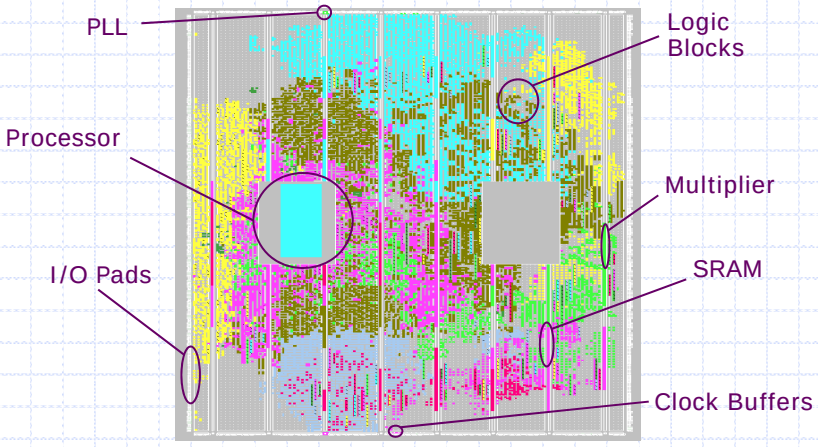
FPGAs

K. Elliott Fleming
Computer Science & Artificial Intelligence Lab
Massachusetts Institute of Technology

FPGA: A Sea of Resources

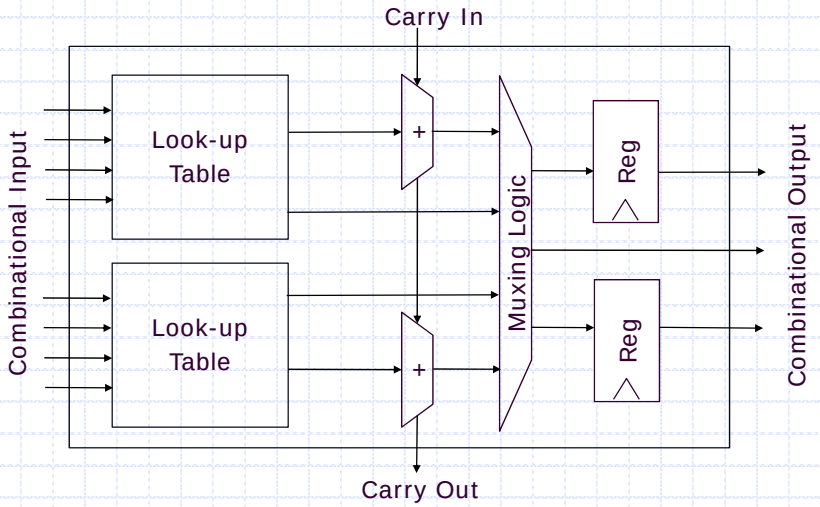


What can we build?

Resource	DE2-70	DE3	802.11ag	SMIPS V2
Logic Elements	68416	135200	85924	6501
Registers	70234	270400	42107	2841
SRAM	250 (4K)	1040 (9k)	265 (9K)	226(4K)
Multipliers	300	576	321	0
Clock Buffers	16	32	7	5
PLL	4	8	1	0
Lines of Code			8762	1603

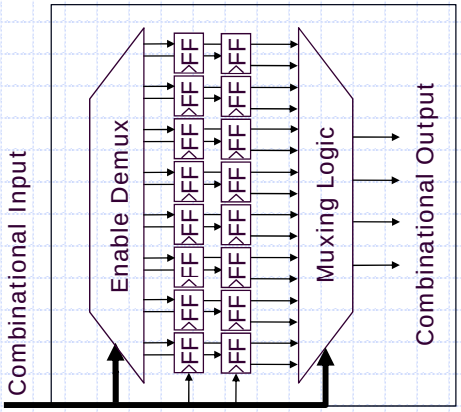
- Very complex systems

Logic Block: Building functionality



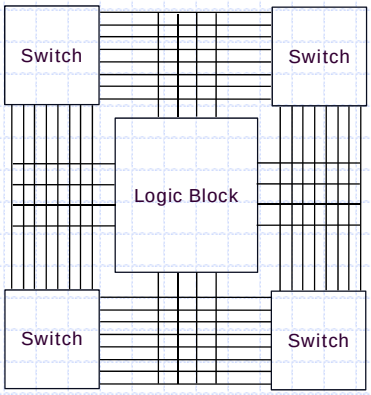
Slice: Look-up Table

- Arbitrary Logic
  - Program flipflops
  - Use inputs to select
- Can we make a ROM?
- Can we make a RAM?
  - Just add enable logic

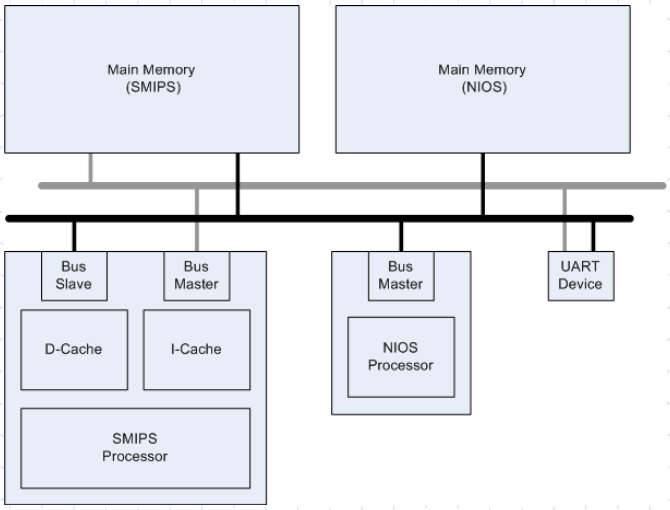


Reconfigurable Wiring

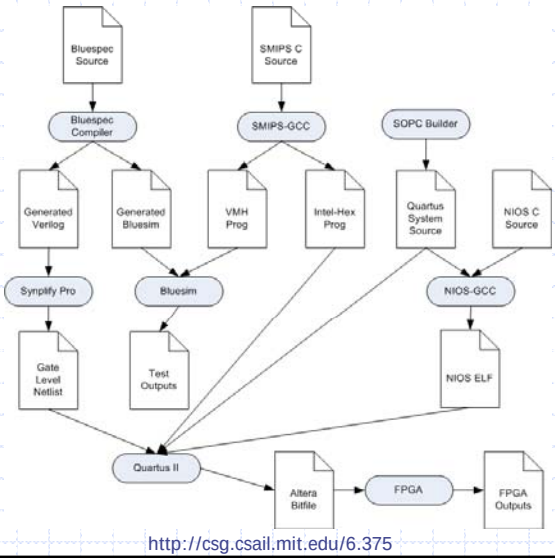
- 2D Mesh Grid
  - Local connections made by driving powerful transistors
  - Switches route across dimensions
- Heterogeneous wire length
  - Many wires to nearby cells
  - Few long-length wires



SMIPS System



SMIPS Infrastructure



SMIPS Infrastructure

- Bus Interface Logic
  - Avalon Master/Slave
- Cbus Devices
  - `mkCBusWideRegRW(addr,reg);`
  - Many interfaces (Get, RegFile, etc.)
  - Mechanism for building memory map automatically
  - Some C drivers included

Demonstration

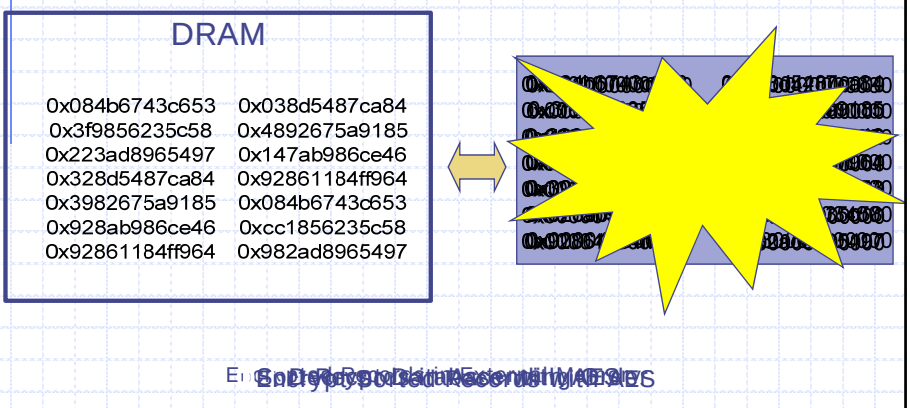
- Synplify Pro
- Quartus II
- Nios-II IDE

Cryptosort: Think Different

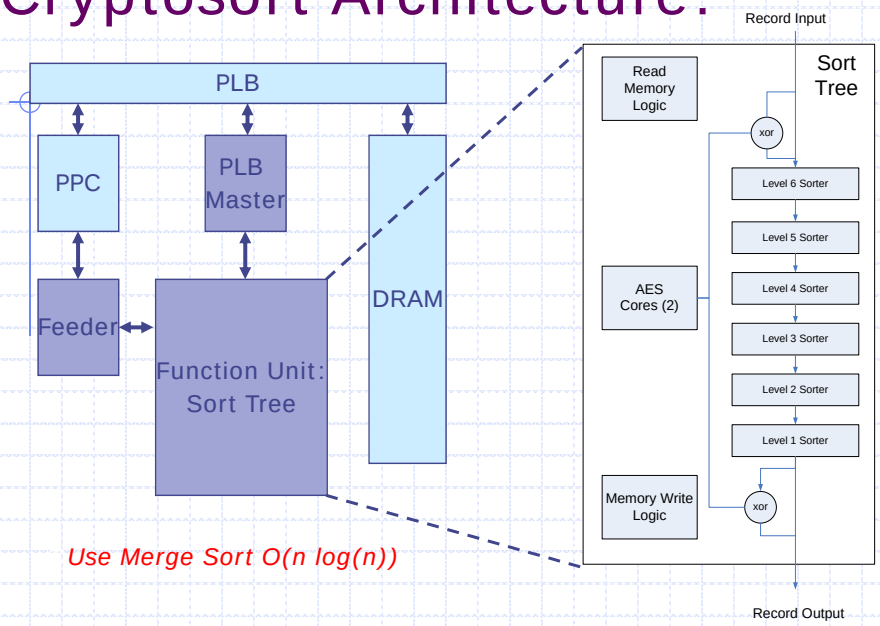
- Large (.5 GB) encrypted database
  - Decrypt Database
  - Sort Database on key
  - Encrypt Database
- Do it fast, on an FPGA
  - Design principals differ from ASIC
  - Must be aware of FPGA hardware

Joint with Myron King, Man Cheuk Ng

From Problem:

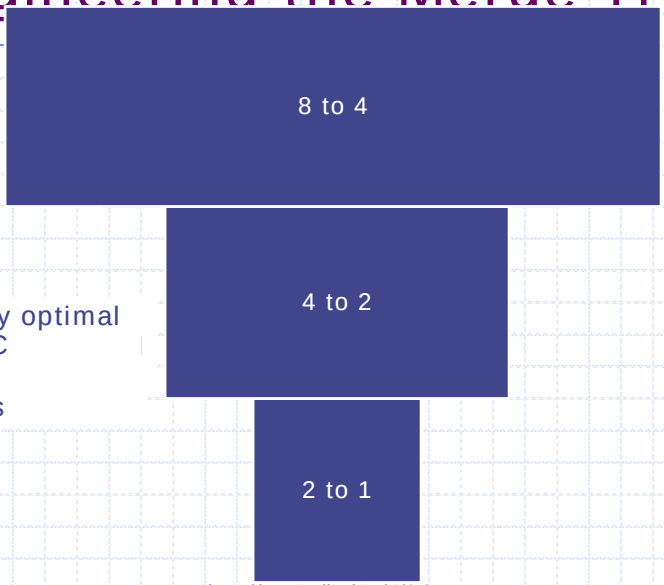


Cryptosort Architecture:



Use Merge Sort O(n log(n))

Engineering the Merge Tree



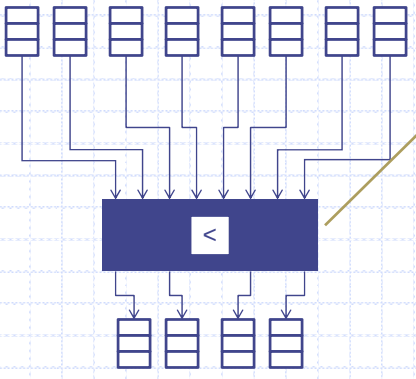
Probably optimal for ASIC tree streams

Refining the Module

- Naïve implementation: exponential resource usage
  - Each comparator takes 3% of slices
  - At most, fit 3 levels
- Key observation:
  - Throughput is rate-limited by final 2-to-1 merge step

This means each level only needs to perform one comparison per cycle

Sharing the Comparator: Idea



Loop:

Choose non-empty input pair corresponding to output fifo with room (scheduling)

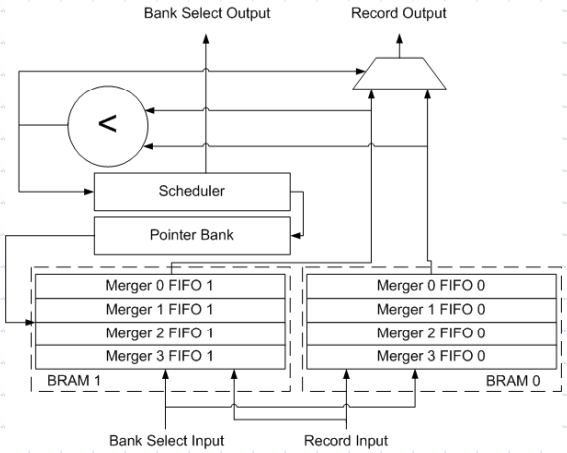
Compare the fifo heads

Dequeue the smaller one and put it on output fifo

But we introduce a comparator scheduling problem

Sharing the Comparator: Physical Implementation Issues

- Not enough regs
  - Each BRAM contains multiple FIFOs
- Aggressive clock
  - Single cycle scheduling is impossible
- Enq happens several cycles after scheduling
  - Credit based flow control



Layout: