Reconfigurable Hardware in Modern Cryptography

ECC 2000
October 4–6
Essen, Germany

Christof Paar

Cryptography and Information Security Group
Electrical & Computer Engineering Dept.
and
Computer Science Dept.
Worcester Polytechnic Institute
Worcester, MA, USA
http://www.ece.wpi.edu/Research/crypt

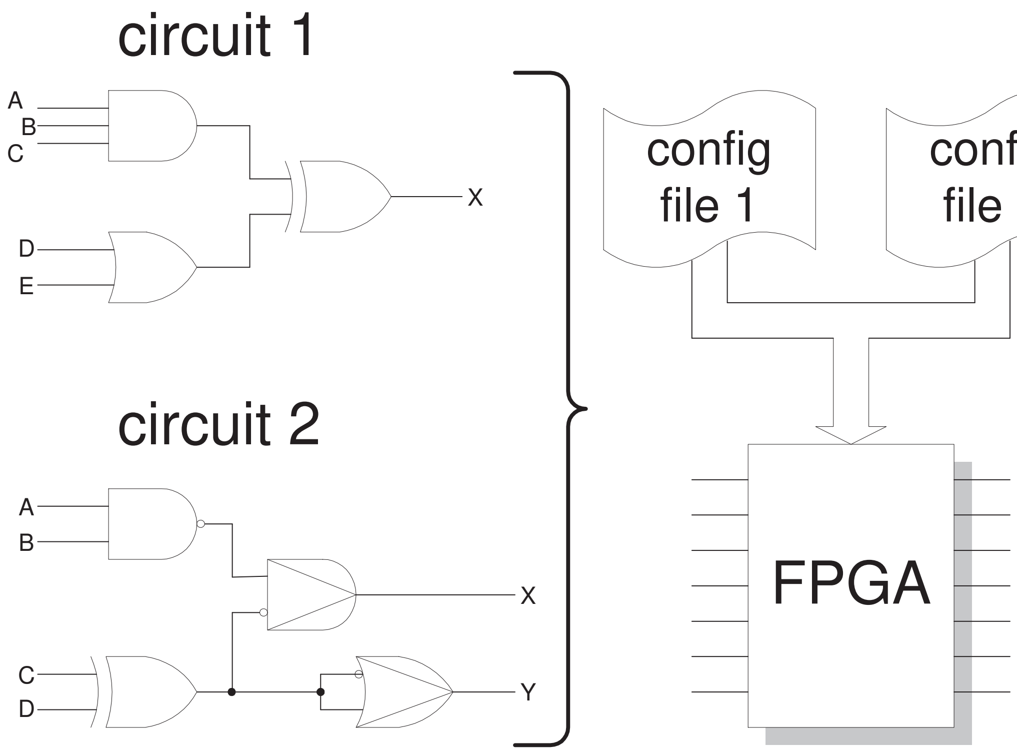
Contents

Reconfigurable Hardware

Reconfigurable Hardware (RCHW) means in commercial applications mostly:

- Field Programmable Gate Arrays (FPGAs)
- Erasable Programmable Logic Devices (EPLD)

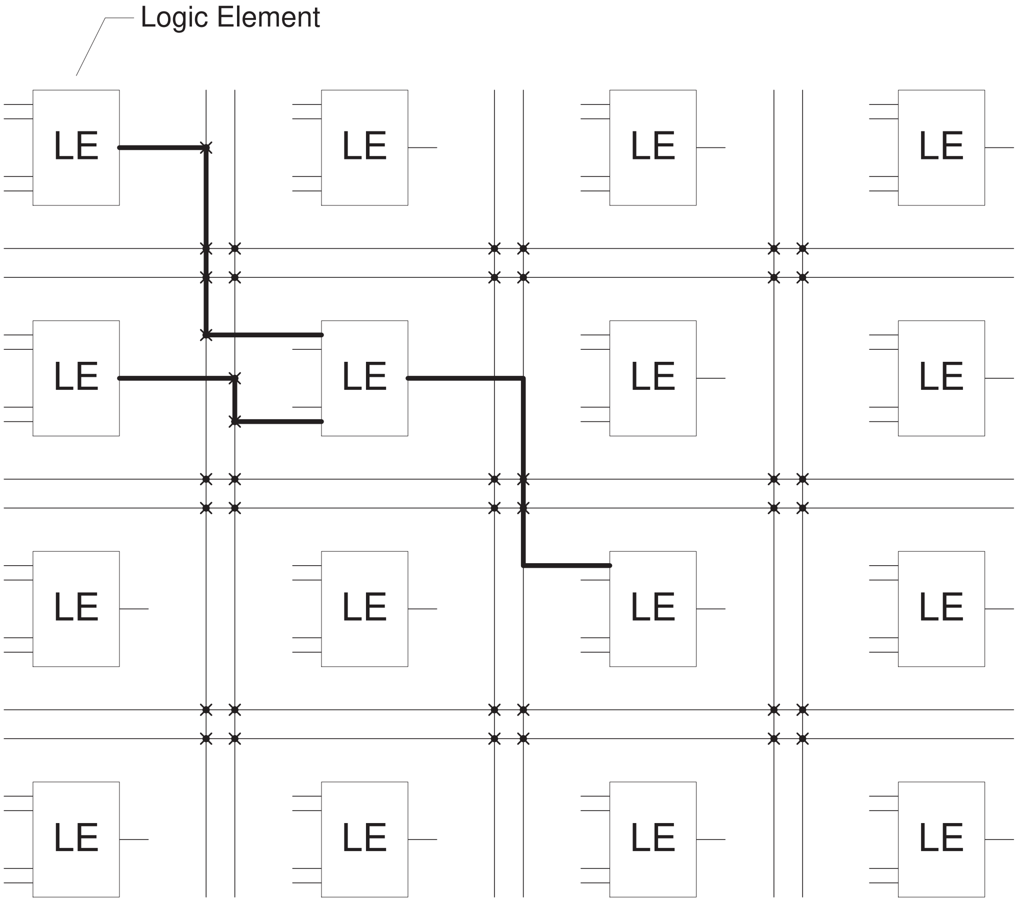
What are FPGAs?



Field Programmable Gate Arrays

- can realize a variety of circuits,
- can be reprogrammed in-system,
- consist of boolean and storage elements,
- can realize fairly large circuits $> 100,000$ gates

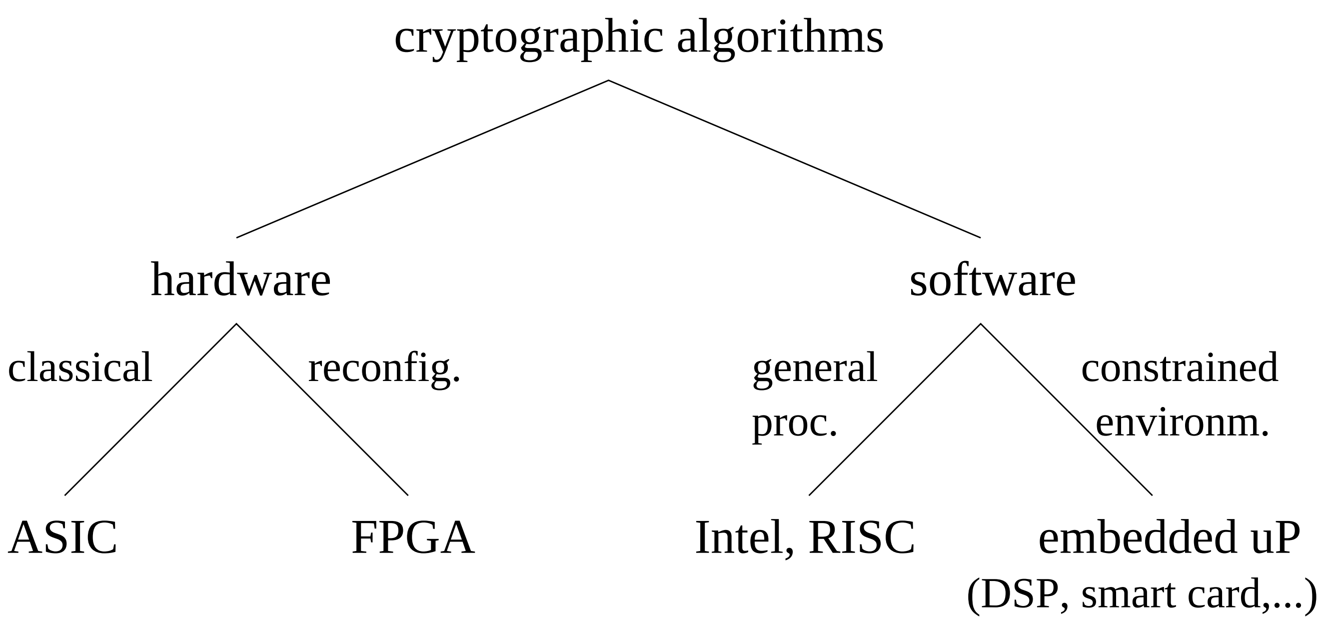
Structure of Commercial FPGAs



- Each logic element contains Boolean logic and registers
- Function of each logic element is programmable.
- Interconnects between elements are programmable.

Implementing Cryptographic Algorithms

Q: Which principle implementation platforms exist?



Choosing a Platform

Choice of implementation is driven by:

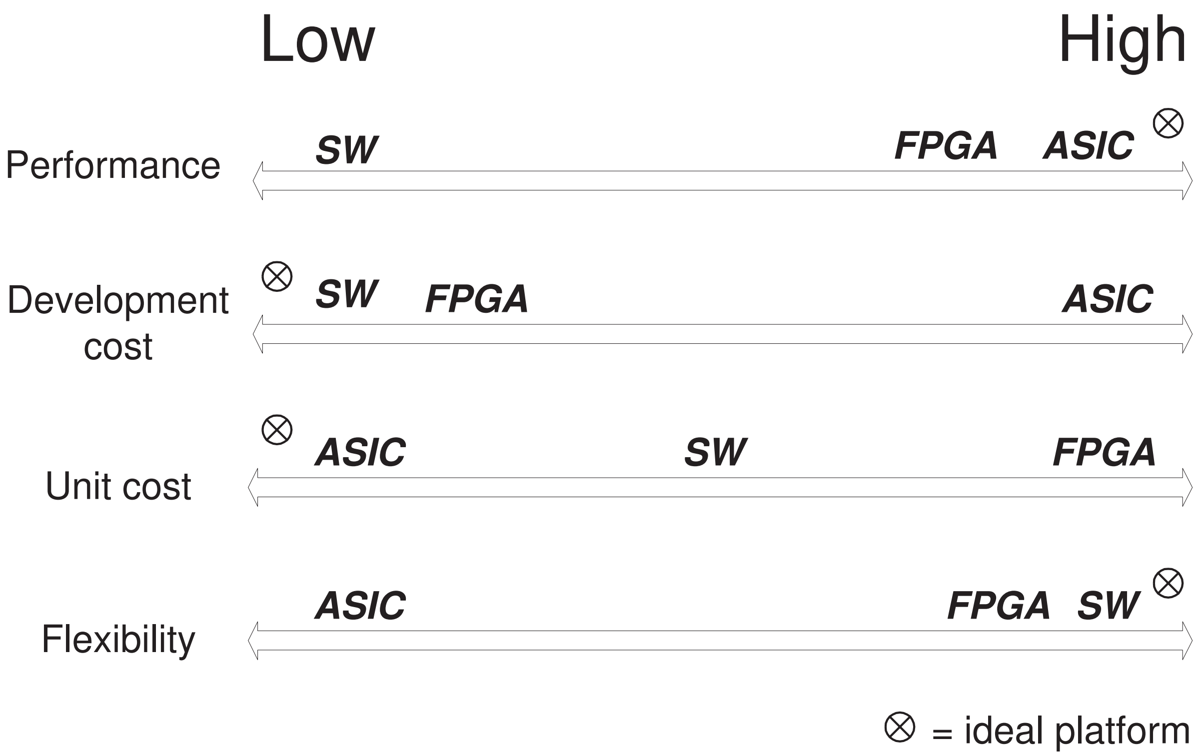
- Algorithm performance
- Cost
  - Per-unit cost
  - Development cost
- Power consumption (wireless devices!)
- Flexibility
  - Parameter change
  - Key agility
  - Algorithm agility
- Physical security

Q: So, which platform is best?

A: Depends on application requirements!

Platform Characteristics

Let's compare hardware/software/FPGA



⇒ To some degree, FPGAs combine the advantageous of SW and HW.

Cryptographic Algorithms: SW vs. HW

Q: Why crypto algorithms in hardware?

Two main reasons:

1. Software implementations are **too slow** for some applications
   (symmetric alg: encryption rates $\leq$ 100 Mbit/sec
   public-key alg: $>$ 10 msec)

2. Hardware implementations are intrinsically more **physically secure**: Key access and algorithm modification is considerably harder.

Q: But why reconfigurable hardware??

Cryptographic Algorithms and Reconfigurable Computing

Potential advantages of crypto algorithms implemented on reconfigurable platforms:

1. Algorithm Agility
2. Algorithm Upload
3. Architecture Efficiency
4. Resource Efficient
5. Algorithm Modification
6. (Throughput relative to software)
7. (Cost Efficiency relative to ASICs)

Crypto and FPGAs: Algorithm Agility

Observation: Modern security protocols are defined to be *algorithm independent*:

- Encryption algorithm is negotiated on a per-session basis.
- Wide variety of ciphers can be required.
- Ex: IPsec-allowed algorithms: DES, 3DES, Blowfish, CAST, IDEA, RC4 and RC6, & future extensions!
- Same holds for public-key algorithms, e.g., Diffie-Hellman and ECDH.

... and: ASIC solutions can provide algorithm agility only at high costs.

Crypto and FPGAs: Algorithm Upgrade

Applications may need upgrade to a new algorithm because:

- Current algorithms was broken (DES)
- Standard expired (again DES)
- New standard was created (AES!!!)
- Algorithm list of algorithm independent protocol was extended

Upgrade of ASIC-implemented algorithm is practically infeasible if many devices are effected or in applications such as satellite communications.

Crypto and FPGAs: Architecture Efficiency

Reconfigurable computing may allow architectures *optimized for specific algorithm instances*.

Exp 1: Key-specific architectures With fixed keys, the main operation in the IDEA cipher degenerates into a constant multiplication which is far more efficient than a general multiplication
[Taylor/Goldstein, CHES '99]

Exp 2: Fixed Galois field arithmetic : Arithmetic architectures for Galois fields tend to be (far) more efficient if field order and irreducible polynomial are fixed.
$\Rightarrow$ FPGAs allow instance-specific architectures.

Ex: Squaring in $GF(2^m)$ takes $m/2$ cycles with a general architecture, but only 1 cycle if the architecture is compiled for one fixed field.

Crypto and FPGAs: Resource Efficiency

Observation: The majority of security protocols uses private-key as well as public-key algorithms during one session, but not simultaneous.

$\Rightarrow$ Same FPGA device can be used for both through run time reconfiguration.

Crypto and FPGAs: Algorithm Modification

Some applications require

- Public algorithms (such as AES candidates) with proprietary modules, e.g., proprietary S-boxes or permutations.
- Change of *modes of operations* (feedback modes, counter mode, etc.)
- Cryptanalytical implementation, such as key-search machines, may use slightly altered version of the algorithms.

With FPGAs, these changes can readily be implemented.

Casestudy I: Elliptic Curve Processor on FPGAs

see also [O/P, CHES 2000]

Design Goals

- (Very) **high performance**
- **Flexible** security levels
- Performance/cost (= speed/area) **scalable architecture**
- **Moderate costs** for medium-volume application

Why Reconfigurable Platform for Elliptic Curve Processor?

Idea: Build optimized architecture for every specific applications.

1. **High performance** Speed-optimized architecture possible for every set of parameters.

2. **Flexible security levels** New architecture for every field order and field polynomial.

3. **Performance/cost scalable** Choose slow+small or fast+large arithmetic units.

4. **Moderate costs** Development using Hardware Description Language (HDL) and unit costs in the US$ 100 range.

Point Multiplication $k\,P$

- Core operation of the Elliptic Curve Processor

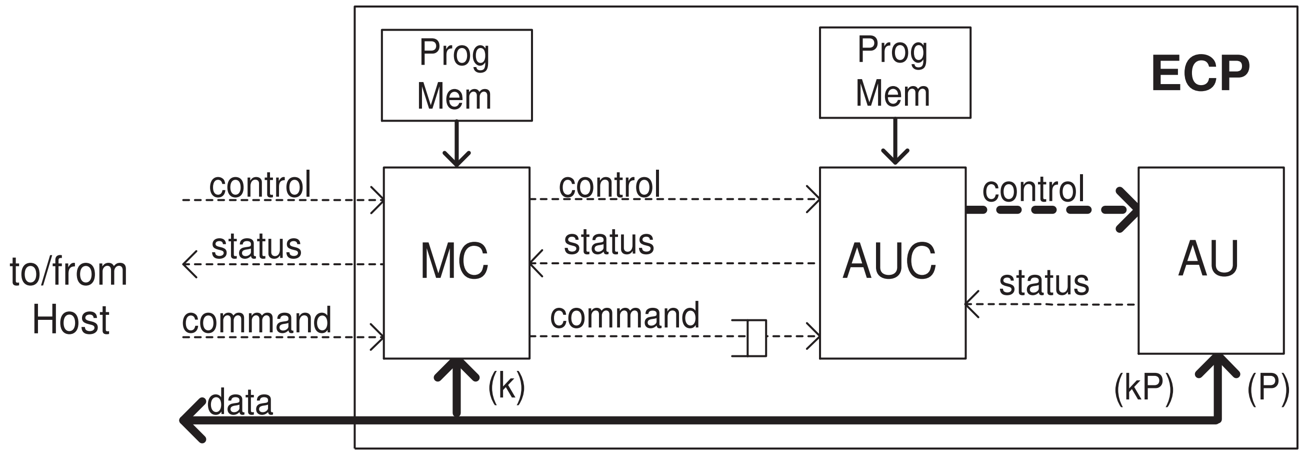
- Two approaches supported:
  - Montgomery point multiplication with projective coordinates
  - Double-and-add with P1363's projective coordinates

Comparison of Point Multiplication Algorithms over $GF(2^m)$

Operation	Montgomery Point Mul (proj coor)	Double-and-Add P1363 (proj coor)
#squares	$5m$	$7m$ (ave.)
#multiplications	$6m$	$10.5m$ (ave.)
#inverses	1	1

Note: Squaring and Multiplication determine system performance.

Block Diagram EC Processor



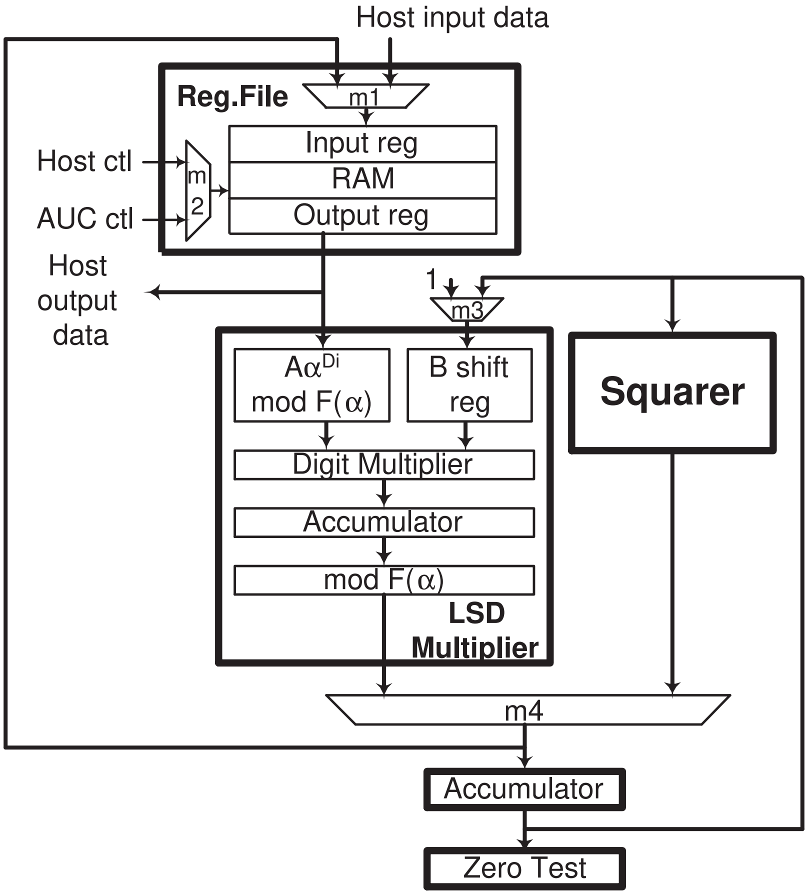
MC — Main Ctrl Point multiplication alg.

AUC — Arithmetic Unit Ctrl Group operations and arithmetic control

AU — Arithmetic Unit add, square, multiply in $GF(2^m)$

Arithmetic Unit

crucial for over-all performance!



- Uses polynomial basis
- Bit parallel squarer (fast!)
- Scalable, digit-serial multiplier
- Large register file allows precomputation algorithms

Squaring in $GF(2^m)$

Field Polynomial	#Gates	#Clock Cycles
fixed	$\leq m$ (trinom.) $\leq 4m$ (pentanom.)	1 1
arbitrary	$\mathcal{O}(m)$	$m/2$

Reconfigurable Hardware allows to treat every field polynomial as "fixed" (i.e., compile new squaring architecture for ever field polynomial)

Digit Serial Multiplier in $GF(2^m)$

- Processes 1 ... D bits per clock cycle (digit size can be configured)
- Multiplication time: $\lceil m/D \rceil$ clock cycles
- Area complexity: $\mathcal{O}(D\, m)$ gates
- Reconfigurable hardware allows choice of specific cost/performance characteristic for every application.

Elliptic Curve Processor Performance

Implementation on Xilinx Virtex XCV400E FPGA

Finite field used: $GF(2^{167})$

Digit Size	Clock (MHz)	Montgomery (msec)	double-and-add (msec)	Speedup rel. to $D = 4$
4	85.7	**0.55**	0.96	1
8	74.5	**0.35**	0.61	1.8
16	76.7	**0.21**	0.36	3.0

Casestudy II: AES Block Ciphers

see also [EYCP, AES 3 Conference]

Q: Why should we use FPGAs for AES?

A: If more than one AES algorithm should run (algorithm agility), one deals with a large number of different atomic operations:

Cipher	**XOR**	**Add**	**Sub**	**Shift**	**Var Rot**	**Mul**	**GF(2^8) Mul**	**LUT**
MARS	•	•	•	•	•	•		•
RC6	•	•		•	•	•		
Rijndael	•			•			•	•
Serpent	•			•				•
Twofish	•	•		•			•	•

Casestudy II: AES Block Ciphers

Q: How do we achieve high performance for the AES candidates?

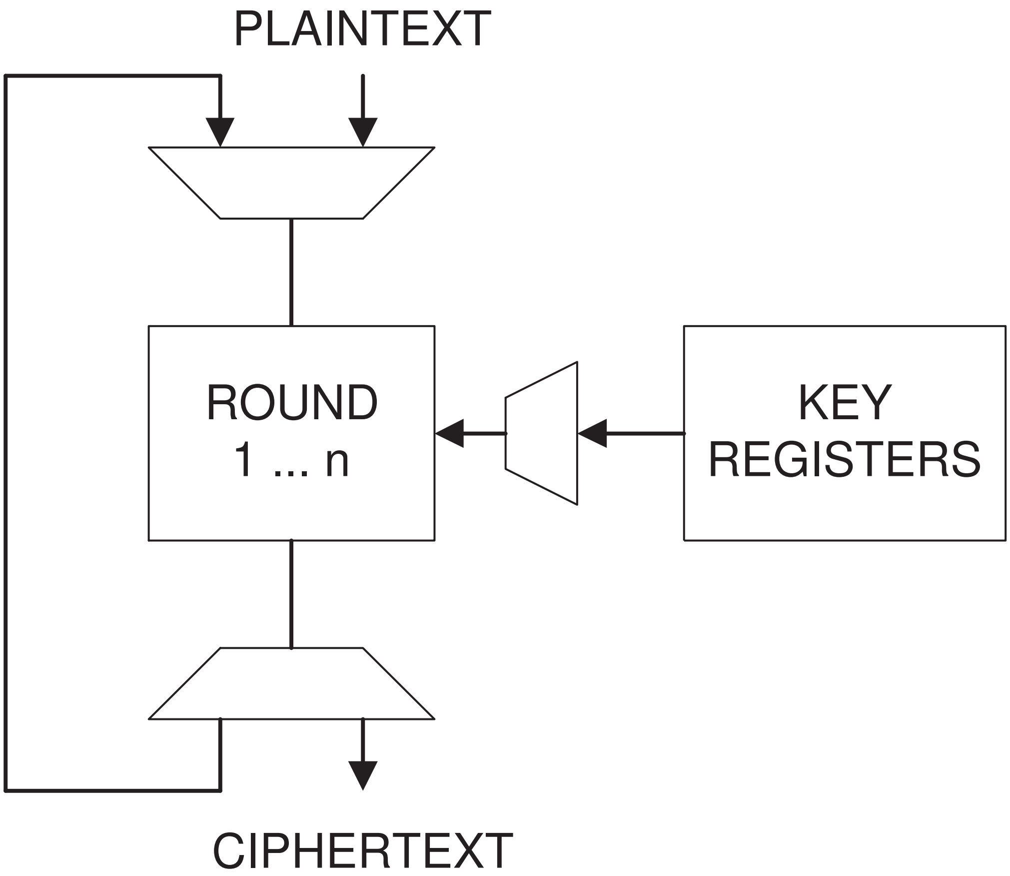
Observation: All AES algorithms are internally based on 32 bit operations
$\Rightarrow$ Software processes 32 bits of data in an inherently serial fashion.

Ideas for high-speed design:

- Process 4×32-bit blocks at a time (parallelization within one round)
- Process multiple data blocks at a time (parallelization of multiple rounds)

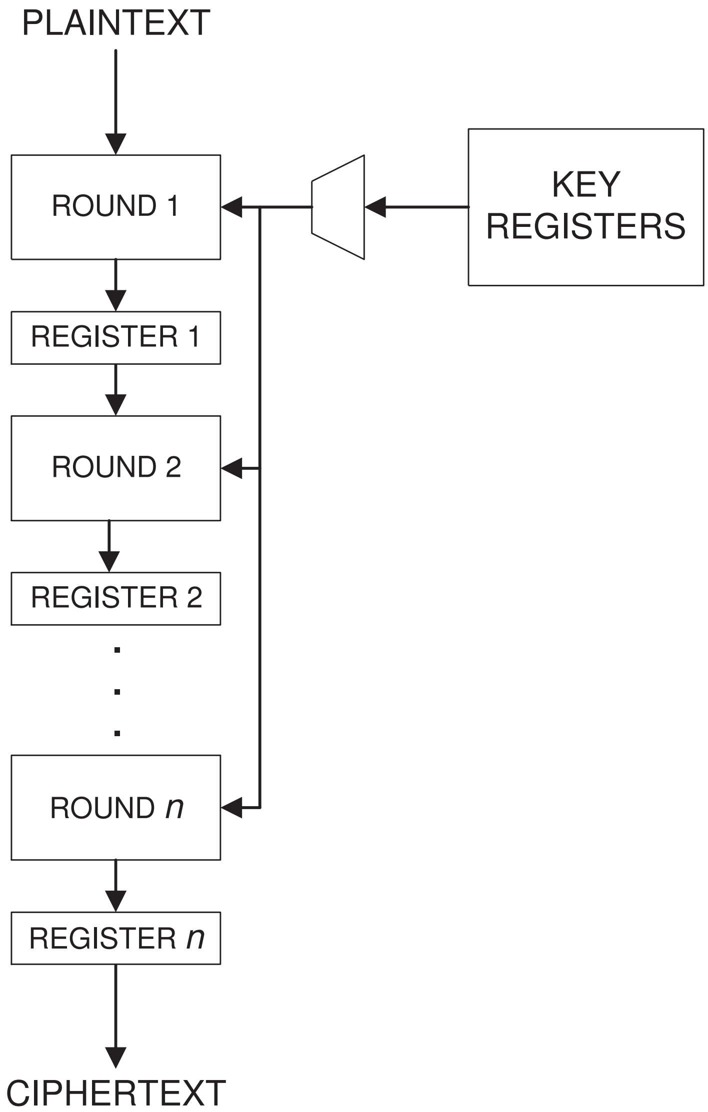
Parallelization within One Round

classical HW architecture w/ iterated blocks:



Pipelining (Parallelization of Multiple Rounds)

classical pipelining:



Results AES Implementation

AES candidates implemented on Xilinx Virtex XCV1000 with various degrees of pipelining.

Alg.	**Throughput (Gbit/s)**	**Slices**
RC6	**2.40**	10856
Serpent	**4.86**	9004
Twofish	**1.59**	9345
Rijndael	**1.94**	10992

Open Problems Related to Cryptography and Reconfigurable Hardware

1. Tamper resistance of FPGAs itself?
2. Threat of unauthorized algorithm manipulation through reconfiguration
3. Added system complexity through reconfiguration capability?
4. Faster reconfiguration times (currently in the msec range)?
5. Unit costs
6. Power consumption
7. Performance advantage of ASIC

Further Reading

[1] S. Brown and J. Rose, "FPGA and CPLD Architectures: A Tutorial," *IEEE Design & Test of Computers*, vol. 13, no. 2, pp. 42–57, 1996.

[2] G. Orlando and C. Paar, "A high performance elliptic curve processor for $GF(2^m)$," in *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2000*, vol. LNCS 1965, (Worcester, Massachusetts, USA), Springer-Verlag, August 2000.

[3] A. Elbirt, W. Yip, B. Chetwynd, and C. Paar, "An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists," in *The Third Advanced Encryption Standard Candidate Conference*, (New York, New York, USA), pp. 13–27, National Institute of Standards and Technology, April 13–14 2000.

[4] J. Goodman and A. Chandrakasan, "An energy efficient reconfigurable public-key cryptography processor architecture," in *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2000*, vol. LNCS 1965, (Worcester, Massachusetts, USA), Springer-Verlag, August 2000.

[5] S. Trimberger, R. Pang, and A. Singh, "A 12 Gbps DES encryptor/decryptor core in an FPGA," in *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2000*, vol. LNCS 1965, (Worcester, Massachusetts, USA), Springer-Verlag, August 2000.

[6] T. Blum and C. Paar, "High radix Montgomery modular exponentiation on reconfigurable hardware for public-key cryptography," *IEEE Transactions on Computers*. to appear.