An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists

AJ Elbirt W Yip B Chetwynd C Paar

Cryptography and Information Security (CRIS) Group
Electrical and Computer Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA

http://ece.wpi.edu/Research/crypt

Presentation Overview

- Motivation
- Methodology
- Hardware Architectures
- Implementation Analysis
- Performance Evaluation
- Conclusions

Motivation

- **Goal:** Efficient Implementation of the AES Candidate Algorithm Finalists in Reconfigurable Hardware
- **Question:** Why Reconfigurable Hardware?
  - Algorithm Agility
  - Algorithm Upload
  - Algorithm Modification
  - Architecture Efficiency
  - Throughput
  - Cost Efficiency

Field Programmable Gate Arrays

- **Reprogrammable hardware**
- Typically SRAM based $\rightarrow$ must be programmed on power-up
- Typically programmed via serial bit-stream
- Typically comprised of routing resources, look-up-tables, and flip-flops

Target Device Selection

- **Target Technology Requirements:**
  - Large I/O capability to support the 128-bit data stream
  - Register-rich architecture to support pipelining
  - High-end device $\rightarrow$ typical FPGA over the AES lifespan
- **Choice:** Xilinx Virtex XCV1000BG560-4
  - 512 usable I/O pins
  - 64 x 96 array of 4-bit Configurable Logic Blocks (CLBs)
  - Each CLB contains 4 flip-flops $\rightarrow$ over 25k register bits
  - 128K embedded RAM

Methodology

- Considered RC6, Rijndael, Serpent, and Twofish
- Implementation of encryption only with 128-bit master key
- Key scheduling assumed to occur externally
  - Sub-keys are stored in internal registers
  - Device may be reconfigured for key scheduling or decryption
- Bottom-up design and test methodology using VHDL
- Findings are dependent on synthesis and place-and-route tools
- Strong focus on high performance (as opposed to smallest area or throughput/area)

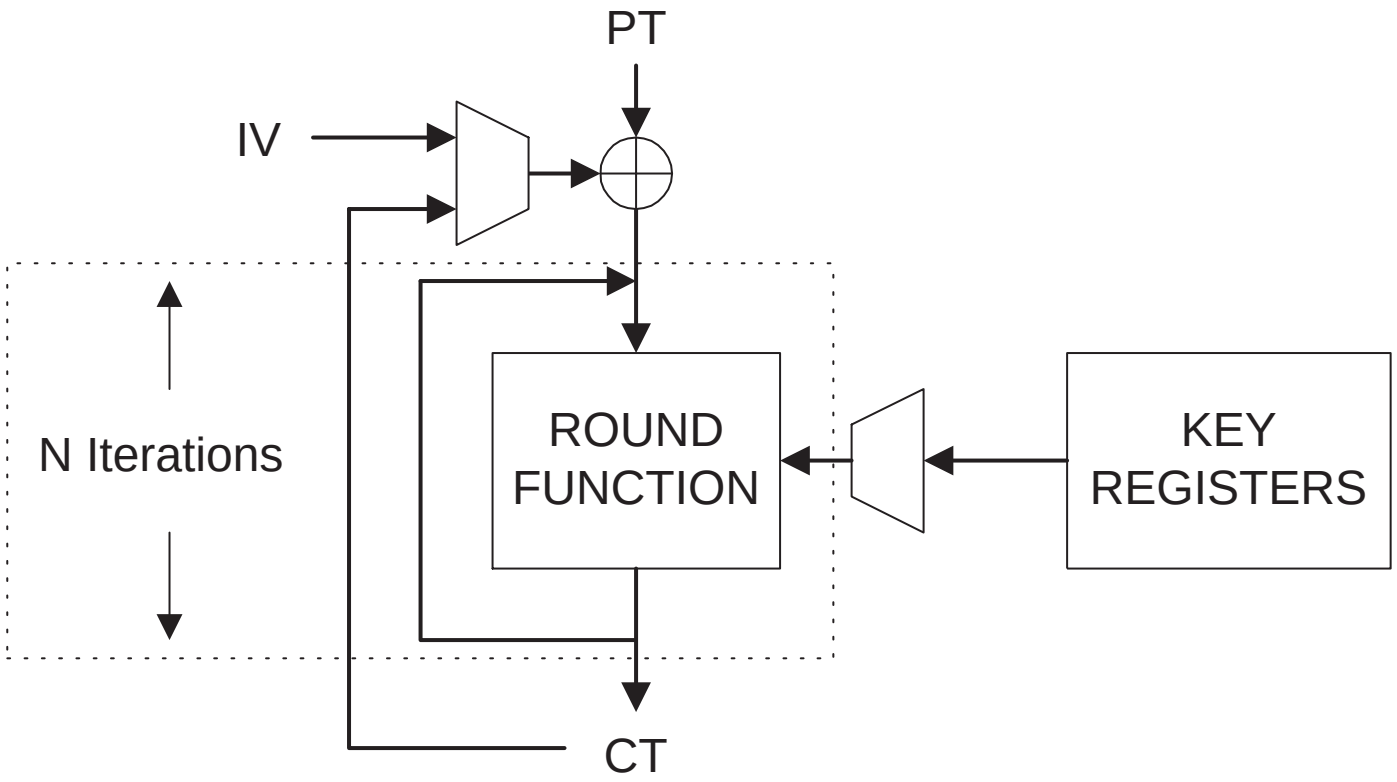
Hardware Architectures

- **Problem:** How to achieve a high performance implementation?
- **Solution:** Implement multiple architectures to find the best solution
  - Iterative looping
  - Iterative looping with partial loop unrolling
  - Full loop unrolling
  - Pipelining
  - Sub-pipelining

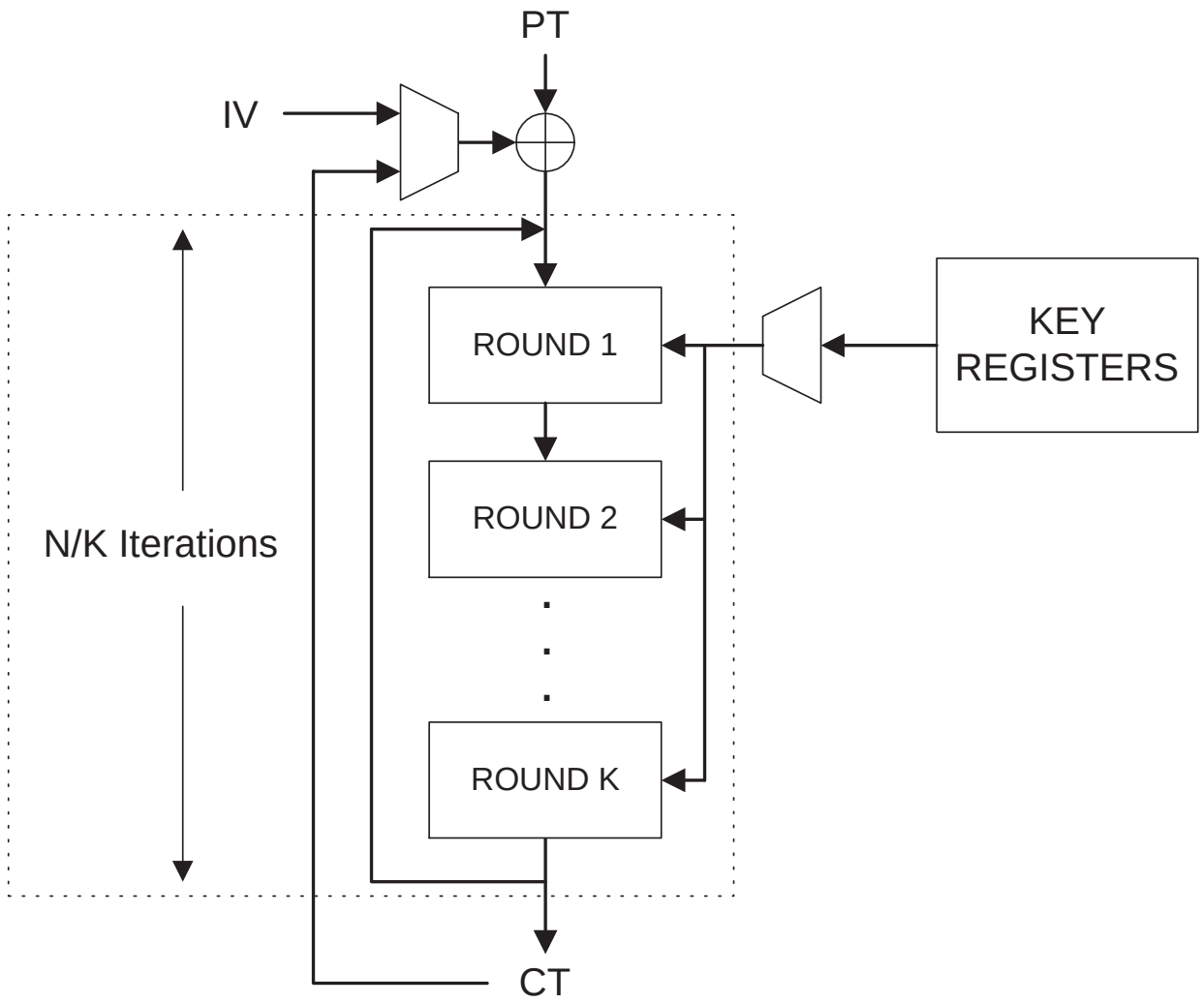
Atomic Operations

Alg	**XOR**	**Mod 2^{32} Add**	**Mod 2^{32} Sub**	**Fix Shift**	**Var Rot**	**Mod 2^{32} Mult**	**GF (2^8) Mult**	**LUT**
MARS	●	●	●	●	●	●		●
RC6	●	●		●	●	●		
Rijndael	●			●			●	●
Serpent	●			●				●
Twofish	●	●		●			●	●

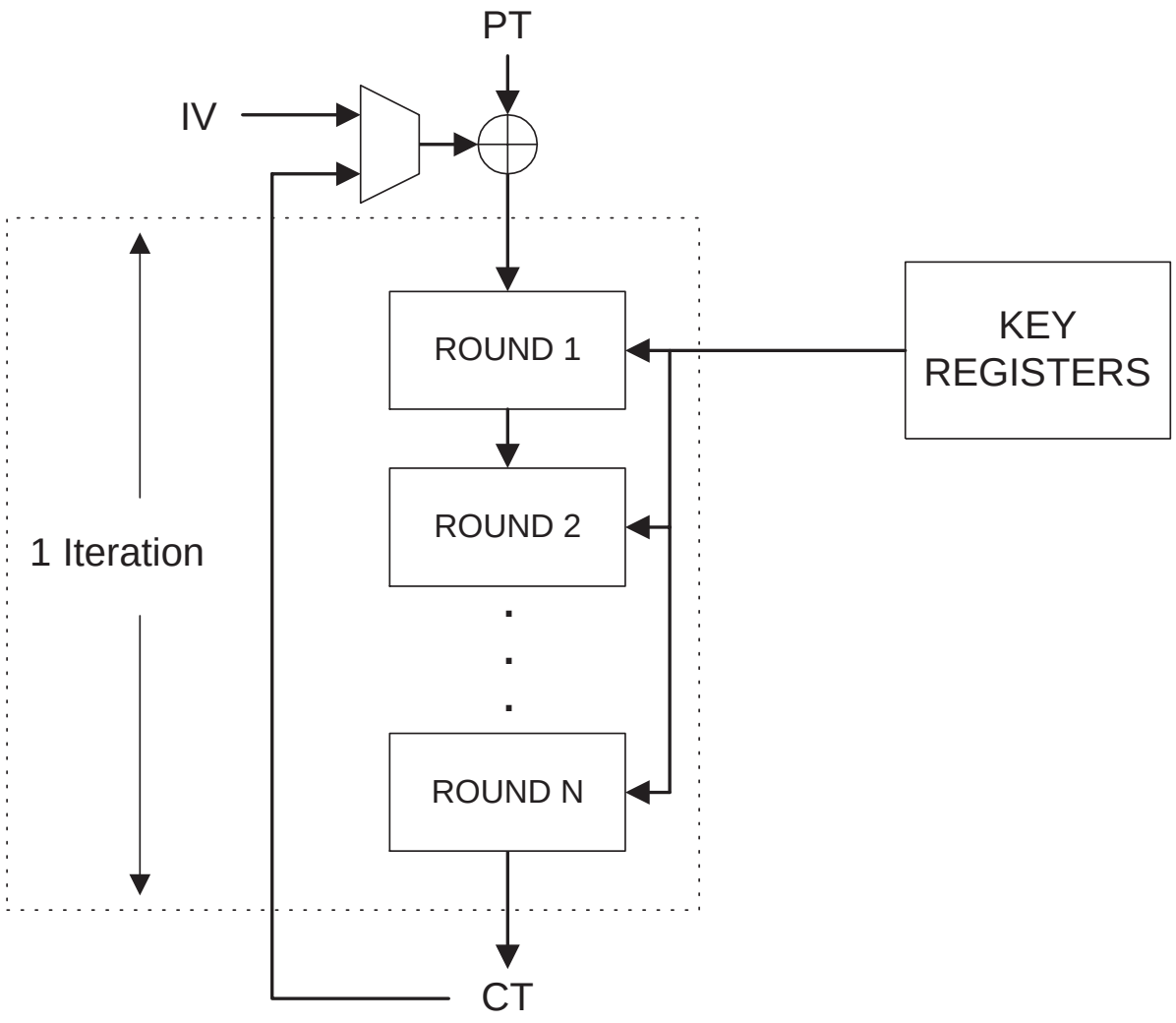
Hardware Architectures — Iterative Looping



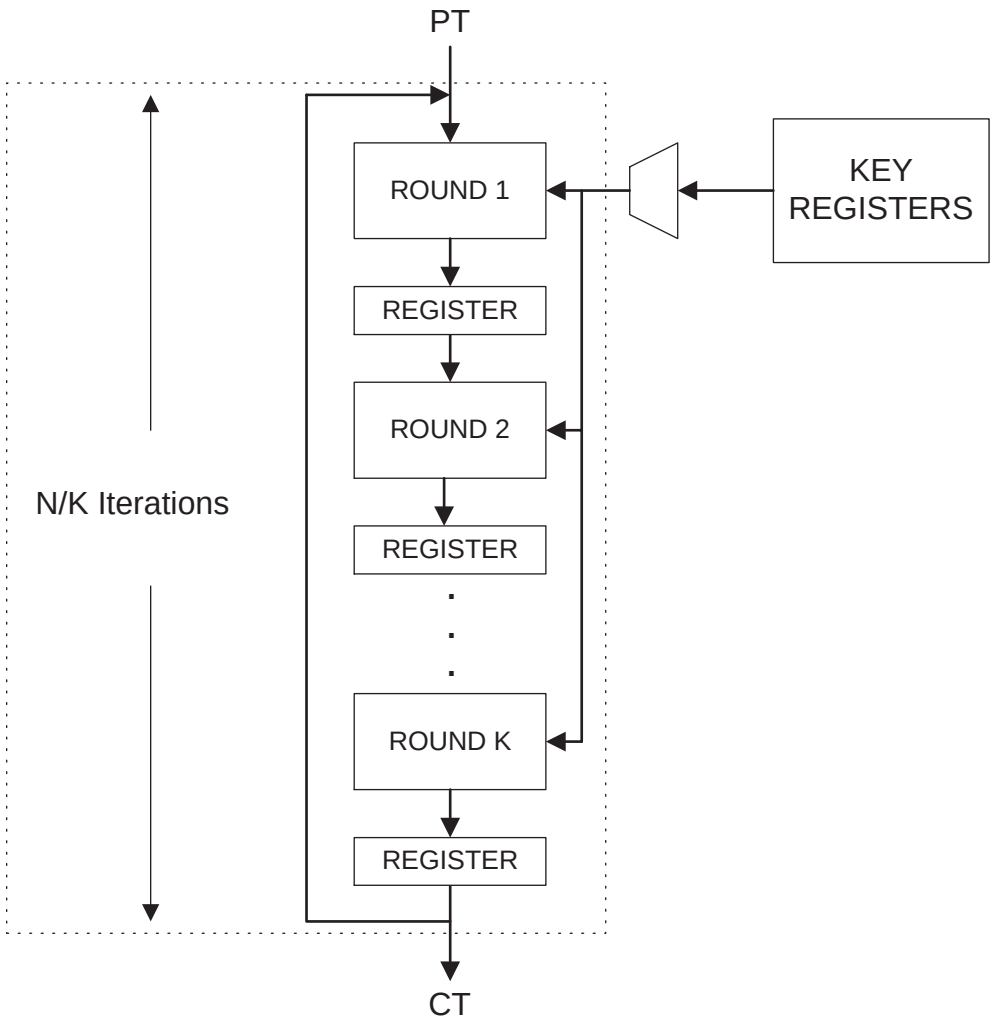
Hardware Architectures — Iterative Looping with Partial Loop Unrolling



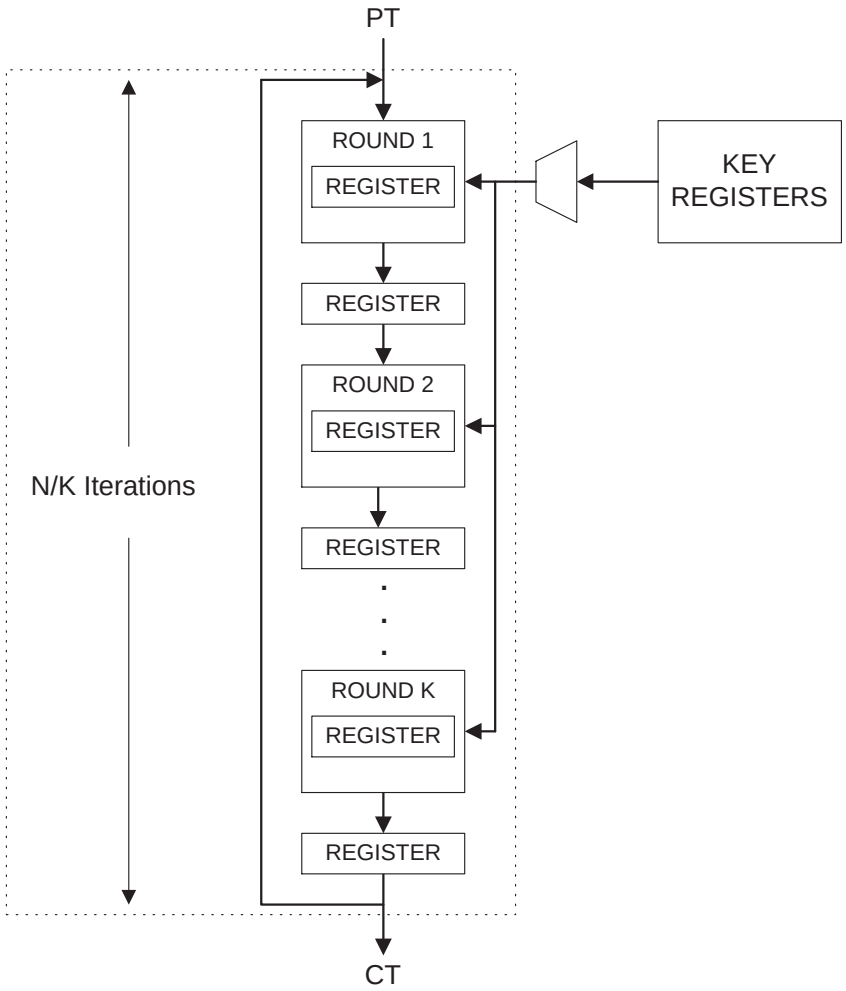
Hardware Architectures — Full Loop Unrolling



Hardware Architectures — Pipelining



Hardware Architectures — Sub-Pipelining



Implementation Analysis — RC6

- Important design characteristics
  - Use squarer not multiplier $\rightarrow 2A^2 + A$ vs. $A(2A + 1)$
  - Variable rotate via 5-level 2-to-1 muxing $\rightarrow$ smaller and faster vs. barrel-shifter
- Maximum of 10 out of 20 rounds fit in the target FPGA
- Smallest implementation $\rightarrow$ 1 Round Iterative Looping
- Best throughput:
  - Non-feedback mode $\rightarrow$ 10 Round Partial Pipeline with 2 Sub-Pipeline stages
  - Feedback mode $\rightarrow$ 2 Round Partial Pipeline

Implementation Analysis — Rijndael

- Important design characteristics
  - Large S-Boxes required for each round (16 8-bit to 8-bit LUTs)
  - Constant GF multiplication maps extremely well to FPGA technology
- Maximum of 5 out of 10 rounds fit in the target FPGA
- Smallest implementation $\rightarrow$ 1 Round Partial Pipeline with 1 Sub-Pipeline stage
- Best throughput:
  - Non-feedback mode $\rightarrow$ 5 Round Partial Pipeline with 1 Sub-Pipeline stage
  - Feedback mode $\rightarrow$ 2 Round Loop Unrolling

Implementation Analysis — Serpent

- Important design characteristics
  - S-Boxes map extremely well to Xilinx CLB slices
  - All round function operations are extremely simple → sub-pipelining yields no performance gain
- Different S-Boxes and linear transformation required for decryption
- Full loop unrolling and pipelining fit in the target FPGA
- Smallest implementation → 1 Round Iterative Looping
- Best throughput:
  - Non-feedback mode → 32 Round Full Pipeline
  - Feedback mode → 8 Round Loop Unrolling

Implementation Analysis — Twofish

- Important design characteristics
  - Large S-Boxes required for each round (key dependent 8-bit to 8-bit LUTs)
  - Constant GF multiplication maps extremely well to FPGA technology
- Maximum of 8 out of 16 rounds fit in the target FPGA
- Smallest implementation $\rightarrow$ 1 Round Iterative Looping
- Best throughput:
  - Non-feedback mode $\rightarrow$ 8 Round Partial Pipeline with 1 Sub-Pipeline stage
  - Feedback mode $\rightarrow$ 1 Round Partial Pipeline with 1 Sub-Pipeline stage

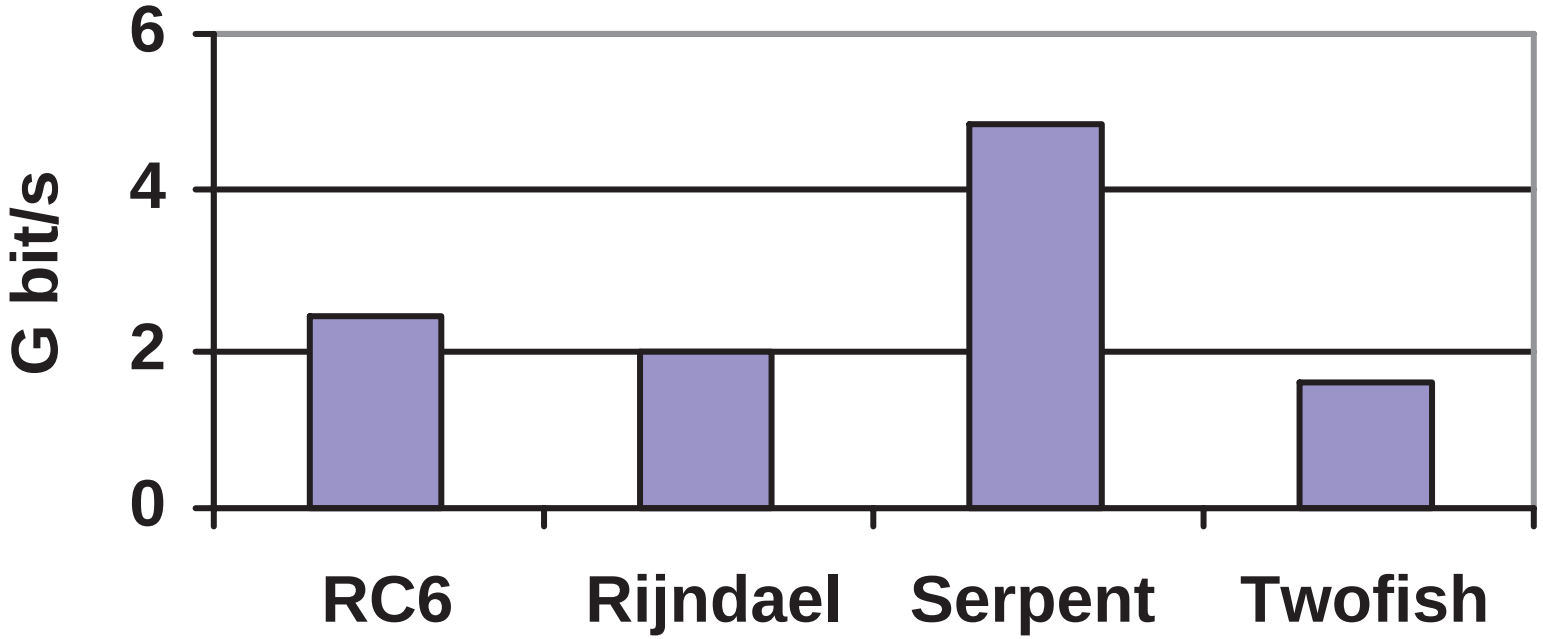
Performance Evaluation

- Throughput Per CLB Slice (TPS) Metric (speed/area)
  - Yields an overall performance-cost value
  - Important to use FPGA slices as opposed to gates for an accurate measurement of chip utilization
  - All designs implemented on the XCV1000BG560-4

Performance Evaluation — Non-Feedback Mode

Alg.	**Arch.**	**Throughput (Gbit/s)**	**Slices**	**TPS**
RC6	SP-10-2	**2.40**	10856	220881
Rijndael	SP-5-1	**1.94**	10992	176297
Serpent	PP-32	**4.86**	9004	539778
Twofish	SP-8-1	**1.59**	9345	169639

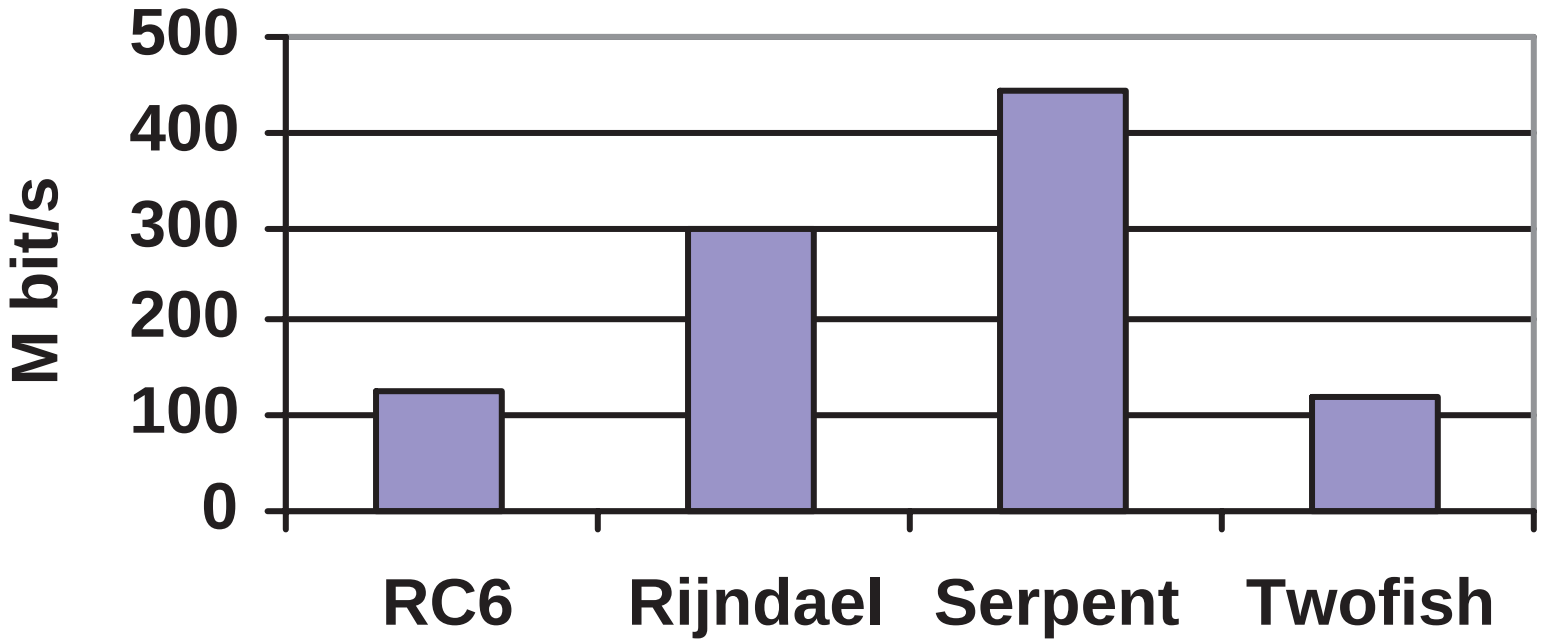
Performance Evaluation — Non-Feedback Mode



Performance Evaluation — Feedback Mode

Alg.	**Arch.**	**Throughput (Mbit/s)**	**Slices**	**TPS**
RC6	PP-2	**126.5**	3189	39662
Rijndael	LU-2	**300.1**	5302	56605
Serpent	LU-8	**444.2**	7964	55771
Twofish	SP-1-1	**119.6**	3053	39169

Performance Evaluation — Feedback Mode



Conclusions

- When **highest throughput** is the optimization parameter:
  - Serpent exhibits by far the best performance in both feedback and non-feedback modes
  - RC6, Rijndael, and Twofish exhibit similar results in non-feedback mode
  - Rijndael outperforms RC6 and Twofish in feedback mode

Conclusions

- Serpent exhibits superior performance in FPGA implementations despite lagging in most software comparisons
- All four algorithms easily achieve Gigabit encryption rates in non-feedback mode via parallelization and wide-operand processing
- **Presented results focus on performance**
- **Algorithm comparison based on area or throughput/area requires different methodology $\rightarrow$ rerun tools with area optimization, architecture modification**