A 100 Kbit/s Single Chip Modular Exponentiation Processor

Holger Orup
Computer Science Department
Aarhus University
DK-8000 Aarhus C, Denmark

in cooperation with

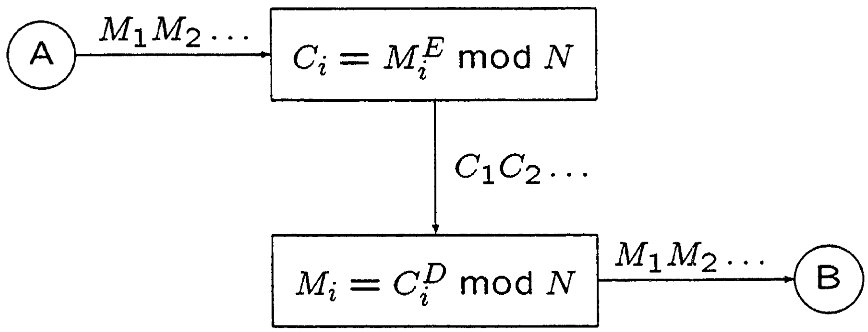
TELE DANMARK
JYDSK TELEFON

Outline of Talk

- Why modular exponentials ?
- Parallel exponentiation algorithm.
- High-radix modular multiplication algorithm.
- Architecture.
- Redundant carry save representation.
- Test and performance.
- Future work.
- Summary.

Why Modular Exponentials ?

- Public-key crypto systems. (RSA, ...).
- Primality test, key generation.



- Requirements for this implementation:
  64 Kbit/s transmission rate (ISDN).
  561 bit operands.

Modular Exponentiation Algorithm

Stimulus: E, M, N, where $E \geq 0$ and $0 \leq M < N$,
$E = e_{n-1} e_{n-2} \cdots e_0$. Binary encoded.

Response: $X = M^E \bmod N$.

Method: $X := 1$; $Y := M$;
for $i := 0$ **to** $n-1$ **do**
  if $e_i = 1$ **then** $X := (X \cdot Y) \bmod N$;
  $Y := (Y \cdot Y) \bmod N$;
end;

$$\mathit{Time}[\![\mathit{Exp}]\!](n) = \begin{cases} 1.5n \cdot \mathit{Time}[\![\mathit{Mult}]\!](n) & \text{average} \\ 2n \cdot \mathit{Time}[\![\mathit{Mult}]\!](n) & \text{worst} \end{cases}$$

Faster Exponentiation

- Previous approaches:
  Reducing the number of modular multiplications.
  Addition chains. High radix encoding of exponent.
  $\mathit{Time}[\![\mathit{Exp}]\!](n) > n \cdot \mathit{Time}[\![\mathit{Mult}]\!](n)$.

- This approach:
  Reducing the time by performing
  two modular multiplications in parallel:

  for $i := 0$ **to** $n-1$ **do in parallel**
      # **if** $e_i = 1$ **then** $X := (X \cdot Y) \bmod N$;
      # $Y := (Y \cdot Y) \bmod N$;
  end;

  $\mathit{Time}[\![\mathit{Exp}]\!](n) = n \cdot \mathit{Time}[\![\mathit{Mult}]\!](n)$.

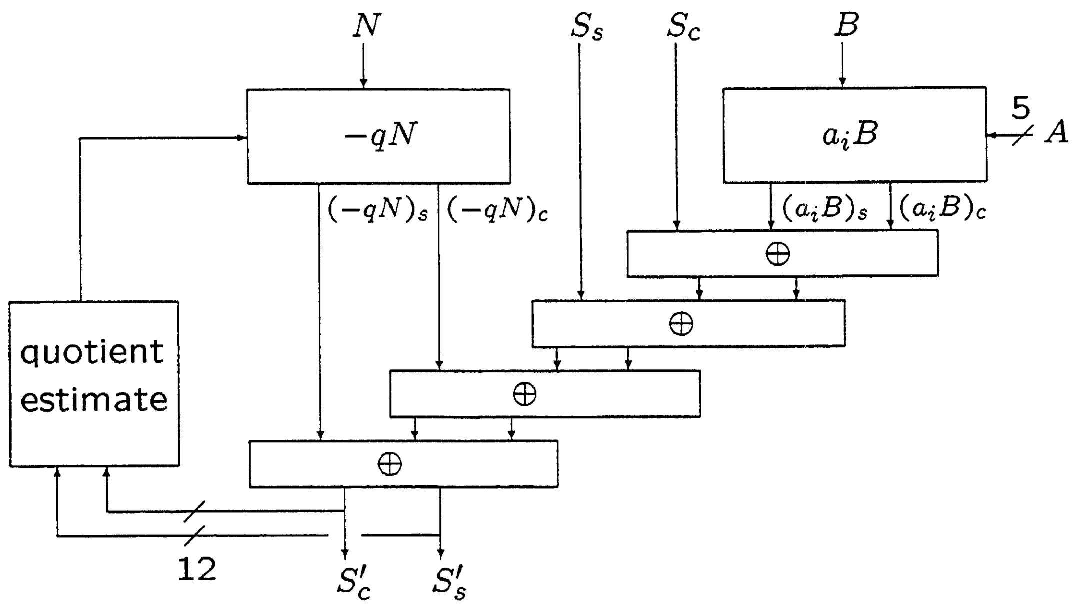
Radix 32 Modular Multiplication

- Increasing the speed by reducing the number of cycles in a serial-parallel multiplication scheme.

Stimulus: A, B, N, where $0 \le A, B < 2N$.
$A = a_{n'-1}a_{n'-2}\cdots a_1 a_0$. Radix 32, $n' = \frac{n}{5}$.
Response: $S \equiv AB \bmod N$, where $0 \le S < 2N$.
Method: $S := 0$;
for $i := n'-1$ **downto** 0 **do**
    $q :=$ Estimate(S div N);
    $S := 2^5 S + a_i B - 2^5 q N$;
end;

- S has the dual role of a partial remainder and partial product.
- $\mathit{Time}[\![\mathit{Mult}]\!](n) = \frac{n}{5} \cdot \mathit{Time}[\![\mathit{Cycle}]\!]$

Architecture for Calculating $S' := 2^5 S + a_i B - 2^5 qN$



... Architecture

- Multiples $-qN$ and aB in redundant carry save representation.
- Accumulator S in redundant carry save representation.
- Quotient estimation by inspecting 12 most significant bits of S and N.
- Carry completion adder converts redundant carry save representation to non-redundant binary representation (not shown).
- Critical Path: Quotient estimate + generation of multiple + two carry save additions.

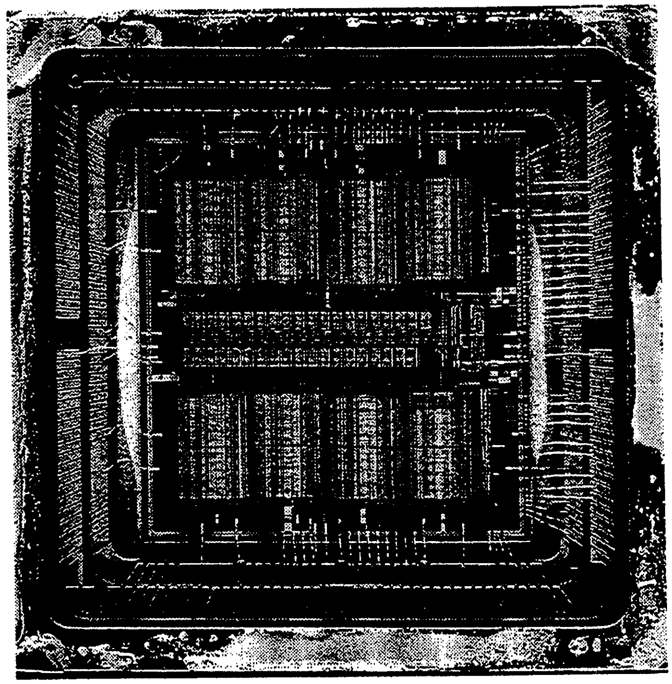
Algorithmic Improvements

- Parallel exponentiation:
  $n \cdot \mathit{Time}[\![\mathit{Mult}]\!](n)$ vs. $2n \cdot \mathit{Time}[\![\mathit{Mult}]\!](n)$
  The architecture is pipelined for simultaneously performing two multiplications, $\mathit{Time}[\![\mathit{Cycle}]\!] = 2 \cdot \mathit{Time}[\![\mathit{Clock}]\!]$.
- Radix 32 multiplication:
  $\frac{n}{5} \cdot \mathit{Time}[\![\mathit{Cycle}]\!]$ vs. $n \cdot \mathit{Time}[\![\mathit{Cycle}]\!]$.
- Redundant representation of intermediate operands:
  The addition time is independent of n.

Test and Performance

- ES2, 1.2 μm double metal layer CMOS.
- 304,000 transistors, 210 mm^2.
- Yield 8%.
- Functionally correct, 25 MHz clocking frequency.
- Power 2.5 W at 25 MHz.
- $\mathit{Time}[\![\mathit{Exp}]\!](561) = 561 \cdot \frac{561}{5} \cdot 2 \cdot 40$ ns $= 5.0$ ms.
  Actual time is less than 5.5 ms, corresponding to a throughput of more than 100 Kbit/s.

Photo of Processor



Future Work

- Increasing the radix without increasing the multiplication cycle time.
- Montgomery's modular multiplication algorithm.
- It is possible to make the complexity of quotient determination *independent* of the choice of radix.
- The multiplication cycle time can be reduced to the delay of a 4-2 adder.

Summary

- Increased speed obtained by algorithmic improvements.
- Parallel exponentiation algorithm.
- Radix 32 modular multiplication algorithm.
- Intermediate operands in redundant carry save representation.
- Further improvements are possible.