
ECE 636

Reconfigurable Computing

Lecture 13

Logic Emulation



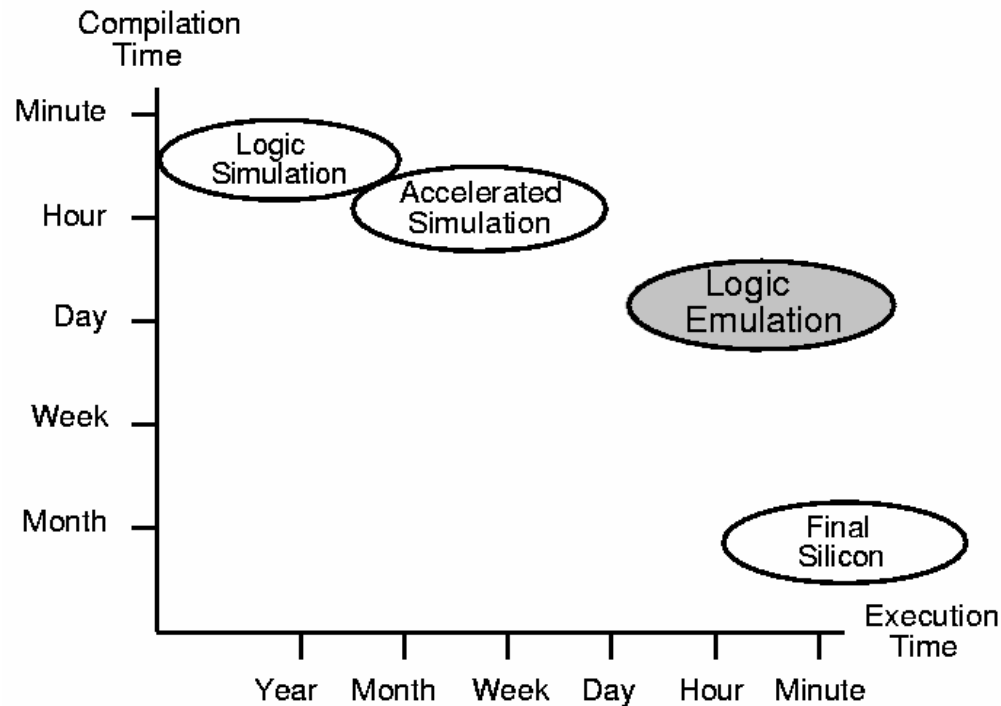
Overview

- **Background**
- **Rent's Rule**
- **Overcoming pin limitations through scheduling**
- **“Virtual wires” implementation**
- **Results / Future work**

The Challenge

- **Making a large multi-FPGA system is easy. Making it programmable is hard.**
- **New approach is a software technology that facilitates hardware implementation.**
- **Effectively make a large number of discrete devices look like one large one.**
- **Leads to low-cost, scalable multi-FPGA substrate.**

Logic Emulation



- **Emulation takes a sizable amount of resources**
- **Compilation time can be large due to FPGA compiles**
- **One application: also direct ties to other FPGA computing applications.**

Are Meshes Realistic?

- The number of wires leaving a partition grows with Rent's Rule

$$P = KG^B$$

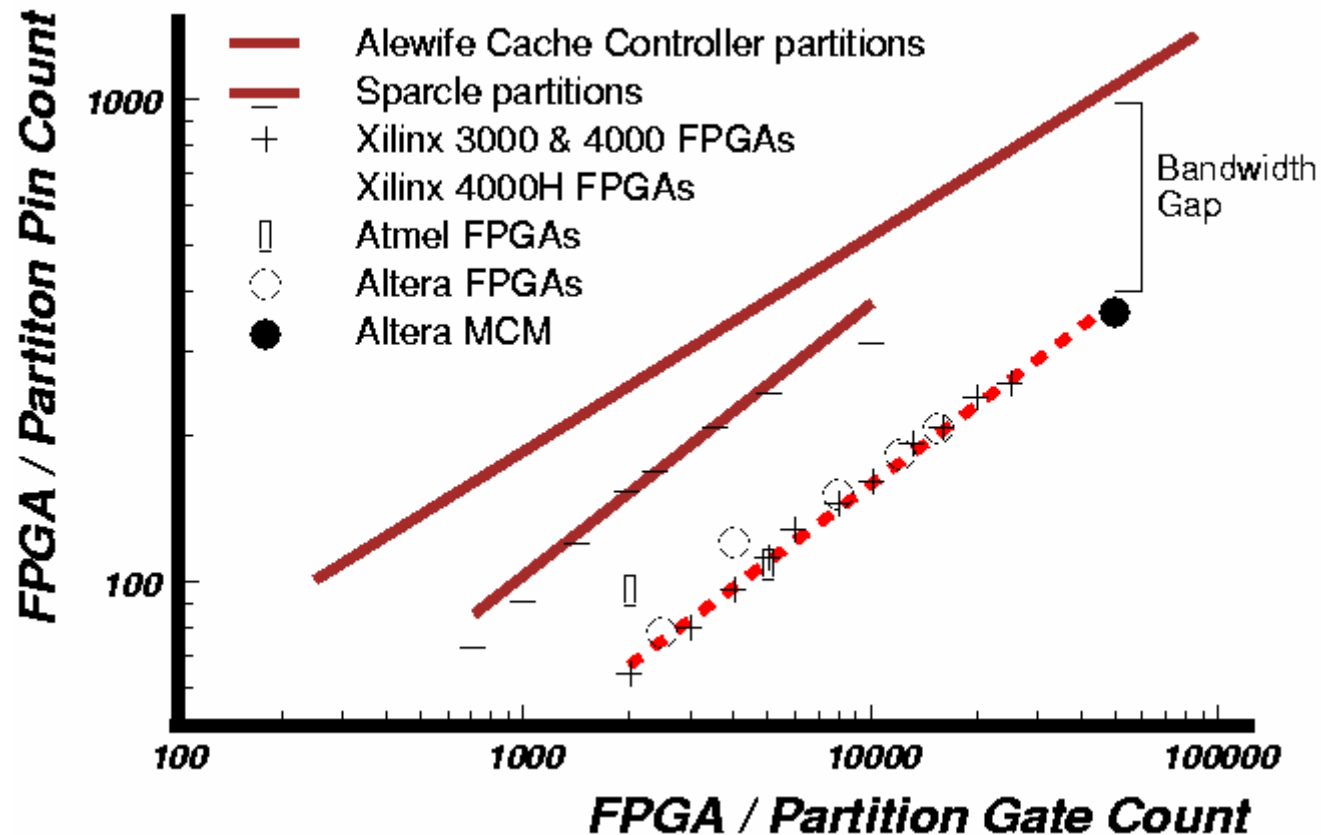
- Perimeter grows as $G^{0.5}$ but unfortunately most circuits grow at G^B where $B > 0.5$
- Effectively devices highly pin limited
- What does this mean for meshes?

Possible Device Scenarios

Not Limited <ul style="list-style-type: none">– unused FPGA pins– unused FPGA gates	Gate Limited <ul style="list-style-type: none">– some unused pins– no unused gates
Pin Limited <ul style="list-style-type: none">– no unused pins– some unused gates	Balanced <ul style="list-style-type: none">– no unused pins– no unused gates

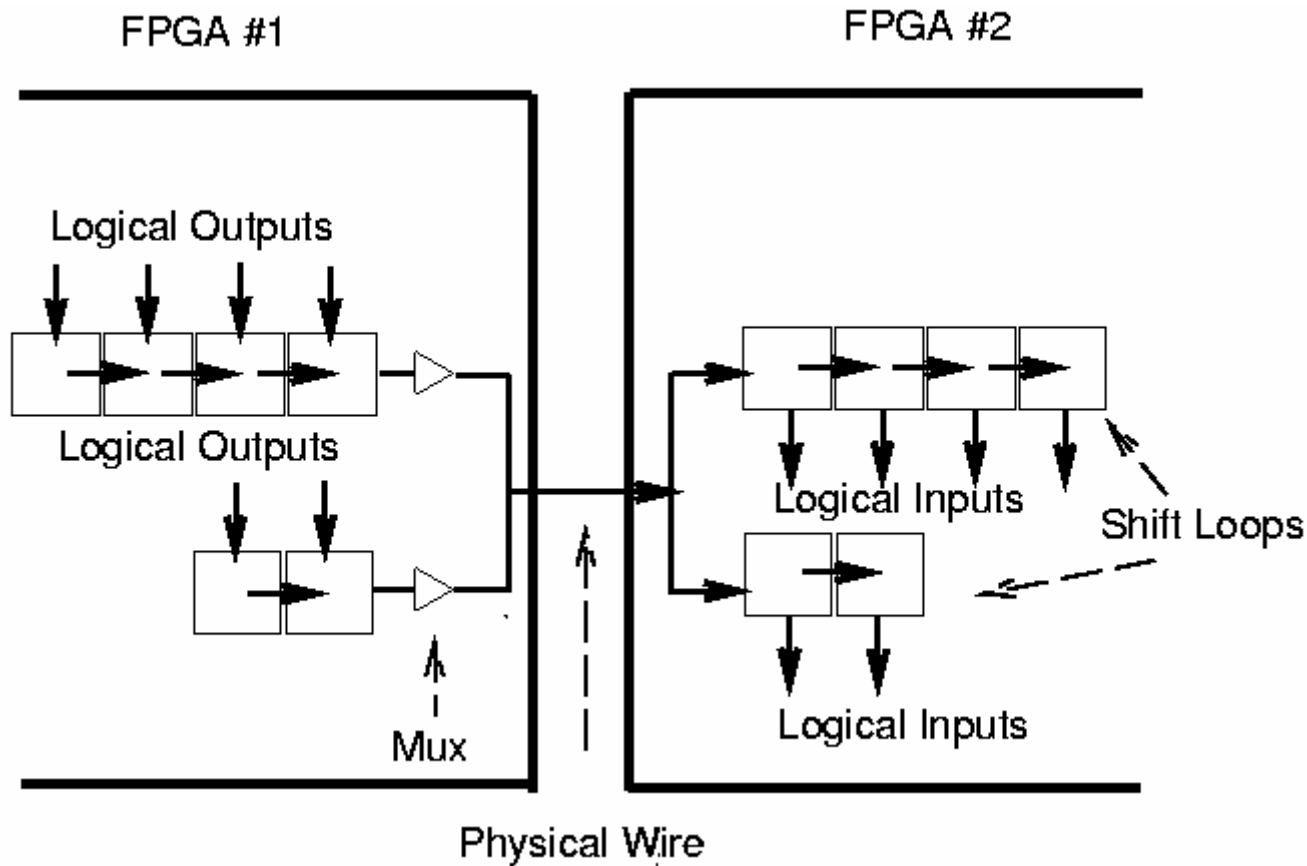
- Rent's Rule indicates that pin limited situation is getting worse.
- Frequently some logic must be left unused leading to limited utilization
- Perhaps this logic can be "reclaimed"

Partition vs FPGA Pin Count



- FPGAs don't have enough pins
- Problem may or may not get worse depending on "structured" design.

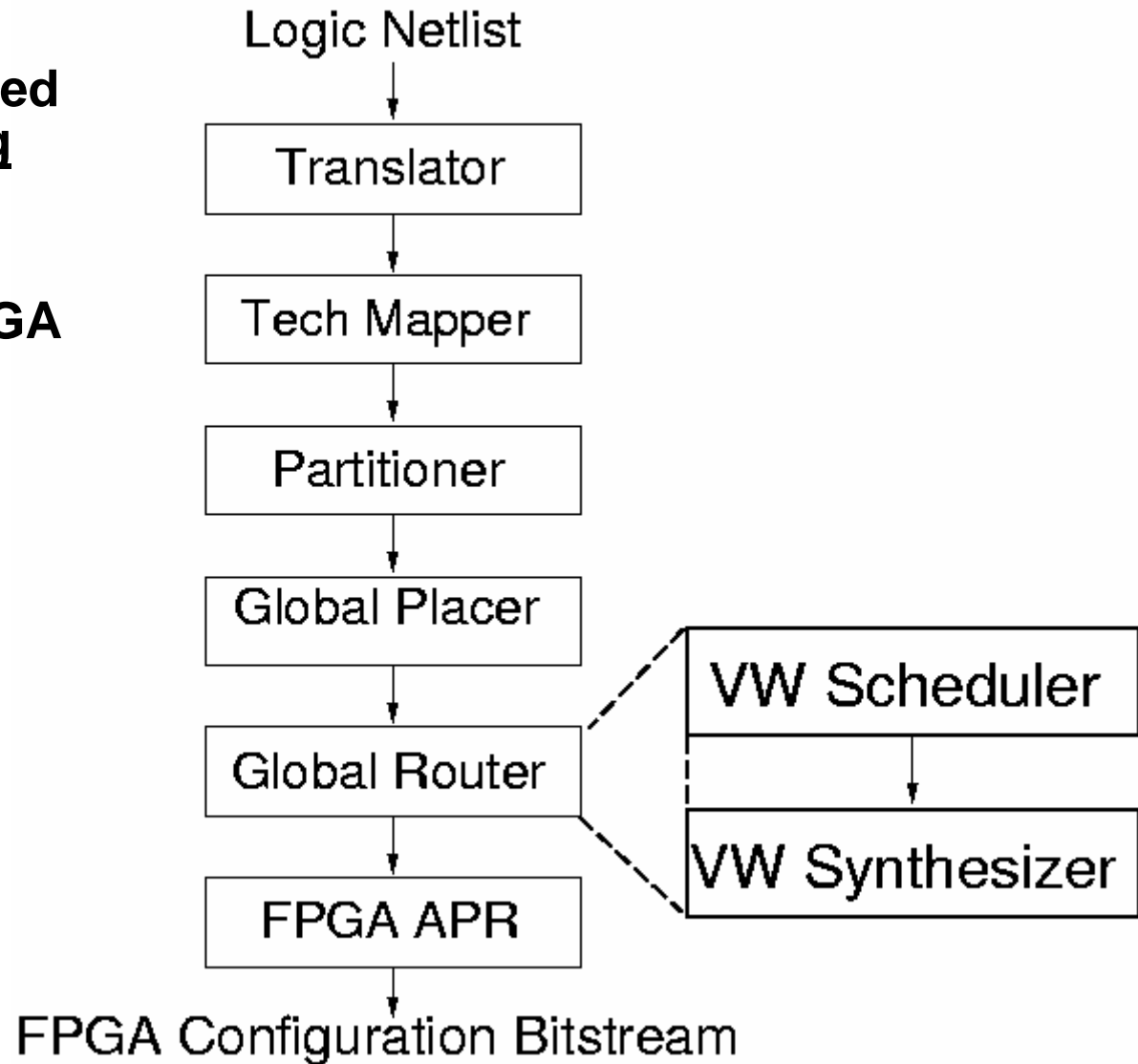
Virtual Wires



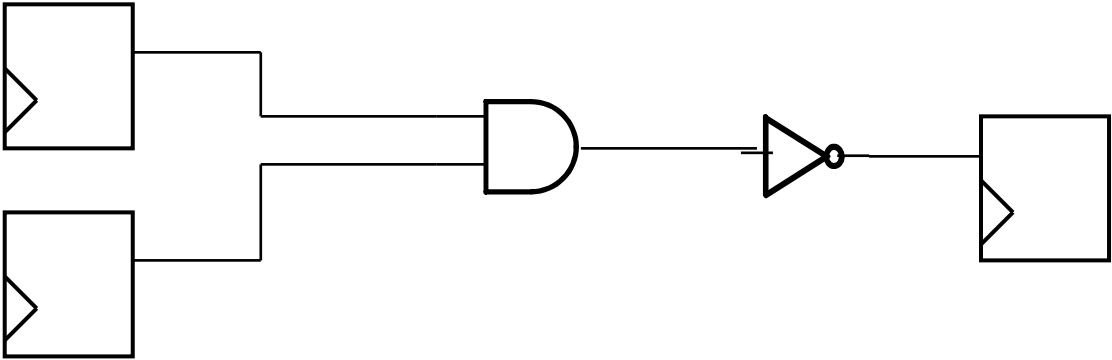
- Overcome pin limitations by multiplexing pins and signals
- Schedule when communication will take place.

Virtual Wires Software Flow

- **Global router enhanced to include scheduling and embedding.**
- **Multiplexing logic synthesized from FPGA logic.**

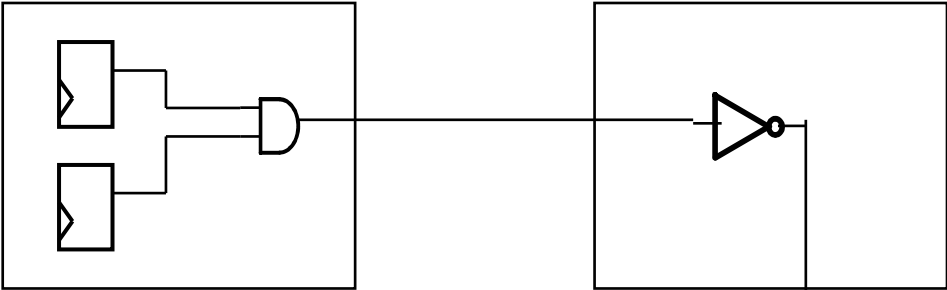


A Simple Example



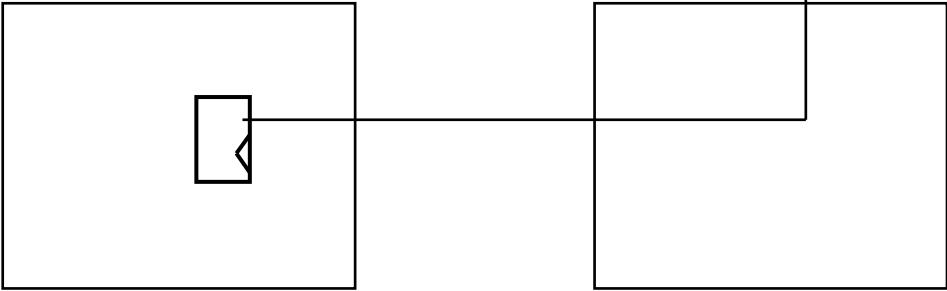
FPGA 1

FPGA 2

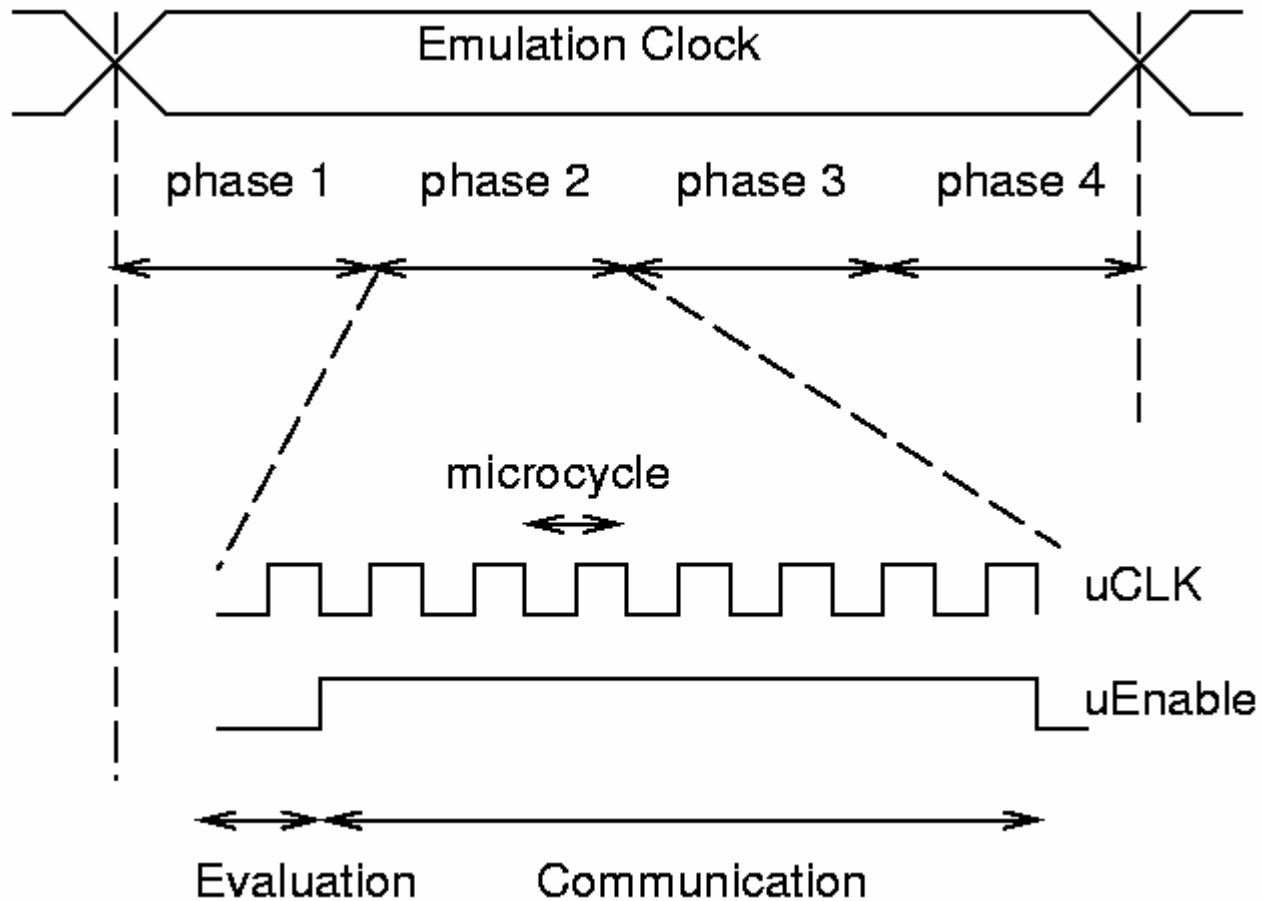


FPGA 4

FPGA 3



Clocking Strategy



- Evaluation and communication split into phases
- Longest dependency path determines number of phases
 - Overall emulation performance

Example Scheduling

- Initial phase requires one uClk for computation, one for communication.
- Second phase requires 2 communication uClks due to through hop.
- Note this example assumed needed bandwidth was available.

Routing Algorithm

- For each phase, only some internal signals are ready for routing.
- Routing resources between FPGAs may be considered channels.
- Solution: Route signals use maze route for each phase.
- If available bandwidth not present, delay signals until later phases.

Worst Case Microcycle Count

$$V \geq \max (L * D, P_c / P_f)$$

L = critical path length

D = network diameter

P_c = max circuit partition pin count

P_f = FPGA pin count

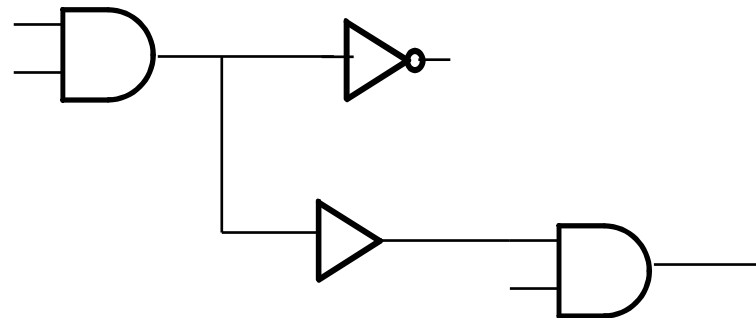
- **Most designs dominated by latency bound.**
- **If original design has been pipelined this is less of an issue**

Improved Scheduling

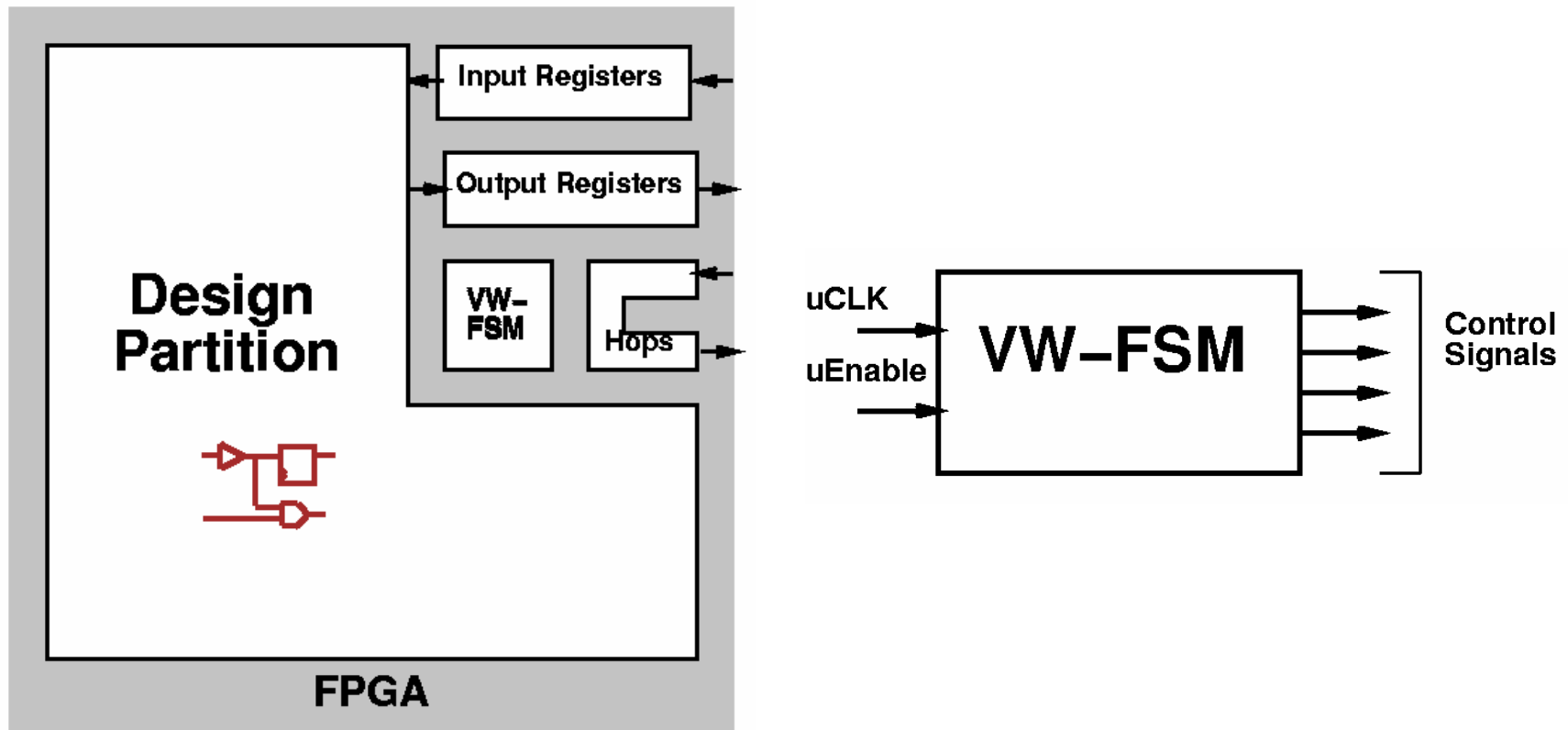
- **Overlap computation and communication.**



- **Effectively create a “data flow” of information**
- **Schedule communication to happen as soon as possible**
 - No need for phases.

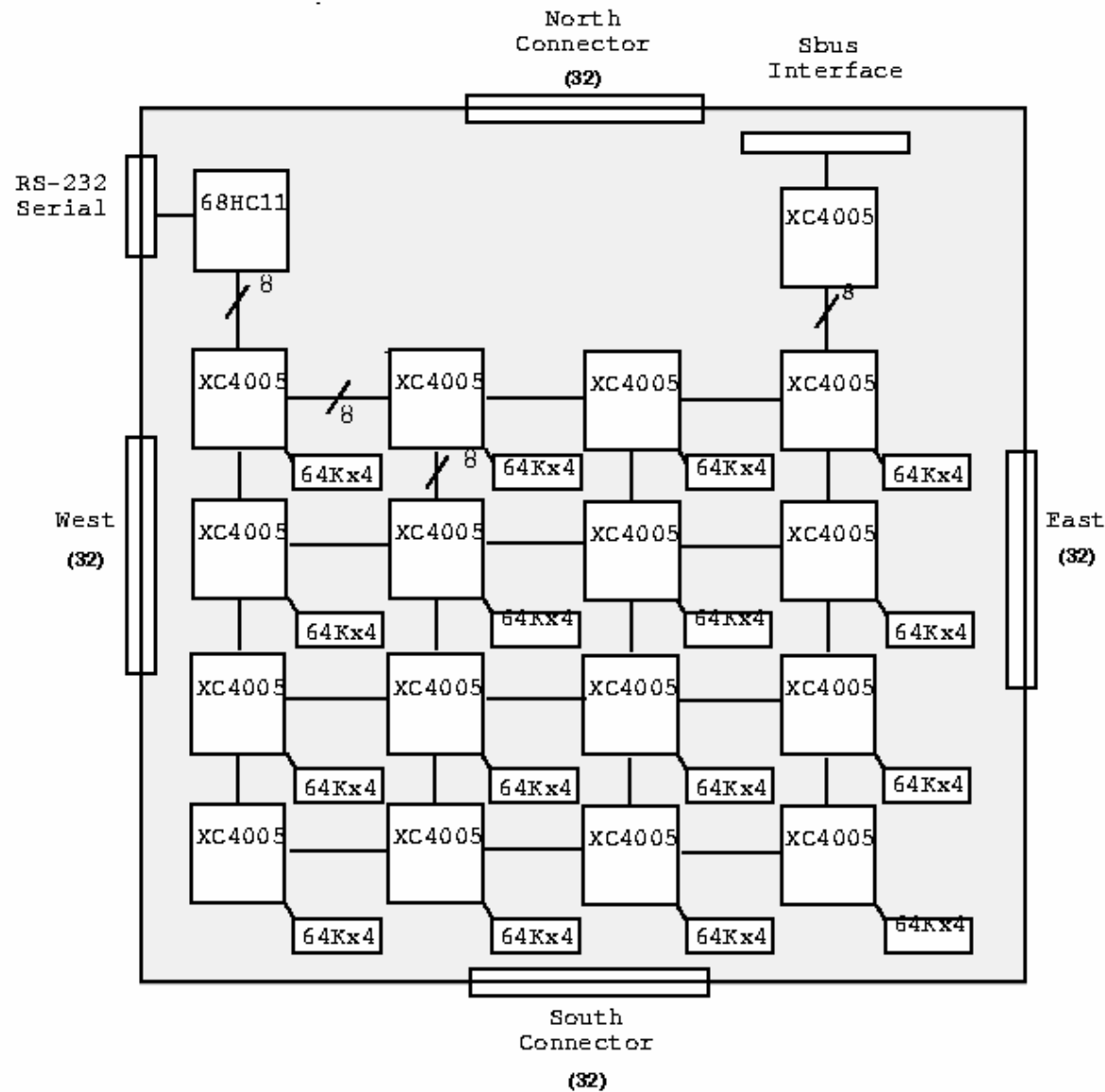


Physical Implementation



- Small finite state machine encoded and placed in each FPGA
- Current implementation is one-hot encoding.

System Implementation



- Low cost hardware
- So simple a graduate student can build it

Benchmark Designs

- **Sparcle – modified Sparc processor**
 - 17K gates
 - 4,352 bits of memory
 - Emulated in circuit.
- **CMMU – cache controller for scalable multiprocessor system**
 - 85K gates
 - Designed as gate array and optimized with SIS
- **Palindrome**
 - 14K gates
 - systolic

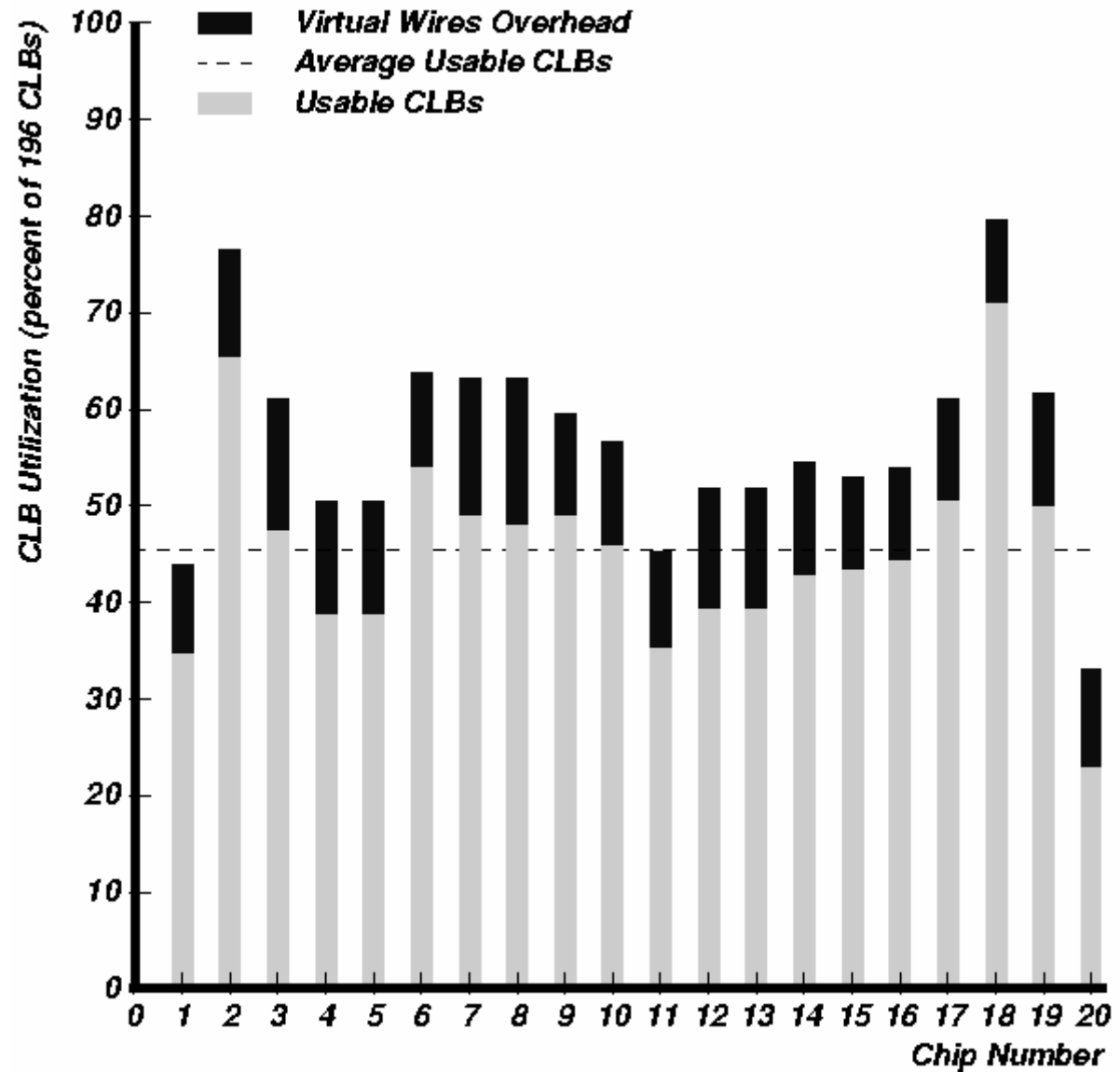
Emulation Results

Results	Sparcle Simulation	Sparcle Emulation
FPGAs	20	24
Critical Path Length, L	10 partitions	11 partitions
Average Route Length, d	2.44 FPGAs	2.34 FPGAs
Maximum Route Length, D	7 FPGAs	6 FPGAs
Avg. VW-mesh I/O	25	29
Avg. HW-cross I/O (PMF)	126 (5.0)	119 (4.1)
Est. HW-mesh I/O (PMF)	294 (11.8)	293 (10.1)
VW-mesh μ CLK Speed	12.0 MHz	18.0 MHz
VW-mesh phases \times cycles	10 \times 10	11 \times 9
VW-cross phases \times cycles	10 \times 2	11 \times 2
VW-mesh Emulation Speed	0.12 MHz	0.18 MHz
Est. VW-cross Emulation Speed	0.60 MHz	0.82 MHz
Est. HW-mesh Speed (ideal)	0.94 MHz	1.01 MHz
Est. HW-cross Speed (ideal)	1.30 MHz	1.43 MHz

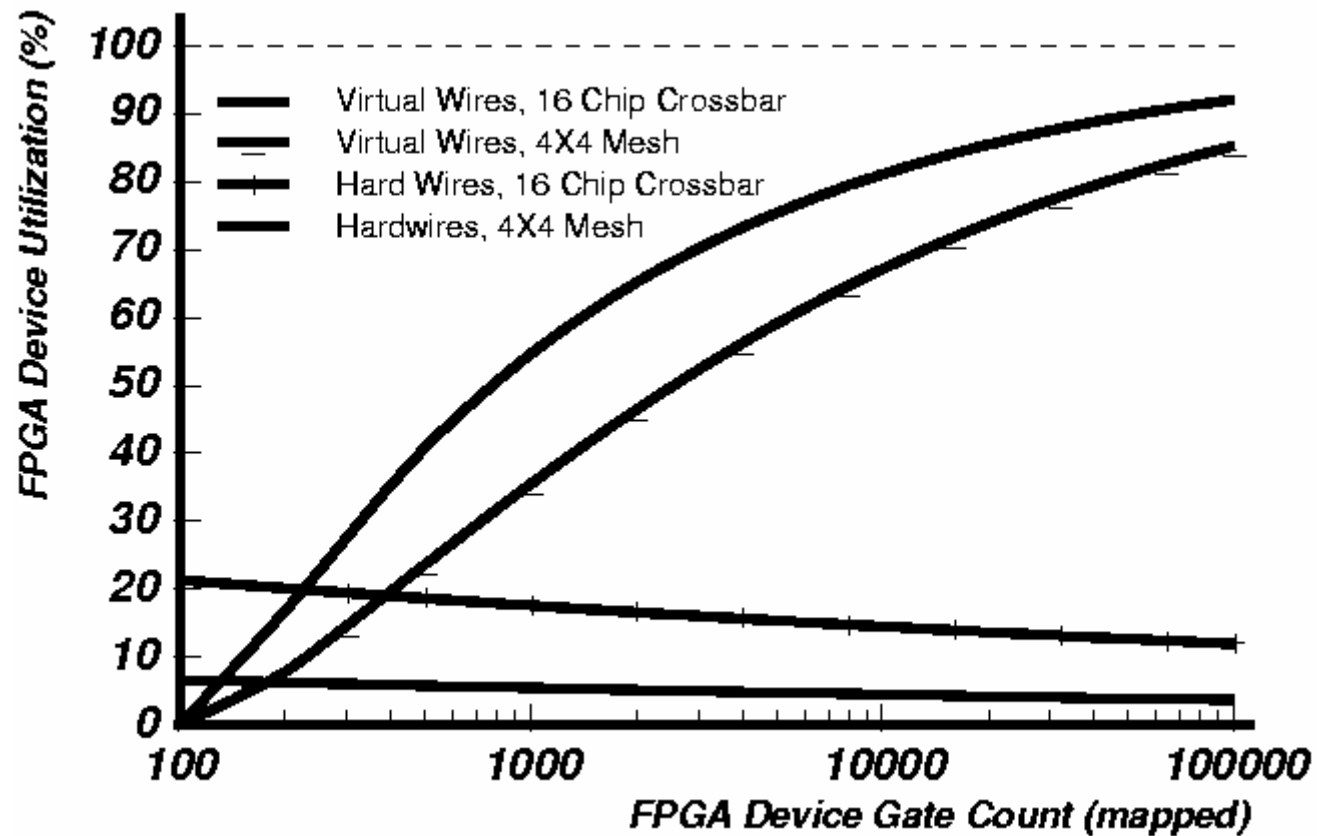
- At least 31 FPGAs needed for HW full connectivity (>100 for torus)
- Some degradation in overall system performance.

Device Utilization

- Approximately 45% of CLBs used for design logic.
- ~10% virtual wires overhead



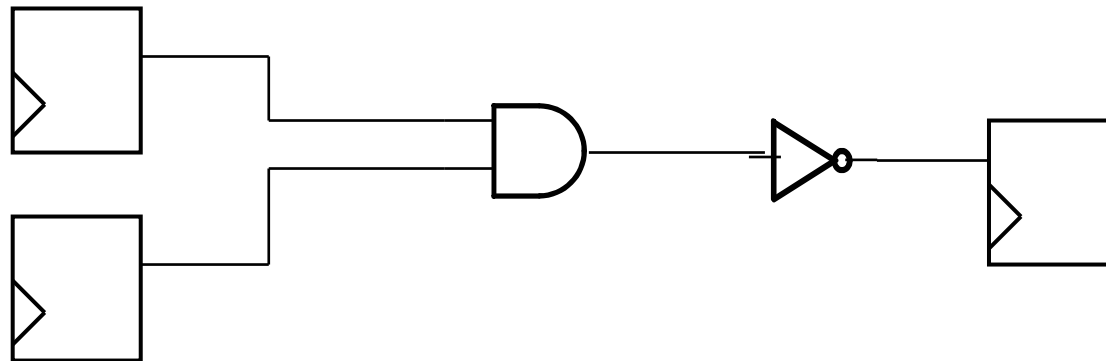
Utilizations



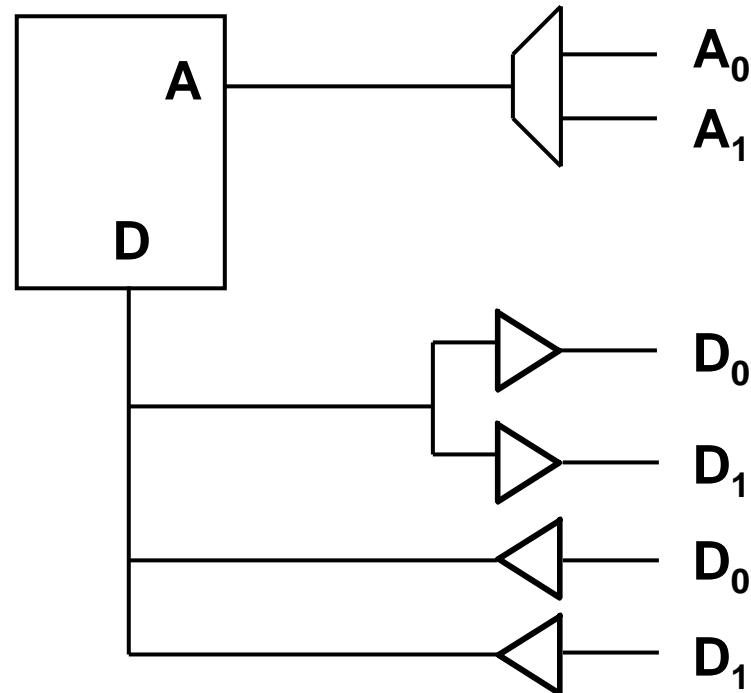
- As devices scale projected utilization increases
- Hardwired approach doesn't scale
- Equation ->

Future Directions

- Incremental compilation
 - FPGAs take a long time to compile
 - Desirable to isolate changes to a small number of partitions
 - Scheduling simplifies issue by allowing additional communication cycles
 - **Rest of circuit unchanged!**
- Perhaps isolate at the macroblock stage
- Impact on topology.



Virtualized Memory



- Multiplex a single-ported memory over time to create a multi-port memory
- Allows use of low-cost memories in system development
- Also multiplexed logic analyzer interface.

Summary

- **Virtual wires overcome pin limitations by intelligently multiplexing I/O signals**
- **Key CAD step is scheduling. Simplifies routing and partitioning.**
- **Latest push is towards incremental compilation**
- **Commercialized by Icos Systems (now Mentor Graphics)**