Incremental Reconfiguration for Pipelined Applications

Herman Schmit
Dept. of ECE
Carnegie Mellon University

Motivation

- Forward-compatibility
  - Preserve “software” investment
  - Expectation:
    - Future generations More performance
- Soft limits
  - Expectation:
    - Software runs on any compatible platform

Current State of FPGA Computing

- Process improvements:
  - Faster, bigger FPGAs
  - Cannot exploit increased area w/o redesign
  - Redesign: expensive
- Resource requirements are exact
  - One cell too few $\Rightarrow$ doesn't fit
  - Extra cells $\Rightarrow$ wasted

Our Vision

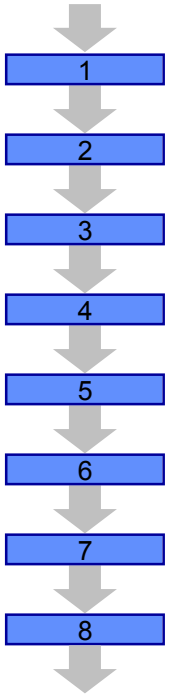
- Design for Infinite hardware
  - "Virtual" hardware design
  - Exploit as much parallelism as possible
- Time-multiplex on real hardware
  - Minimal hardware: functional
  - Increased hardware: higher performance
    - Until Real = Virtual
  - Run-time Reconfiguration (RTR)
- Regularly Pipelined Applications

Overview

- RTR for Pipelined Applications
  - Reconfiguration techniques
    - Component-level reconfiguration
    - Incremental reconfiguration
  - Throughput, Latency, and Memory
- Striped Reconfiguration
  - Support for incremental reconfiguration
  - Concurrent configuration and execution
- Example Application: IDEA encryption
- Conclusions

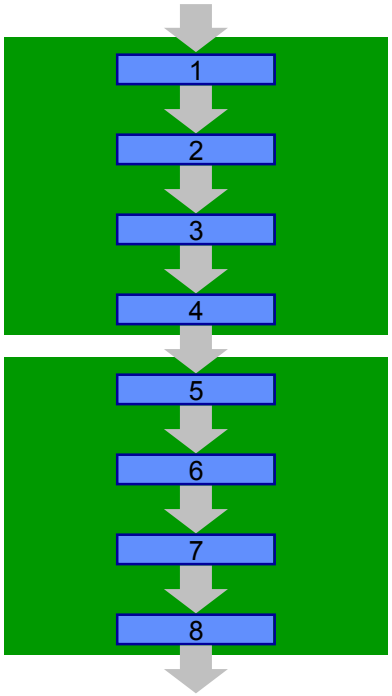
Application Definitions

Application:



S stages
(S = 8)

Static Implementation:



N FPGAs
(N = 2)
T cycle time

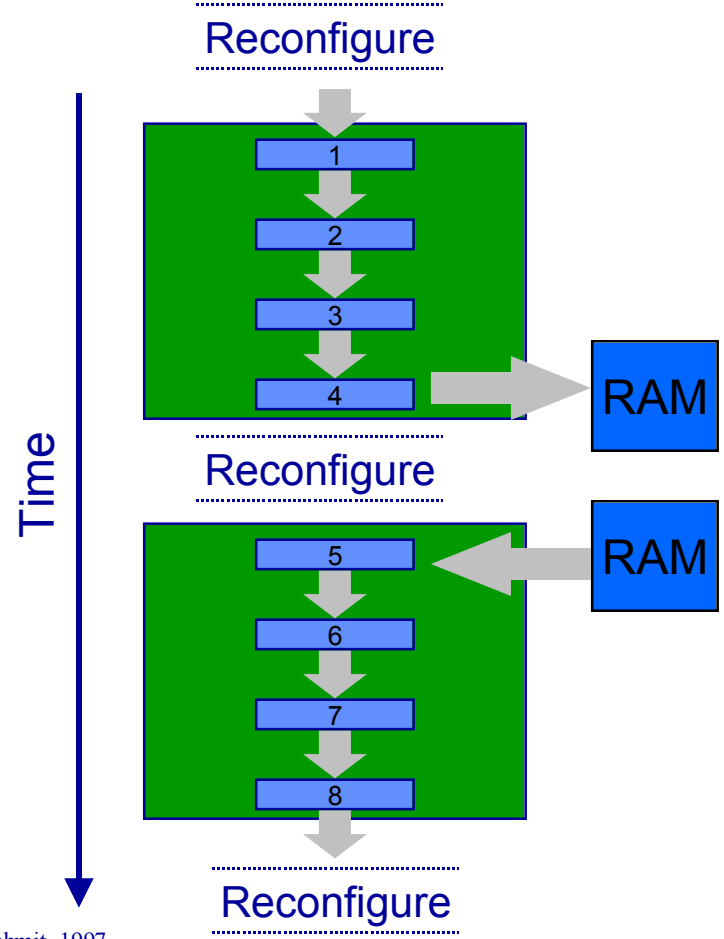
Throughput: $\frac{1}{T}$

Run-Time Reconfiguration

- Assume we only have 1 FPGA
  - Time-multiplex different parts of pipeline
- Ideal Throughput: reduce by factor N

$$\frac{1}{T\ N}$$

Component-Level Reconfiguration



Configuration Time: CT
Data items per config: X

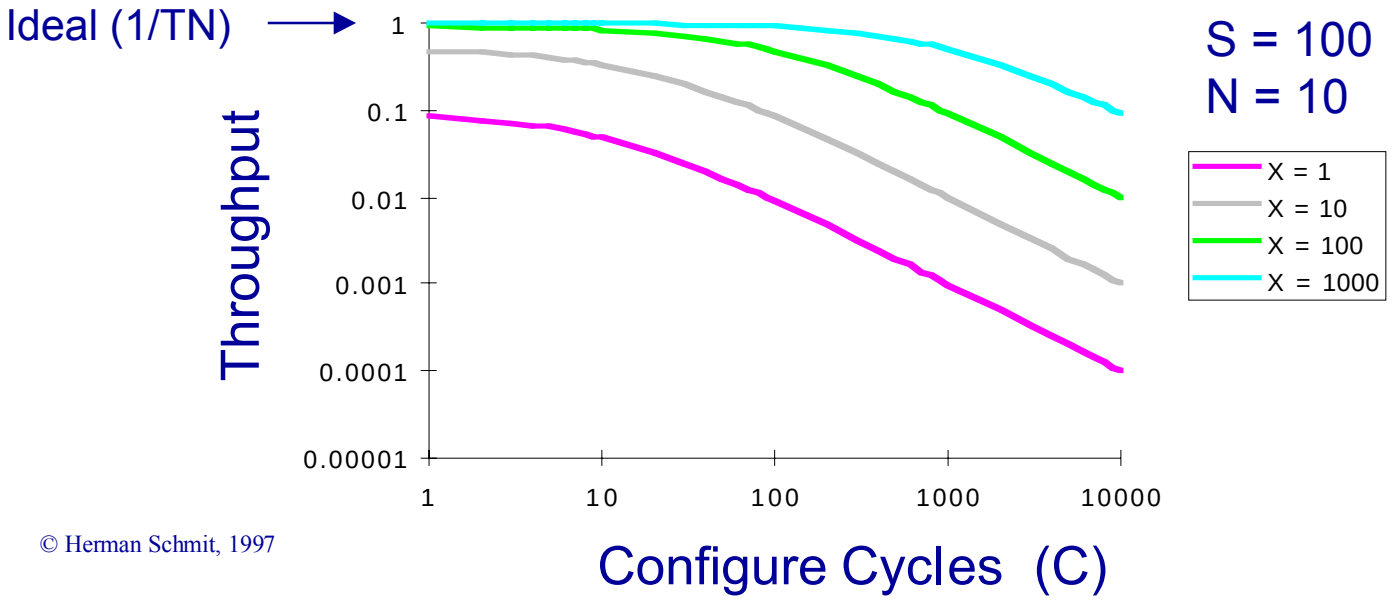
Component-level Throughput

$$\frac{X}{T\,N\,(X + S/N + C)} = \frac{1}{T\,(N + S/X + C\,N/X)}$$

data — X

execute — S/N

configure — C



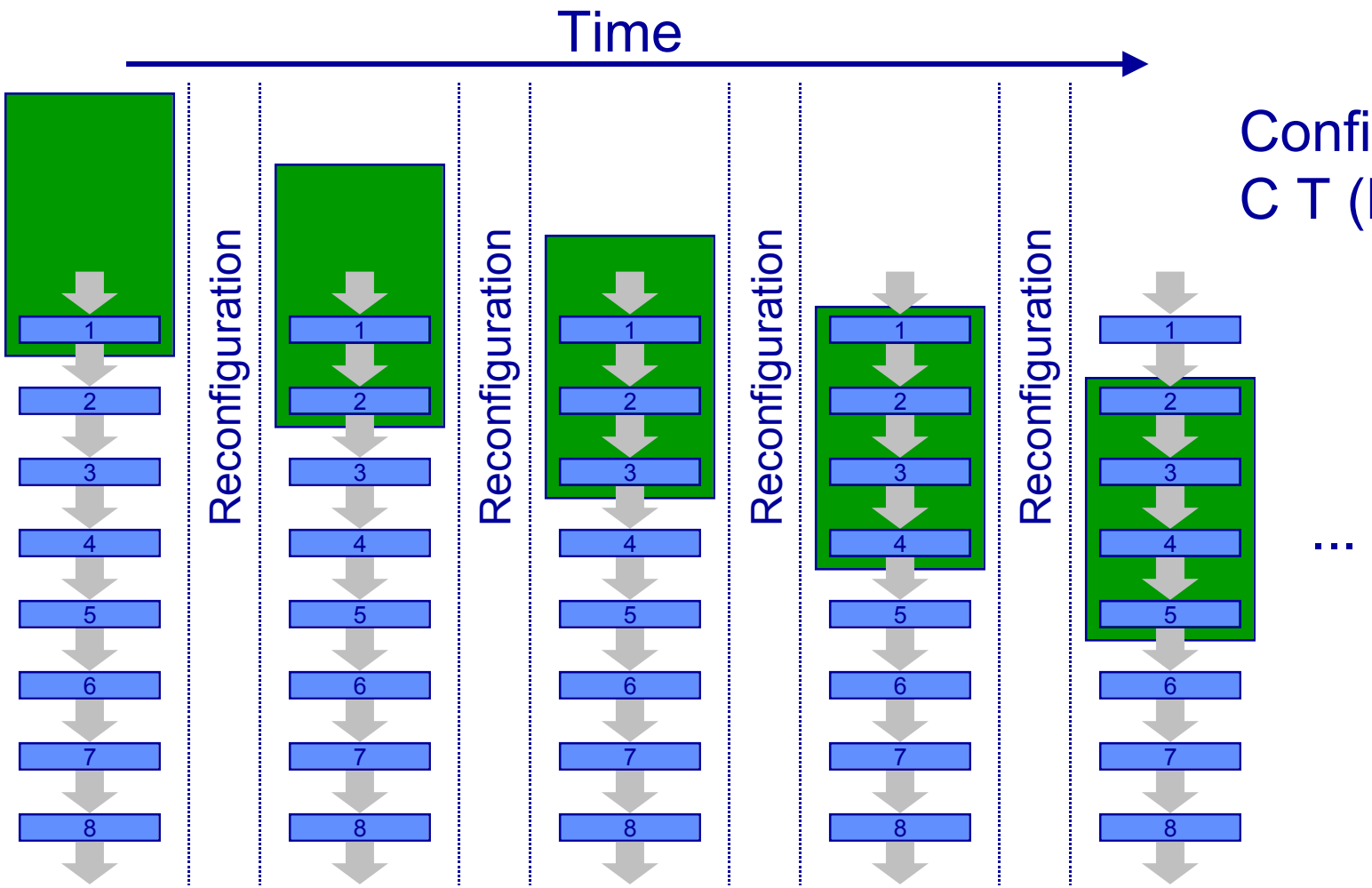
Implementation Issues

- Works for any FPGA
- Reduce C or increase X
- C is large
  - XC4030: C = ~100,000
  - XC6216: C = ~3000
- Multiple Context FPGAs
  - DPGA and Xilinx
  - $C \Rightarrow 0$
  - Still pay the pipeline fill/empty penalty
  - $N \leq$ Contexts

Increasing X

- Memory
  - Increases linearly with X
  - Too large to fit on-chip?
    - Off-chip memory access drives performance
    - Increases T
- Latency
  - Increases linearly with X
  - Real-time applications have latency limitations

Incremental Reconfiguration



Incremental Performance

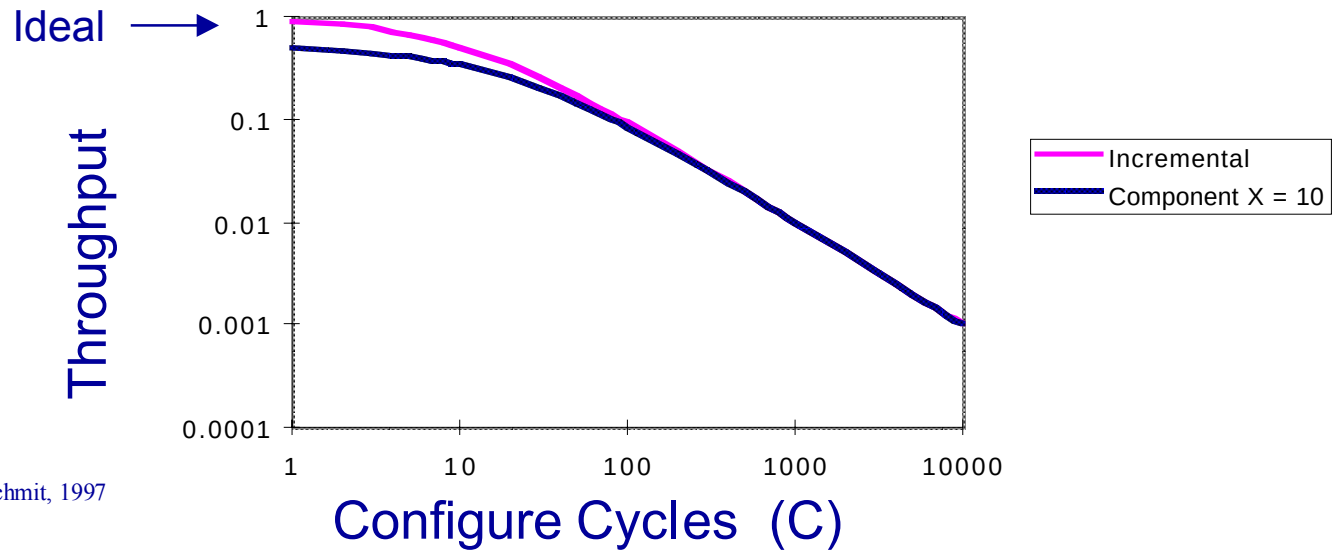
- Throughput:

$$\frac{S / N}{T\,(S + (S / N) - 1) + T\,N\,C} = \frac{1}{T\,(N + N^2\,C / S)}$$

data → S / N

execute → (S / N)

configure → T N C



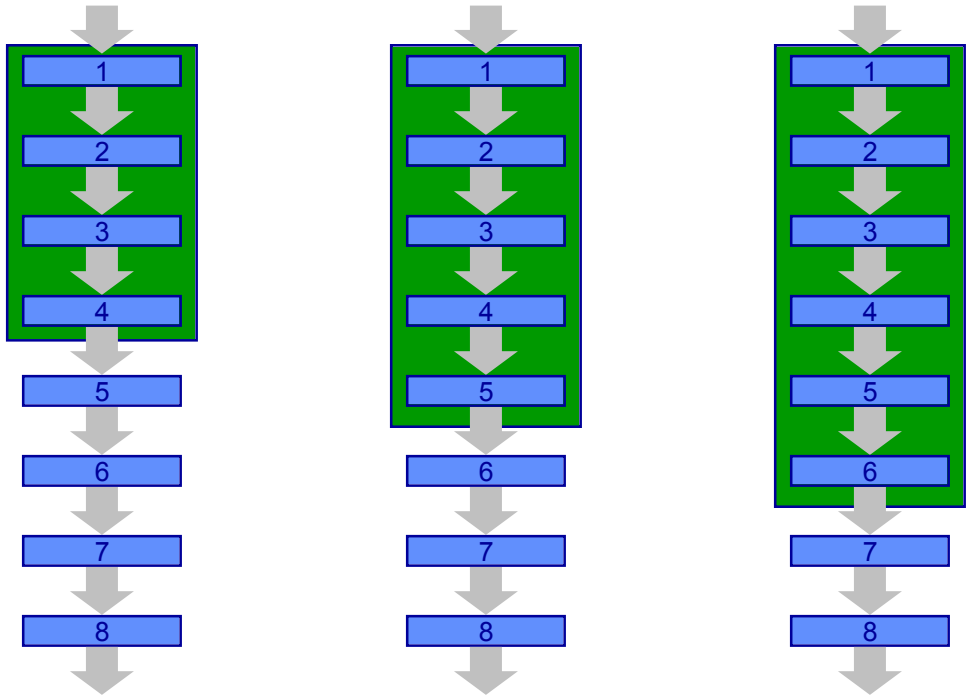
Implementation Issues

- No pipeline penalty
  - Difference important when C is small
- No storage required
  - Intermediate stored in fabric
- Low latency

- Requires:
  - partially reconfigurable FPGA
  - unusual interconnect

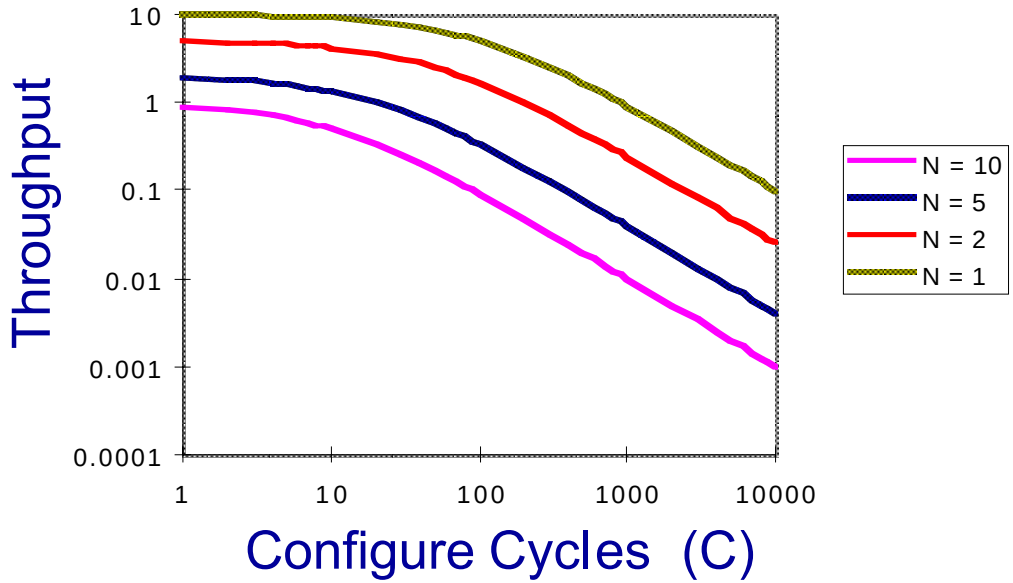
Virtualization

- More physical hardware = more throughput



Virtualization

- Reducing N (more hardware)



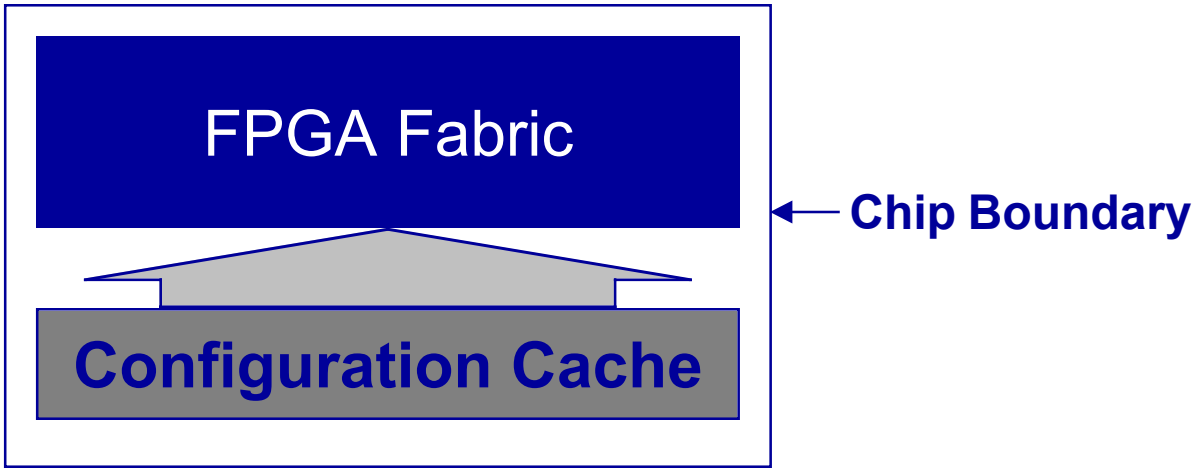
- C is still important

Concurrent Configuration

- How can we reduce C?
- Configuration concurrent with execution
  - Execute stage n, n-1, n-2, ...
  - Configure stage n+1
  - C $\Rightarrow$ 0, Ideal throughput
- No FPGA supports this
- Striped reconfiguration

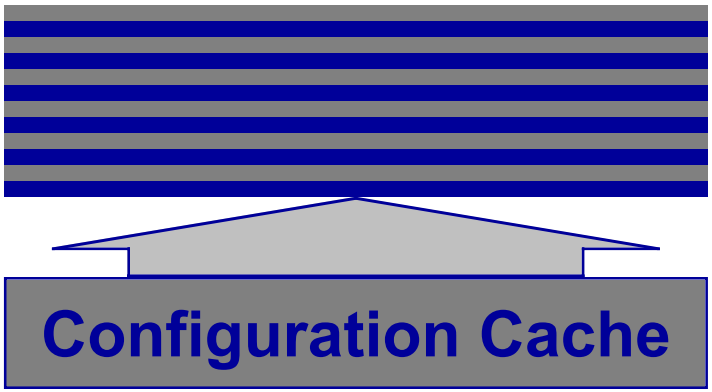
Striped Reconfiguration

- Need to load one pipestage every cycle
  - Store virtual design on-chip
  - Wide configuration bus (~1024 bits)

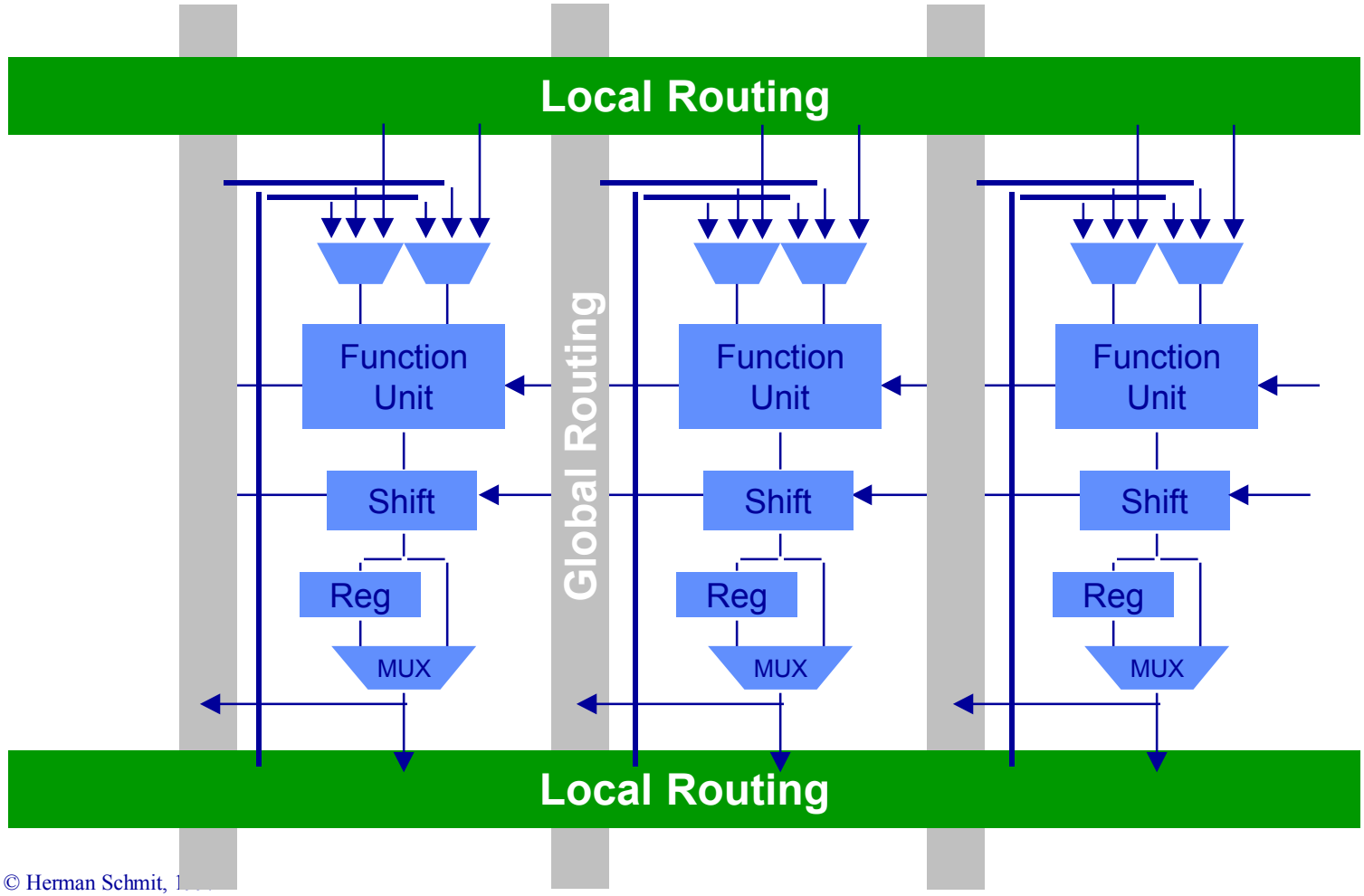


Striped Reconfiguration

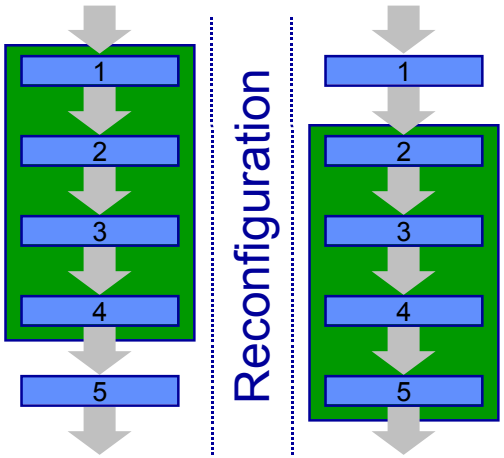
- Load rows (stripes) of the FPGA
- Rows implement pipeline stages
- Uniform interconnect:
  - Global and neighbor
  - Only relative placement is important

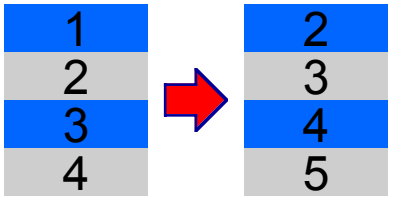


Row Architecture

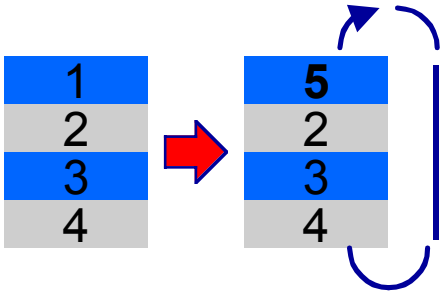


Row Placement and Interconnect

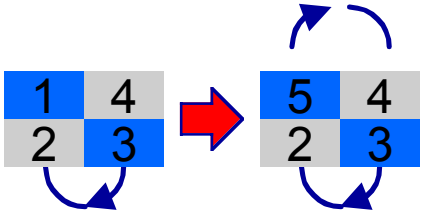




Configuration Moves
Local Interconnect

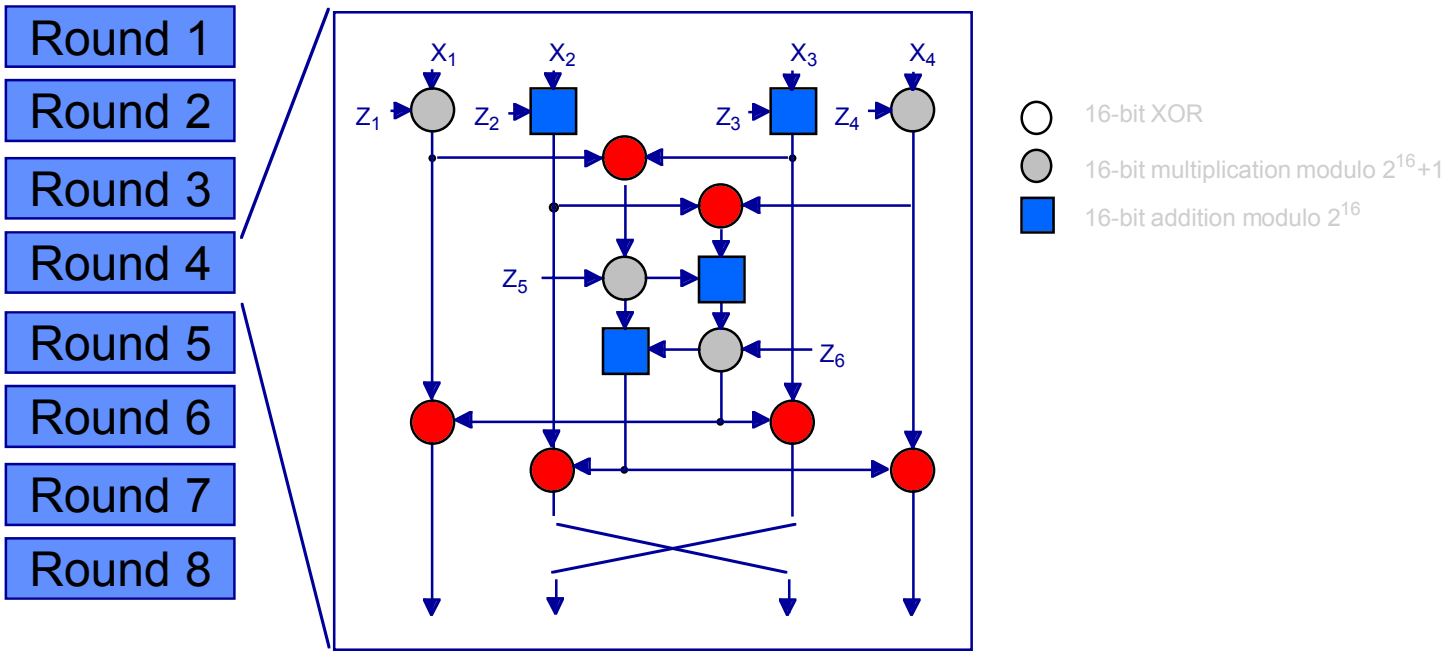


Stationary: SRAM
Interconnect Problem



Stationary: SRAM
Ring Interconnect

Example: IDEA Encryption



- Widely used: PGP
- Completely Pipelineable
- BIG: 32 32-bit multipliers

Accelerating IDEA

- One cm^2 of silicon (0.35μm):
  - 32 rows of active FPGA
  - 256 rows of stored configuration
  - 50 MHz operation
- Deep pipeline:
  - 232 16-bit stages
  - 538 Mb / sec with 32 rows
    - 177 Mb / sec on 25 MHz VLSI chip (1 cm^2, 1.2 μm, 1993)
  - Scales to 3.2 Gb / sec with 232 rows

Other Deeply Pipelined Applications

- Sandia Labs' ATR Algorithms
- Image recognition and understanding
- Image and Signal Processing
- Genetic Algorithms for EDA

Summary

- Incremental Pipeline reconfiguration
  - High throughput, low latency, low memory
- Striped Reconfiguration
  - Concurrent configuration and execution
  - No reconfiguration time
  - Local and global interconnect
    - Ring structure for local interconnect
- Forward-compatiblity, soft resource limits
  - Performance increases until Real = Virtual