VLSI Design

KU Sommersemester 2007
RSA-2048 Implementation
Motivation

RSA: asymmetric cryptography
- Signature generation
- Based on modular exponentiation
  - Integer factorization as underlying “hard” problem

- Encryption:
  - Ciphertext \( c = p^e \mod m \)
- Decryption
  - Plaintext \( p = c^d \mod m \)
- Parameters
  - RSA modulus \( m = p*q \)
  - Keys: public key \( e \), private key \( d \)
    - Encryption key: \( e = 2^{16} + 1 \)
    - Decryption key: \( d \times e = 1 \mod (p-1)*(q-1) \)

RSA needs large wordsizes for sufficient security
- RSA-2048 fits well to AES-128
Modular Exponentiation

Square-and-Multiply algorithm

- Square in each iteration: \( r, r^2, r^4, \ldots \)
- Multiply if exponent bit \( e_i = 1 \)
- \( 1.5 \times 2048 = 3072 \) modular multiplications for RSA-2048

\[ \text{result} = b^e \mod m \]

```java
result = 1;
for (int i = e.length(); i >= 0; i--)
{
    result = (result * result) mod m;
    if (e[i] == 1)
        result = result * b mod m;
}
```
Modular Multiplication

A*B mod M: Only operation in RSA

Difficulty: Wordsizes (2048 bit)
- Digit-serial processing necessary

Difficulty: Modular Reduction
- Division needed to determine quotient
  - Guessing possible

Montgomery Multiplication
- Avoids division
- Needs short precomputation
  - And back-transformation

\[
\begin{align*}
\text{Integer Domain} & : \\
\text{MonMul}(a,R^2) & = a' \\
\text{a mod p} & , b \leftarrow b \\
\text{b mod p} & , a + b \mod p \\
\text{a' mod p} & \\
\text{Montgomery Domain} & : \\
\text{a' = a - R mod p} & \\
\text{b' = b - R mod p} & \\
\text{MonAdd(a',b')} & = (a + b)R \mod p \\
\text{MonMul(a',b')} & = (a - b)R \mod p \\
\text{MonInv(a')} & = a' - R^2 \mod p \\
\end{align*}
\]
Modular Multiplication: Montgomery (Orup-3)

Algorithm 7 Montgomery multiplication algorithm, version 4

Input: \( a, b, k, m \)

Output: \( p = a \cdot b \cdot r^{-1} \mod m \) and \( 0 \leq p \leq 2 \cdot \tilde{m} \)

Require: \( m > 2 \) with \( \gcd(m, 2) = 1 \)

Require: \( k, n \) such that \( 4 \cdot \tilde{m} < 2^{k \cdot n} \) where \( \tilde{m} = (m' \mod 2^k) \cdot m \)

Require: \( r, m' \) with \( 2^{k \cdot n} \cdot r^{-1} \mod m = 1 \) and \( -m \cdot m' \mod 2^k = 1 \)

Require: \( a \) with \( 0 \leq a \leq 2 \cdot \tilde{m} \)

Require: \( b \) with \( \sum_{i=0}^{l-1} (2^k)^i \cdot b_i \) and \( 0 \leq b \leq 2 \cdot \tilde{m} \)

1: \( \tilde{m} = \frac{\tilde{m} + 1}{2^k} \)

2: \( p = 0 \)

3: for \( i = 0 \) to \( n \) do

4: \( q = p \mod 2^k \)

5: \( p = \frac{p}{2^k} + q \cdot \tilde{m} + b_i \cdot a \)

6: end for

7: return \( p \)
Proposed Hardware Architecture

Digit-serial multiplier / exponentiator
- Radix-4: \( q_i, B_i \) have 2 bits
- Carry-save Addition

Datapath:
- \( \text{PPG: } A \times B_i \)
- \( \text{PPG: } M \times q_{i-1} \)
- Adder:
  - \( C_{[i-1]} \)
  - \( S_{[i-1]} \)
- \( \text{Reg: Carry} \)
- \( \text{Reg: Sum} \)

Control:
- \( \text{Reg: B} \)
- \( \text{Reg: A} \)
- \( \text{Reg: M} \)

Interface:
- \( \gg \)
Possible Improvements

Higher Radix
- Higher radix than 4 possible on 10 mm²?

Pre-computations
- Storage instead of computation: 3R, 3B

Booth recoding
- Halving of partial products
- Problems with Orup!

Pipelining
- Increasing clock frequency

Clock gating
- No overhead for multiplexors

Efficiency
- System level
- Algorithm
- Architecture
- Circuit level

Effort
Goals

RSA-2048 hardware
- Explore
  - Different algorithms
    - Optimization of proposed solution
- Estimate
  - Cost of approaches
- Implement
  - Most promising variant
  - Standard-cell test chip

Constraints
- RSA-2048
  - Arbitrary RSA modulus
- Maximum silicon area: 10 mm²
  - After place and route
- Clock frequency > 66 MHz

Selected algorithm
- Hardware-oriented high-level model
- Specification

Optimization
- Highest throughput [Mbit/s]
  - On given silicon area

Also of interest
- Simple architecture
  - “Keep the things easy”
- Interface
  - Keep overhead at minimum
  - Digit-serial interface
    - Same digit size as radix?
- Parameterizability
  - Other wordsizes than 2048
Requirements

Hardware architecture
  – Optimization goal: throughput

High-level model
  1. True high level model
  2. Model reflecting proposed architecture

Cost estimation
  – Target technology: Austriamicrosystems c35 0.35 µm CMOS

Hardware implementation
  – Synchronous design
    • Synchronous clock
    • Asynchronous power-on reset
  – Synthesizable HDL
    • Verilog or VHDL
  – Simulation using TCL
    • TCL scripts
      – No HDL test benches
    • Testdata files

Test chip
  – Standard-cell layout for c35 process

Design document
  – Introduction including motivation
  – Short description of algorithms
  – Hardware estimates
    • Area, time, power consumption
  – Hardware implementation
    • General considerations
    • Architecture and modules
  – Hardware results
    • Area, time, power consumption
    • Picture of layout
  – Comparison of estimates and results
  – Conclusions
  – Literature

KU Journal (“IngenieurInnen-\lagebuch”)
Relevant Literature


4. Austriamicrosystems, 0.35 µm CMOS Libraries (C35), [-> asic.austriamicrosystems.com] oder StudentNet: [thalys.student.iaik.tugraz.at]
Schedule

Mon 26. Feb 2007 12:00, HS i13  IT-Sec Summer Term Kick-Off
Wed 14. Mar 2007 10:00, SR IAIK Assignment Presentation
Wed 21. Mar 2007 10:00, SR IAIK Designflow Presentation
Wed 02. May 2007 10:00, SR IAIK Intermediate Presentation
< Fri 15. June 2007 Submission Deadline (including colloquium)

Deadlines

- Registration <= 16. March 2007 (TUGonline)
- Intermediate <= 30. April 2007 (email submission)
- Final << 15. June 2007 (email + “Abgabegespräch”)
Deliverables Intermediate

High-level model
- Model reflecting architecture (for algorithm with best estimate)

Design document v1.0
- Not more than 10 pages
- Introduction including motivation
- Short description of algorithms
- Hardware estimates (for three algorithms)
  - Area, time, power consumption
- Hardware implementation
  - Proposed architecture (for algorithm with best estimate)
  - Architectural options (speed vs. area)
- Literature

Submission
- Mail to Johannes.Wolkerstorfer@iaik.tugraz.at
Deliverables Final

Project files
  - Cleaned and zipped project directory

Design document v2.0
  - Fully blown design document (~15 pages)
  - In addition to document v1.0:
    • Hardware results
      - Area, time (incl. critical path analysis)
        1. Synthesis results
        2. Results from layout editor
          » Picture of layout
    • Comparison of estimates and results
    • Conclusions

Submission
  - Mail to Johannes.Wolkerstorfer@iaik.tugraz.at
    • Propose a date for colloquium (Abgabespräch)
High-level model

```c
void MixColumn(word8 a[4][4]) {
    word8 b[4][4];
    int i, j;

    for(j = 0; j < 4; j++)
        for(i = 0; i < 4; i++)
            b[i][j] = mul(2,a[i][j])
                    ^ mul(3,a[(i + 1) % 4][j])
                    ^ a[(i + 2) % 4][j]
                    ^ a[(i + 3) % 4][j];

    for(i = 0; i < 4; i++)
        for(j = 0; j < 4; j++)
            a[i][j] = b[i][j];
}
```
High-level Model: Refined Model

Example

```
void ecGf2AffDouble() {   // Point double
    alu.loada(mem[mem_x]);
    alu.inv2();   // x1^-1
    alu.mul2(mem[mem_y]);   // y1/x1
    alu.add2(mem[mem_x]);   // y1/x1 + x1
    mem[mem_t1] = alu.get(); // L (lambda)
    alu.sq2(mem[mem_t1]);   // L^2
    alu.add2(mem[mem_t1]);   // L^2 + L
    alu.add2(mem[mem_a]);   // L^2 + L + a
    mem[mem_t2] = alu.get(); // x3
    alu.add2(mem[mem_x]);   // x3 + x1
    alu.mul2(mem[mem_t1]);   // (x3 + x1)*L
    alu.add2(mem[mem_t2]);   // (x3 + x1)*L + x3
    alu.add2(mem[mem_y]);   // (x3 + x1)*L + x3 + y1
    mem[mem_y] = alu.get(); // y3
    mem[mem_x] = mem[mem_t2]; // x3
    memPrint("DOUBLE");
}
```
Hardware Estimates

Datapath will determine chip size
  – Control usually < 10%

Largest standard cells (here c35 process):
  • Flip-flop (e.g. 350 µm² @ 0.35 µm)
  • Full-adder (e.g. 275 µm² @ 0.35 µm)
  • Multiplexer (e.g. 150 µm² @ 0.35 µm)
  – Determine datapath size

Estimated chip size =
  = (#flipflop *350 + #fulladder *275 + #mux *150)
  + 50% overhead

Estimated maximum clock frequency
  – Depends on critical path
    • Longest combinational path
    • Sum of cell delays
Hardware Implementation

Implementation of proposed architecture

- RTL description
  - Using Verilog / VHDL
  - Bit-accurate, cycle-accurate
  - Verified by simulation
- Synthesis
- Place-and-Route
- Post-layout simulation
  - Power simulation
- Backend verification

Diagram:
- High L Model
  - HDL Model
    - Synthesis
      - Placement
        - Routing

Logo © Cadence
Groups and Accounts

Groups
- Groups of 3 students
  - Register on printed form
- Group members get listed
  - On VLSI web
    - including Email address
- Anyone without a group?

Computer accounts
- 1 account per group
  - Username \textit{vlsi07\_08}
    - for group 8
  - No password / blank
- Without email
- Without regular backup (!)

Computer rooms
- IAIK F1.01
  - Across Seminarraum
- Linux workstations
  - \textit{Dual1.student.iaik.tugraz.at} – \textit{Dual7.student.iaik.tugraz.at}
- Key available on request

Remote access
- SSH + X11
  - \textit{Servitus.student.iaik.tugraz.at}
  - \textit{Dual1.student.iaik.tugraz.at} – \textit{Dual7.student.iaik.tugraz.at}
- File transfer
  - \textit{Thalys.student.iaik.tugraz.at}
Contact

Web

- VLSI Teaching
  http://www.iaik.tugraz.at/teaching/05_vlsi-design/

- IAIK StudentNet
  http://studentinfo.iaik.tugraz.at/

Newsgroup

- lv.vlsi-design
  news://news.tugraz.at/tu-graz.lv.vlsi-design

Johannes Wolkerstorfer

- Johannes.Wolkerstorfer@iaik.tu graz.at
- Tel 0316 873-5515
- IAIK, 1. Stock, Raum F2.10
- Sprechstunden: immer
  - Ca. 9:00 – 18:00
  - Einfach vorbeischauen!
What next?

Read literature

Highest-level model
  – Java or maths program

Make first estimates

Play with IAIK designflow