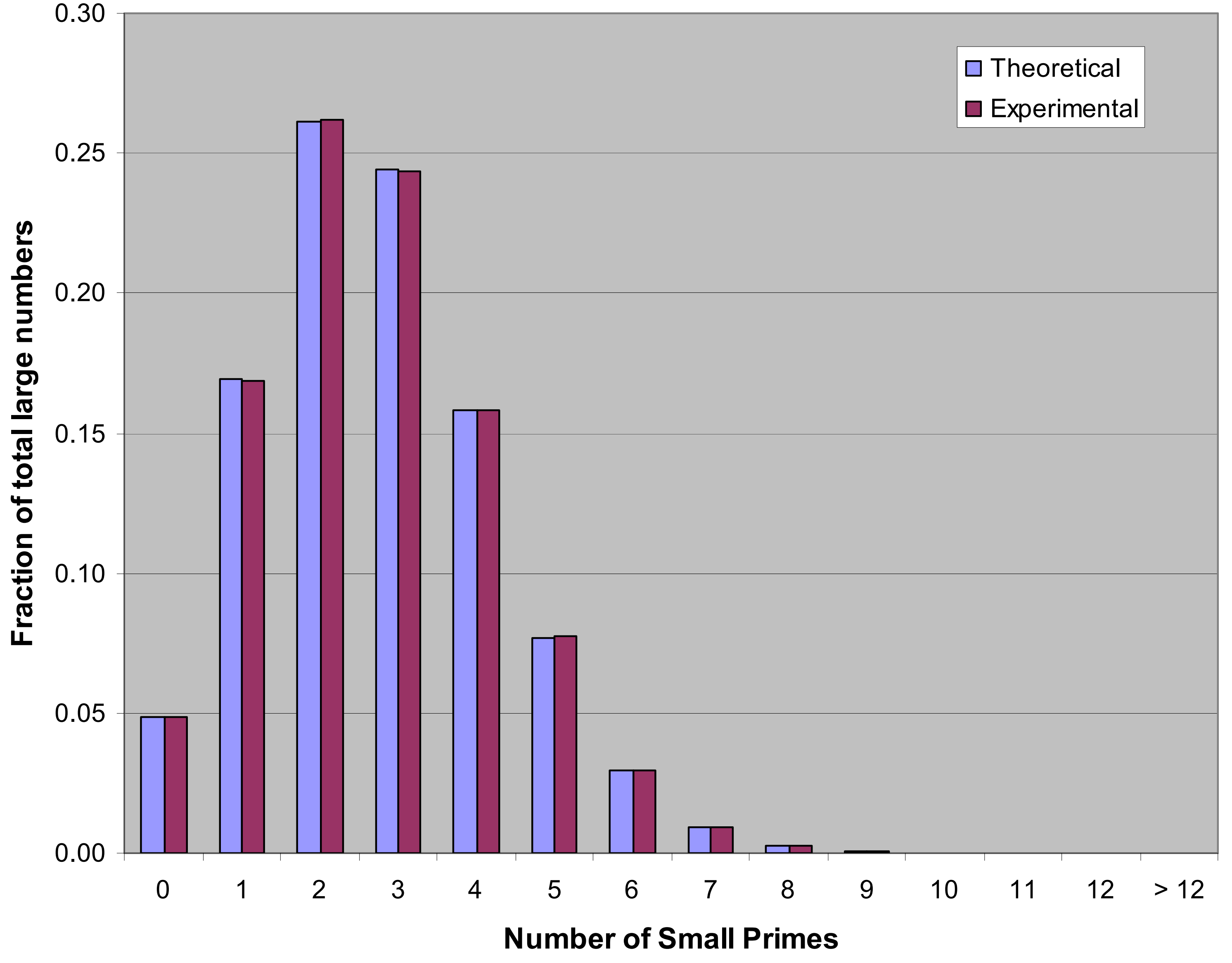
- Developed software program to determine expected number of small prime factors
- Demonstrated that about 99.9% of random numbers have 9 or fewer small prime factors
- Independently verified results theoretically (details in paper)

c – number of prime factors < B

q_c – fraction of numbers with c prime factors

c	Calculated q_c	Experimental q_c
0	0.048753	0.048750
1	0.169584	0.168777
2	0.261423	0.261627
3	0.244033	0.243688
4	0.157985	0.157970
5	0.076659	0.077459
6	0.029327	0.029584
7	0.009167	0.009327
8	0.002404	0.002396
9	0.000540	0.000510
10	0.000105	0.000104
11	0.000018	0.000016
12	0.000003	0.000000
> 12	0.000000	0.000000

Expected Number of Small Prime Factors (< 100,000) for 512-bit Number



Design Decisions

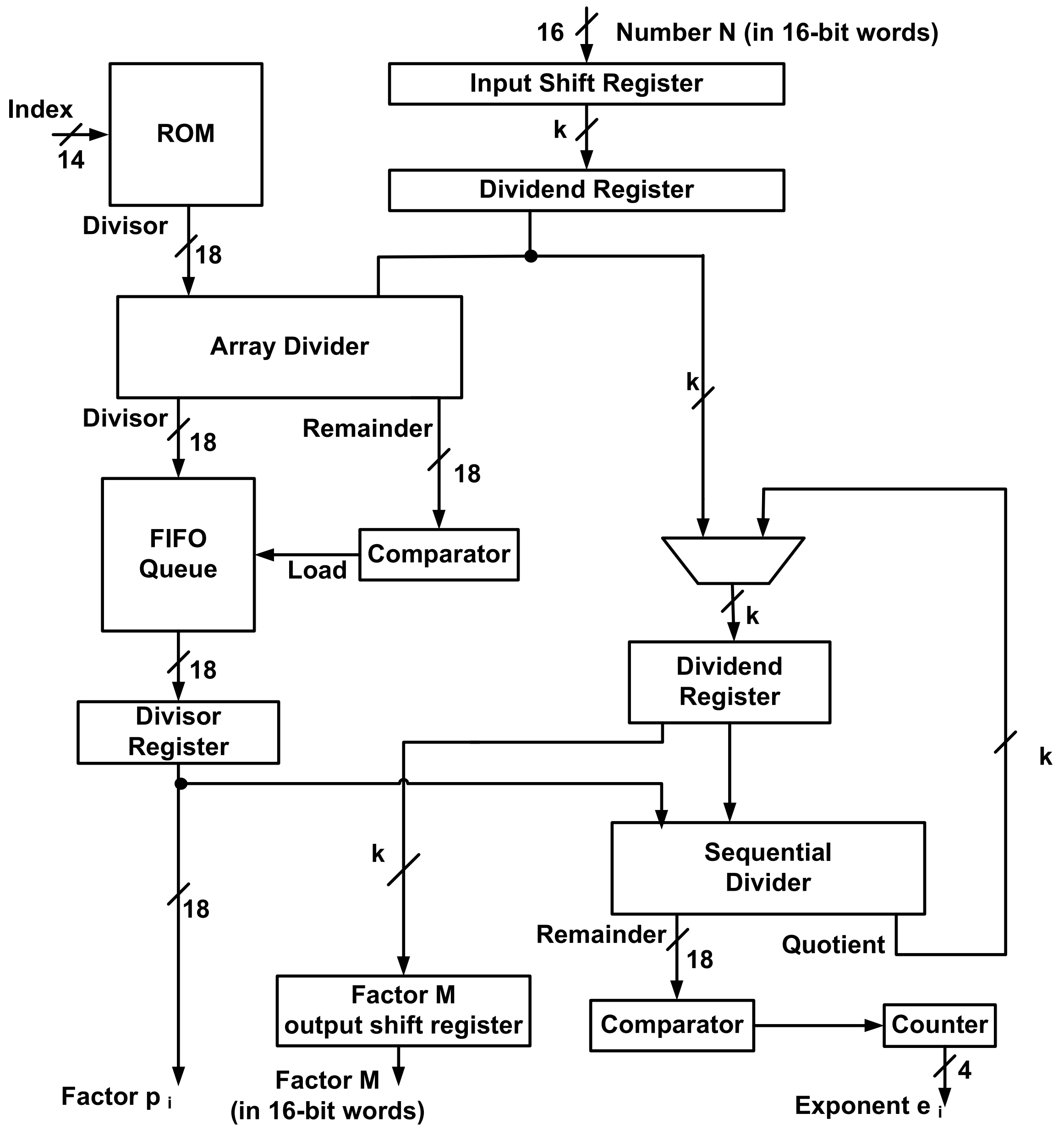
- Partition circuit into two main components
  - Fast pipelined array divider to determine divisibility
  - Small sequential divider to process numbers found to be divisible
- Array divider determines divisibility of each large integer by 9592 small primes
  - One result obtained every clock cycle
  - Circuit area scales linearly with dividend size
- Sequential divider processes numbers when array divider finds a factor
  - Small circuit area
  - Execution time proportional to dividend size
  - Processes relatively few numbers in parallel with further operation of array divider

Hardware Architecture

Circuit Algorithm

```
**Data:** Dividend input: $N = p_1^{e_1} \cdot p_2^{e_2} \cdots p_{c-1}^{e_{c-1}} \cdot p_c^{e_c} \cdot M$
**Result:** Prime factors $p_i$, corresponding exponents $e_i$, and factor $M$
foreach prime p in prime_set do
    if p divides N then
        validPrimeList.Add(p);
    end
end
while validPrimeList.HasElements do
    prime p = validPrimeList.GetNextPrime();
    count = 1;
    while p divides N do
        N = N / p;
        count = count + 1;
    end
    resultList.Add(p, count);
end
```

Circuit Block Diagram



Division Example

230 / 6 = 38 remainder 2

```
   38
  ________
6 | 230
   18
  ----
    50
    48
   ----
     2
```

```
        100110
     __________
110 | 11100110
      110
     -----
       0100110
       110
      -----
        100110
        110
       -----
        100110
         110
        -----
          1110
          110
         -----
           010
           110
          -----
           010
```

Array Divider Algorithm

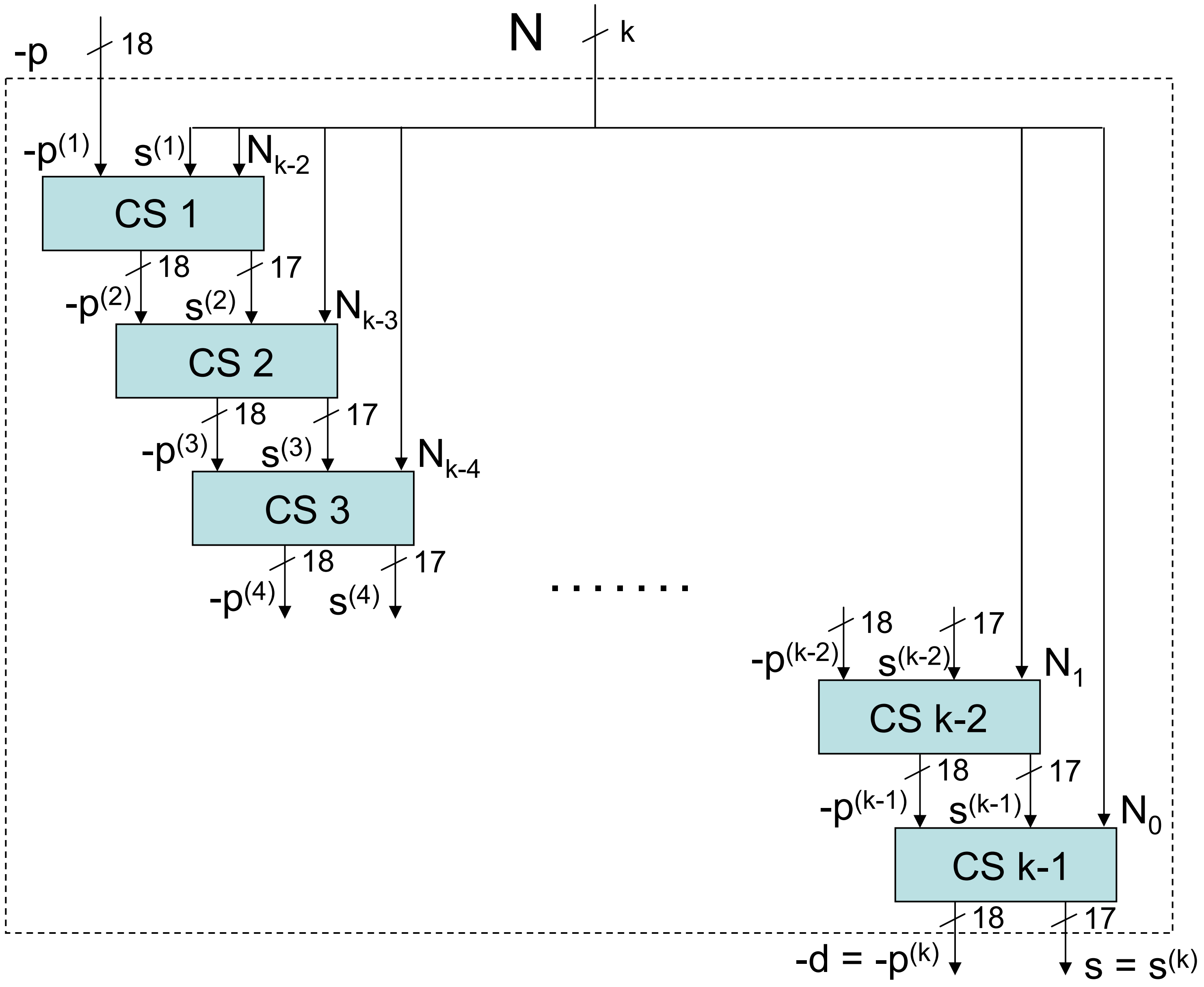
Given: N (k-bit integer), -p (where p is a small prime, $p < 2^{17}$)

Compute: $s = N \bmod p$

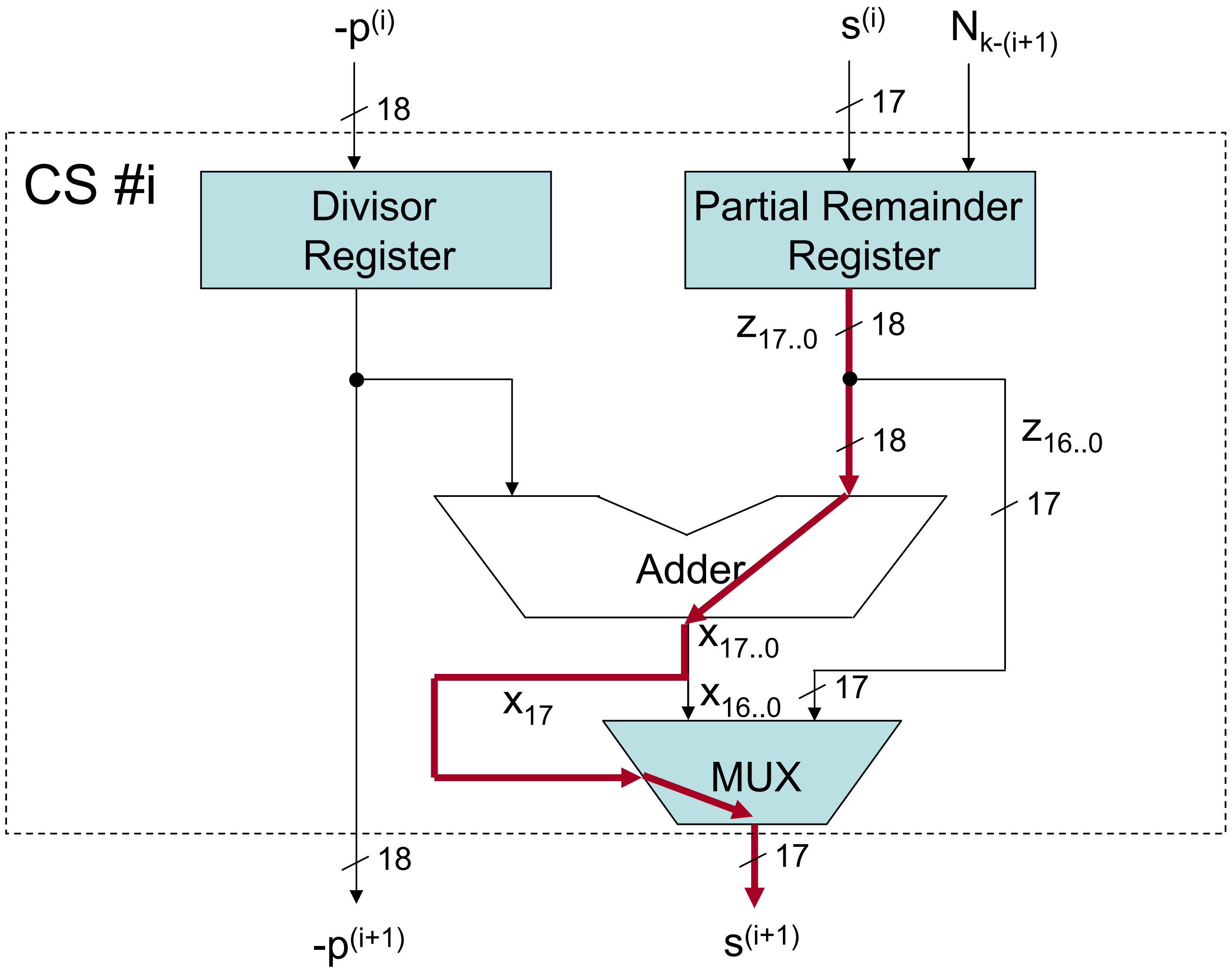
```
$s^{(1)} = 000\ldots0N_{k-1}$     ; 17 zeros followed by $N_{k-1}$
for  i = 1 to k-1 do
   $s^{(i)} = s^{(i)} \,||\, N_{k-(i+1)}$   ; equivalent to  $s^{(i)} = 2*s^{(i)} \,|\, N_{k-(i+1)}$
   if $(s^{(i)} + (-p^{(i)}) > 0$ then
     $s^{(i+1)} = s^{(i)} + (-p^{(i)})$
   else
     $s^{(i+1)} = s^{(i)}$
end do
$s = s^{(k)}$
```

|| denotes concatenation, | denotes bitwise or

Array Divider Block Diagram



Controlled Subtractor Cell

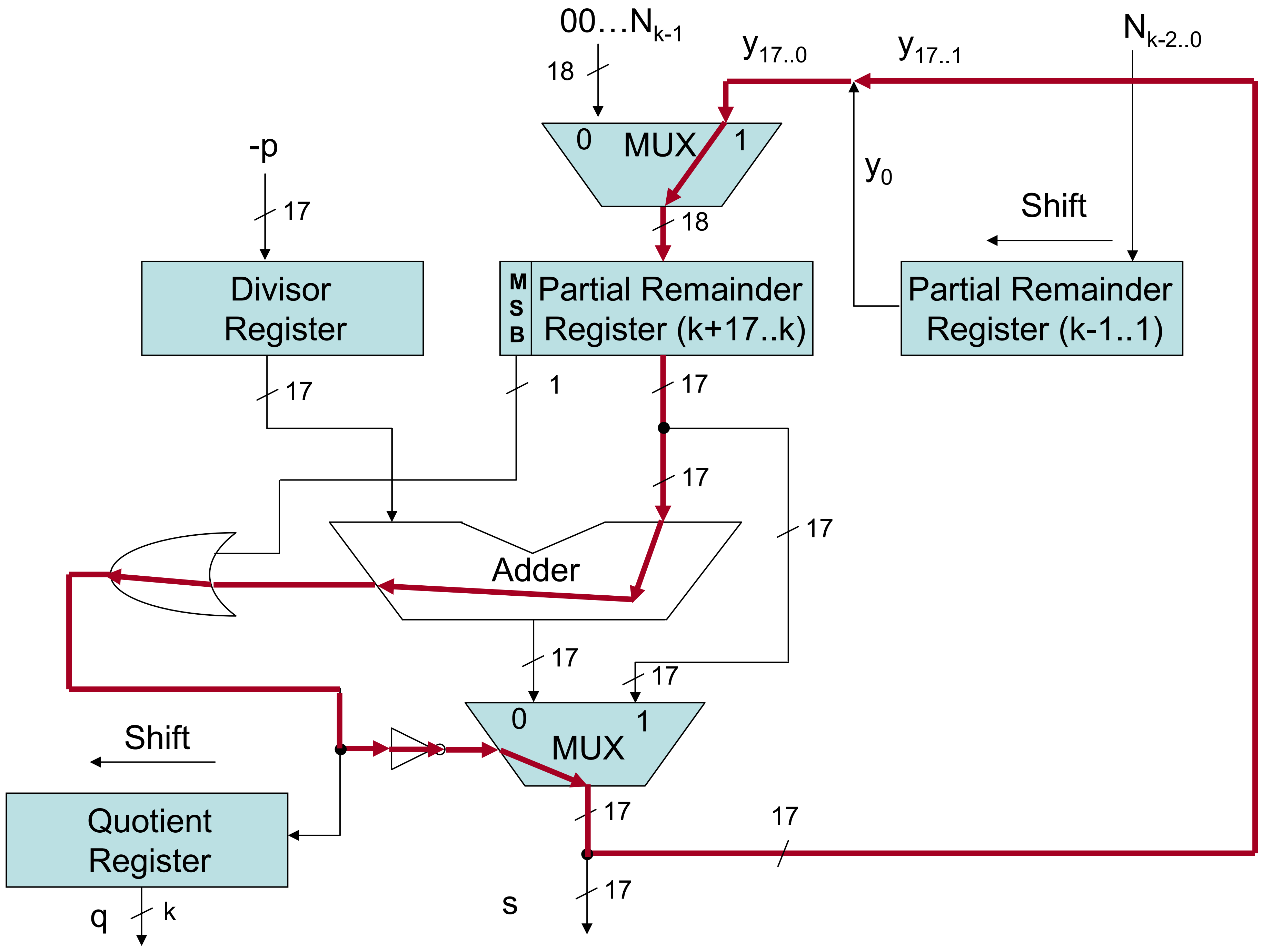


Sequential Divider Algorithm

Given: N (k-bit integer), -p (where p is a small prime, $p<2^{17}$)
Compute: $q = N / p$; $s = N \bmod p$

```
$s^{(0)} = 000…0 \,||\, N$    ; 18 zeros followed by N
for  $i = 1$ to $k$ do
if $(2 \cdot s^{(i-1)} + 2^k \cdot (-p) > 0)$ then
     $q_{k-i} = 1$
     $s^{(i)} = 2 \cdot s^{(i-1)} + 2^k \cdot (-p)$
   else
     $q_{k-i} = 0$
     $s^{(i)} = 2 \cdot s^{(i-1)}$
end do
$q = q_{k-1..0}$
$s = s^{(k)}_{k+16..k}$
```

|| denotes concatenation


00…N$_{k-1}$
18
y$_{17..0}$
y$_{17..1}$
N$_{k-2..0}$
0
MUX
1
-p
17
y$_0$
Shift
18
Divisor Register
M S B
Partial Remainder Register (k+17..k)
Partial Remainder Register (k-1..1)
17
1
17
17
17
Adder
17
17
0
1
MUX
Shift
Quotient Register
17
17
q
k
s
17

Synthesis Results

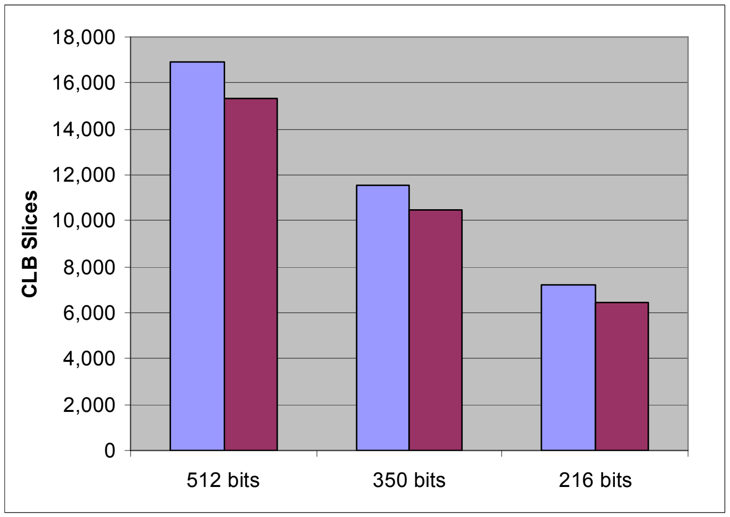
Circuit Area CLB Slices

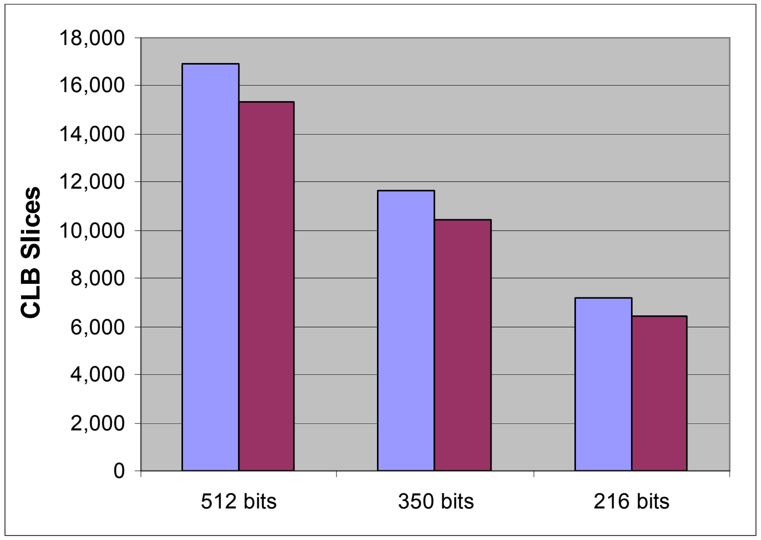
Spartan-3

Dividend	Total	Array Divider
512 bits	16,922	15,323
350 bits	11,614	10,462
216 bits	7,216	6,441

Virtex-4

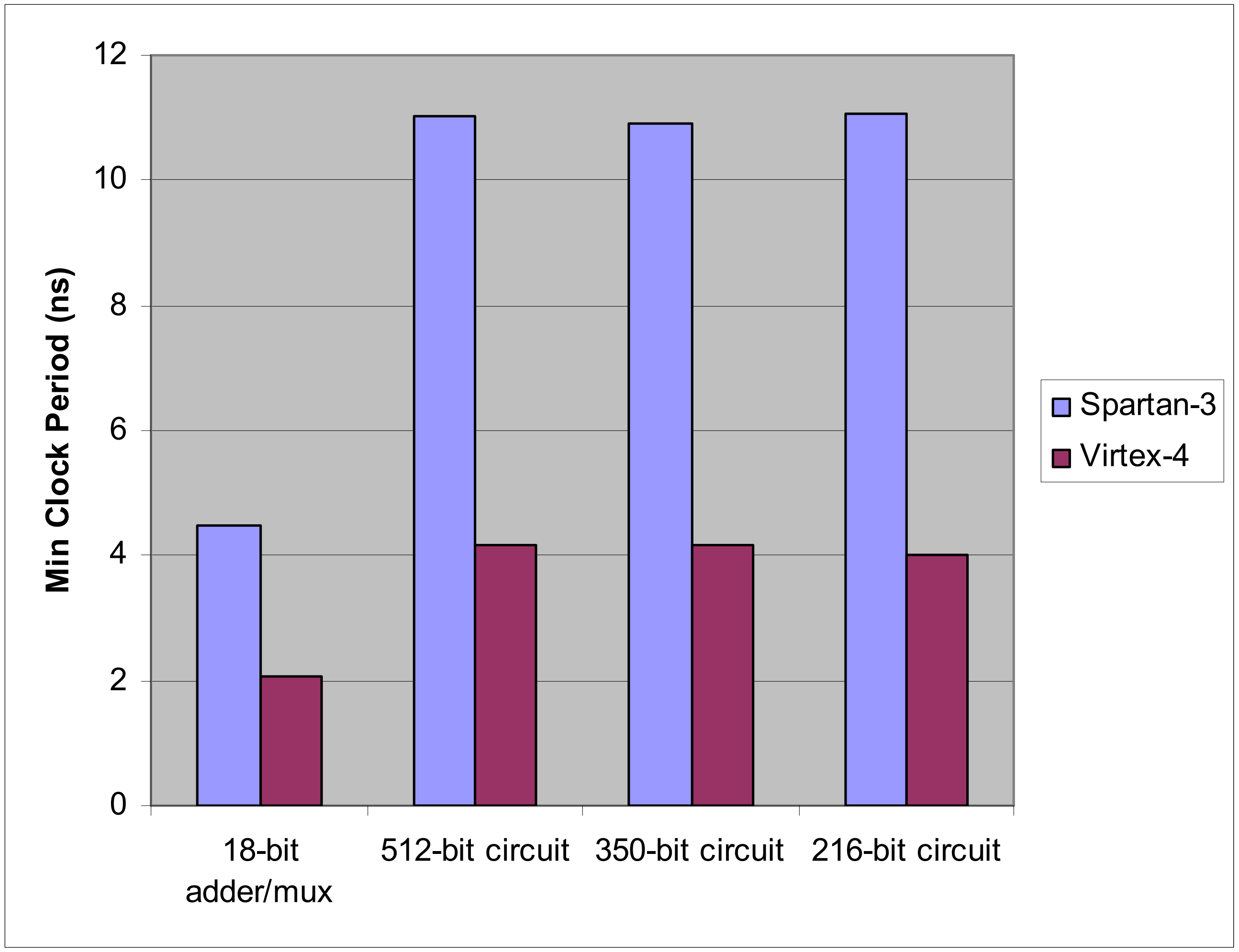
Dividend	Total	Array Divider
512 bits	16,895	15,323
350 bits	11,578	10,462
216 bits	7,182	6,441





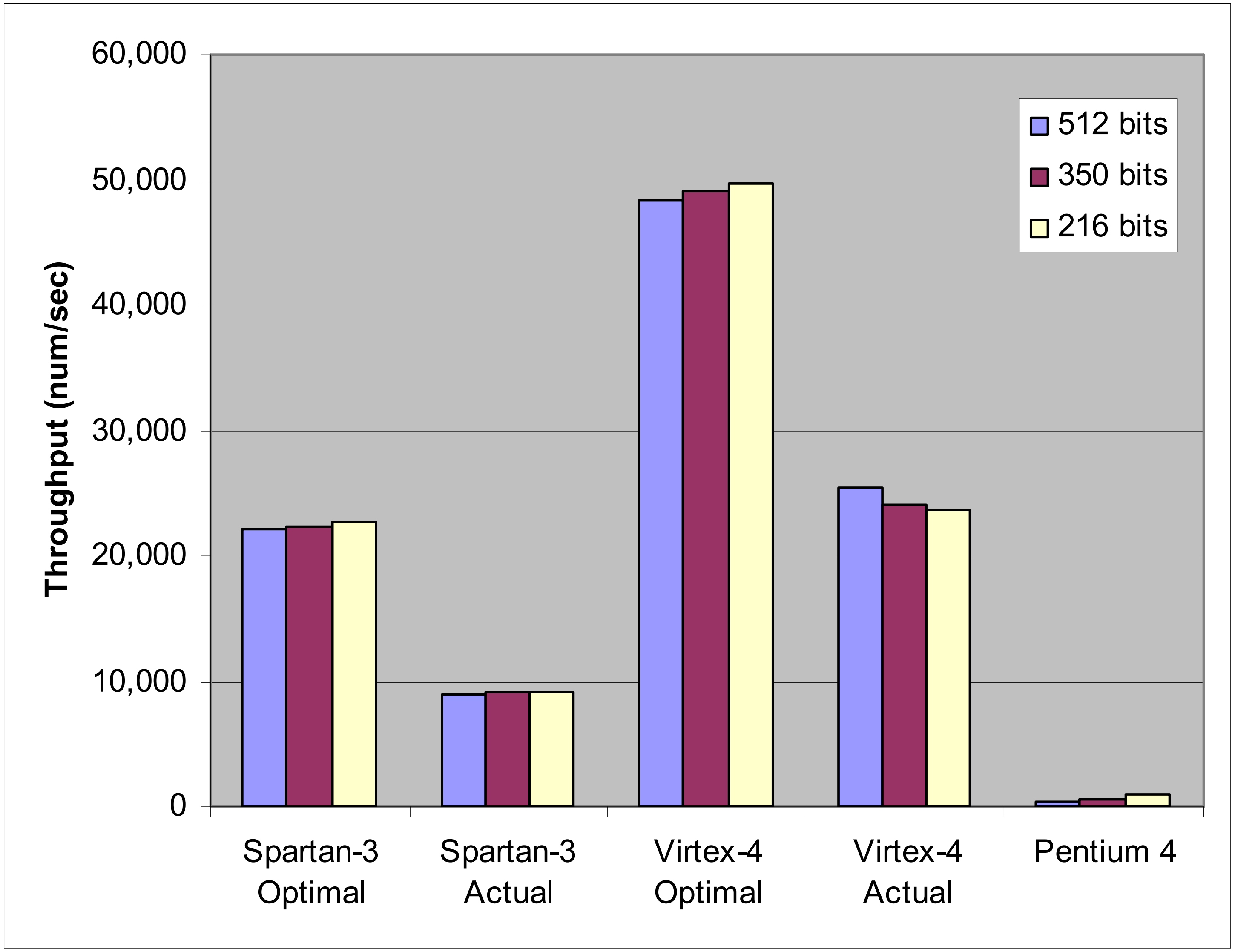
🟦 Total
🟥 Array Divider

Circuit Minimum Clock Period

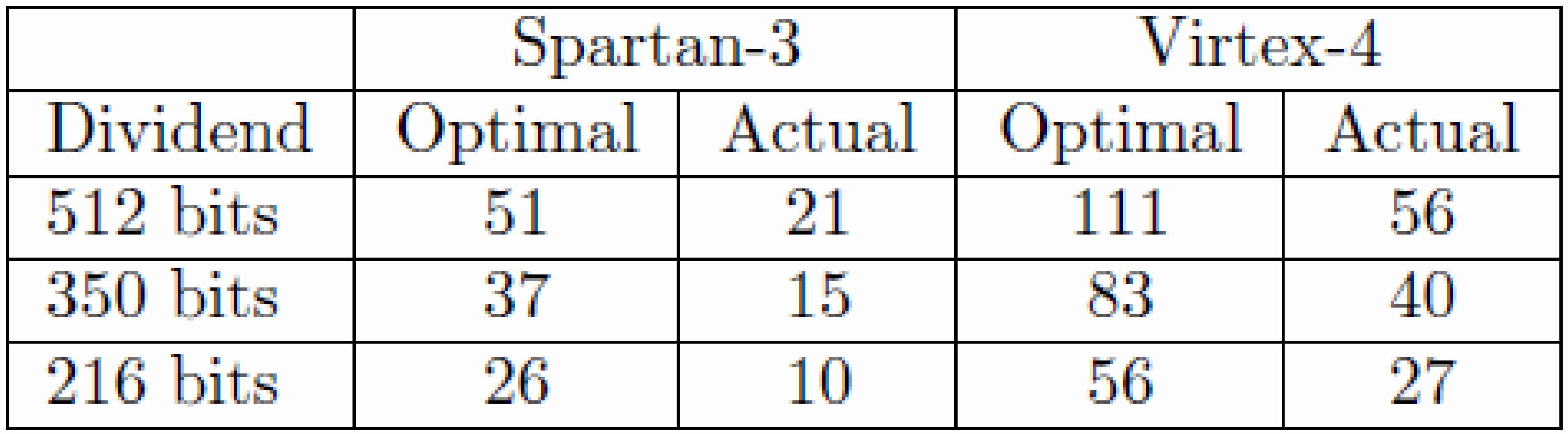


Comparison with Software

Circuit Throughput



Hardware Speedup vs. Software



	Spartan-3		Virtex-4	
Dividend	Optimal	Actual	Optimal	Actual
512 bits	51	21	111	56
350 bits	37	15	83	40
216 bits	26	10	56	27

Approximate Cost of FPGA and CPU in Quantities of 100

• Spartan 3 XCS1500, XCS2000:	$50
• Virtex 4 XC4VLX25:	$200
• Virtex 4 XC4VLX40:	$450
• Intel Xeon 2.8 GHz:	$200

Sources: www.em.avnet.com
www.nuhorizons.com
www.compuvest.com

Performance / Cost Gain Hardware vs. Software

Dividend	Spartan-3 Optimal	Spartan-3 Actual	Virtex-4 Optimal	Virtex-4 Actual
512 bits	203	**83**	49	26
350 bits	151	62	37	**18**
216 bits	103	42	56	27

Conclusions

- Developed novel hardware architecture for trial division with applications in NFS and QS

- Our architecture relies on parallel execution of two units
  - Array divider: optimized for maximum throughput, checking divisibility by small primes
  - Sequential divider: optimized for minimum area, performing the actual divisions

- Spartan 3 and Virtex 4 FPGAs implementations outperform Intel Xeon 2.8 GHz / GMP library implementations by a factor
  - 10 to 50 in terms of performance
  - 20 to 80 in terms of performance to cost ratio

Thank you!

Questions??
?