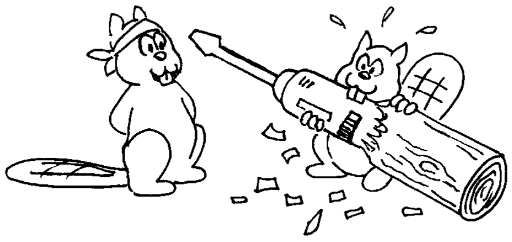
DPGA
Utilization and Application

André DeHon
<**andre@ai.mit.edu**>

Reinventing Computing



Acknowledgments: This research is supported by the Advanced Research Projects Agency of the Department of Defense under Rome Labs contract number F30602-94-C-0252.

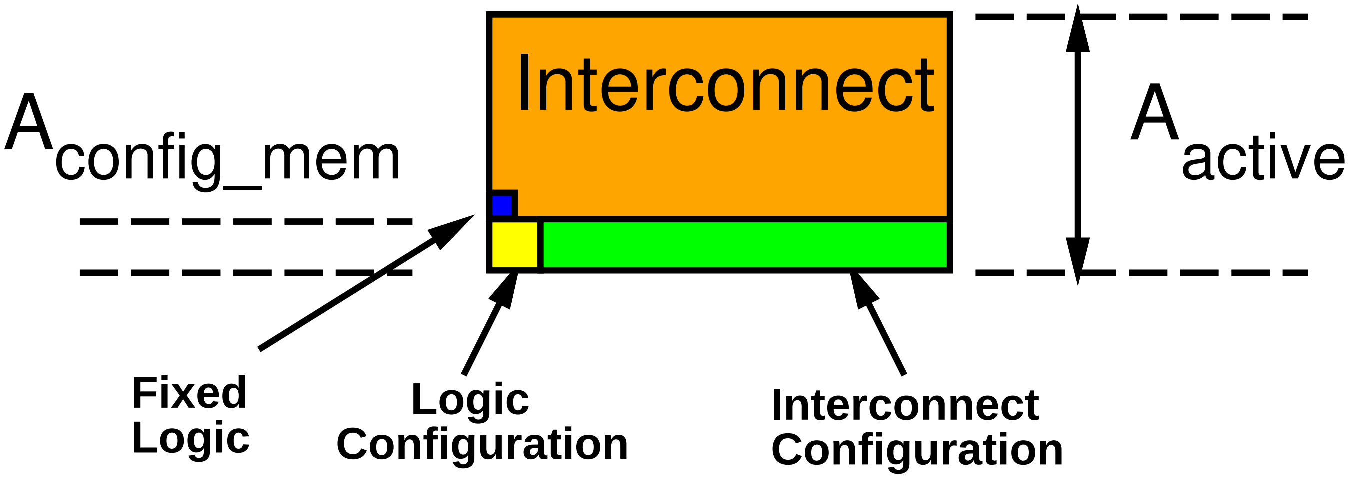
MIT
AI
Lab

Outline

- Key Idea: increase functionality through reuse
- Multicontext Devices
- Application: Temporal Pipelining
- Device Capacity and Utilization
- Application: FSMs
- Bottlenecks, Technology, and Application Trends
- Summary

DPGA – Key Idea

Key Idea: efficiently reuse limited resources to perform different functions



$$A_{active} >> A_{config\_mem}$$

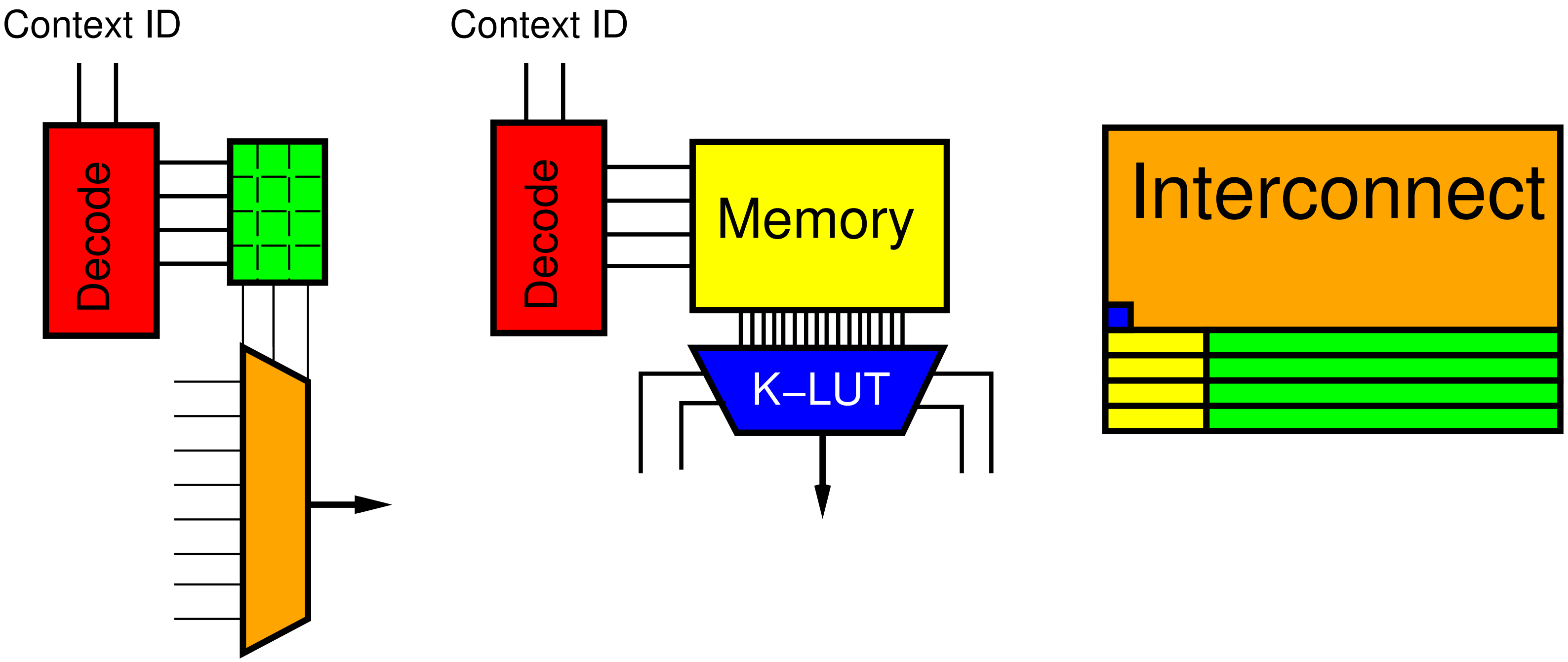
5:1 conventional devices
10:1 easily achievable

MIT AI Lab

Multicontext FPGAs

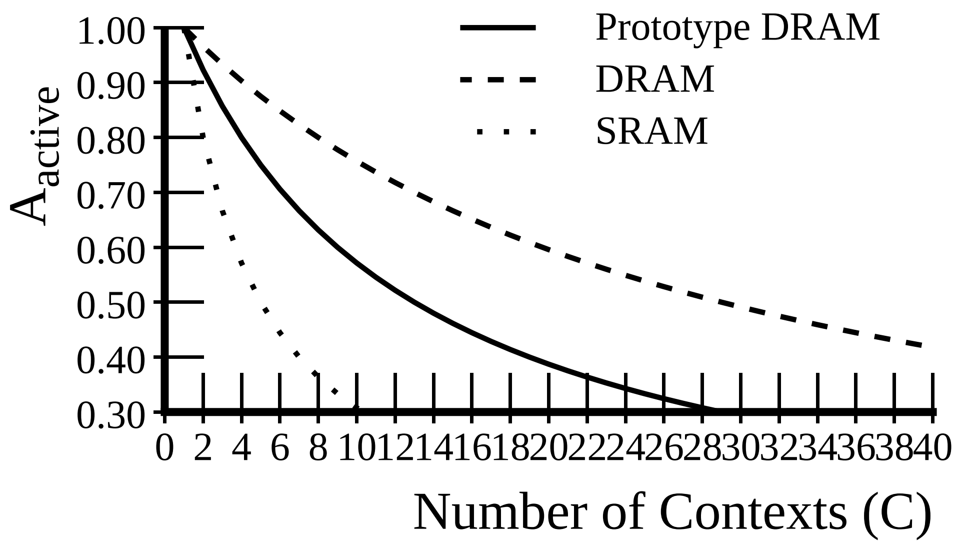
MIT
AI
Lab

DPGAs



Sacrifice active capacity to memory allowing device to support a variety of functions.

DPGA – Active Area Reduction



$$A_{active} = \frac{1}{1 + (c - 1)C_{mem}}$$

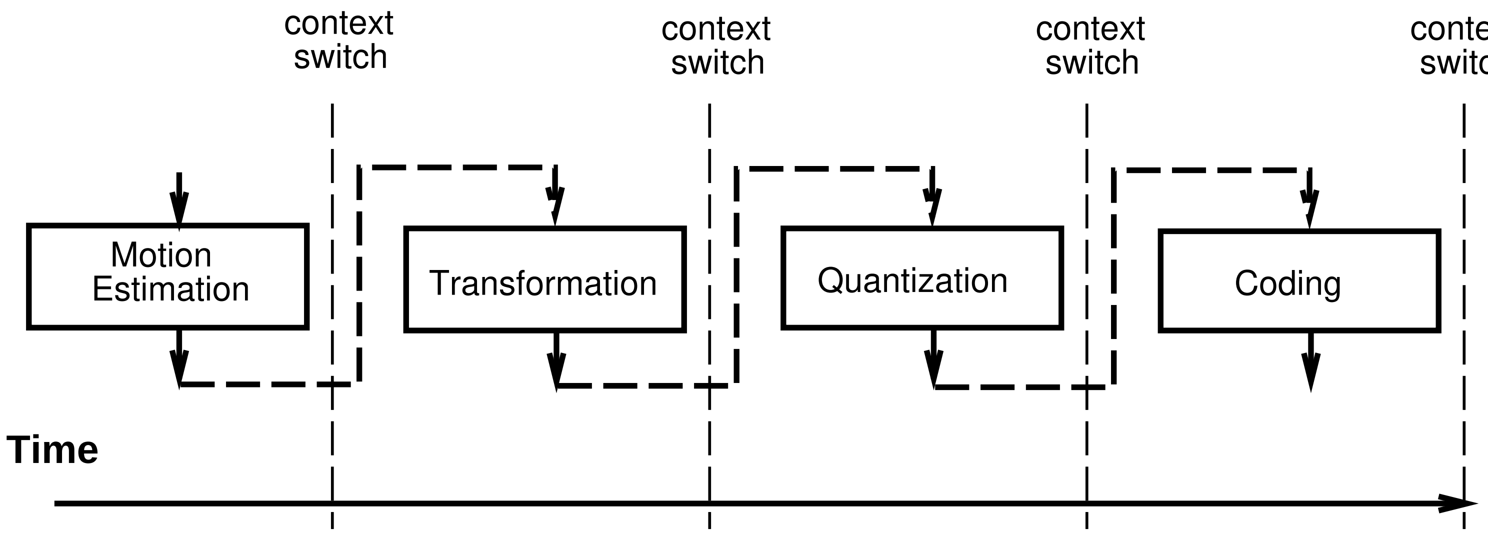
MIT AI Lab

Application

Temporal Pipelining

Fold systolic pipe in time

E.g.

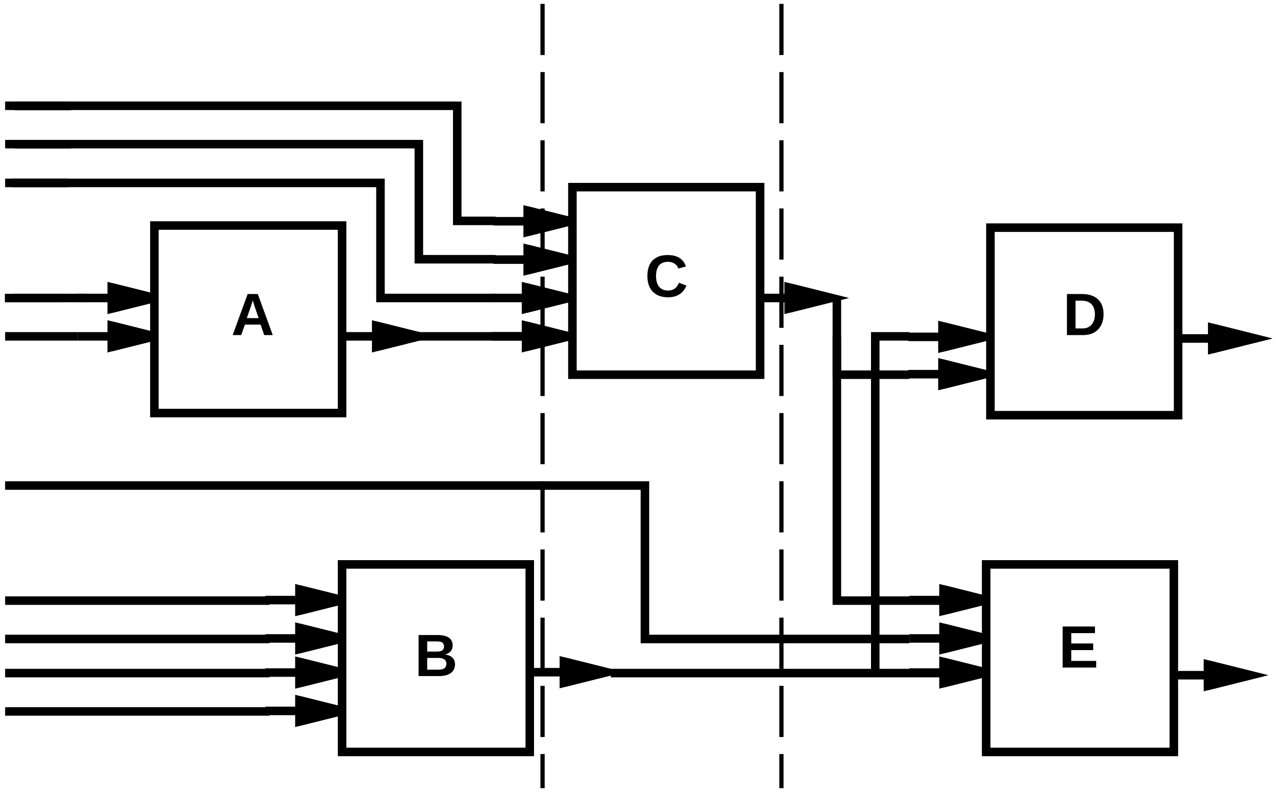


MIT AI Lab

Temporal Pipelining – Levelized Logic

Automatically temporal pipelining of existing netlists

- temporal partitioning (Bhatt)
- Spacetime circuitry (Margolus)



MIT AI Lab

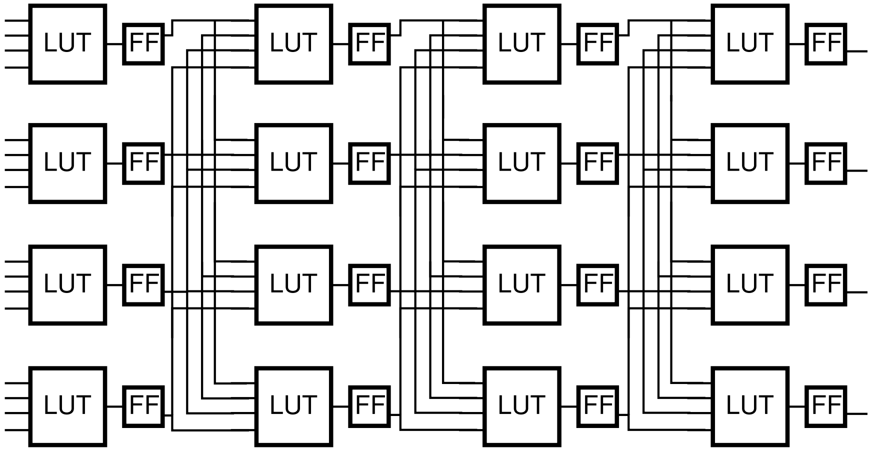
Capacity and Utilization

Capacity

Capacity: Computational throughput

$$C_{peak} = \frac{N_{resource}}{t_{min\_cycle}}$$

$$C_{mcpeak} = C_{peak} \times A_{active}$$



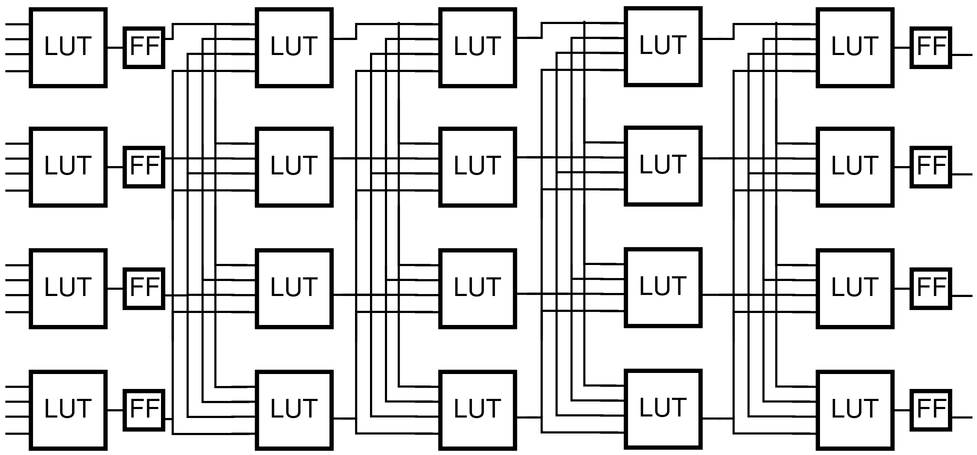
MIT AI Lab

Utilization

Utilization: Useful/Yielded computational throughput

$$U = \frac{N_{resource\_used}}{t_{clk}}$$

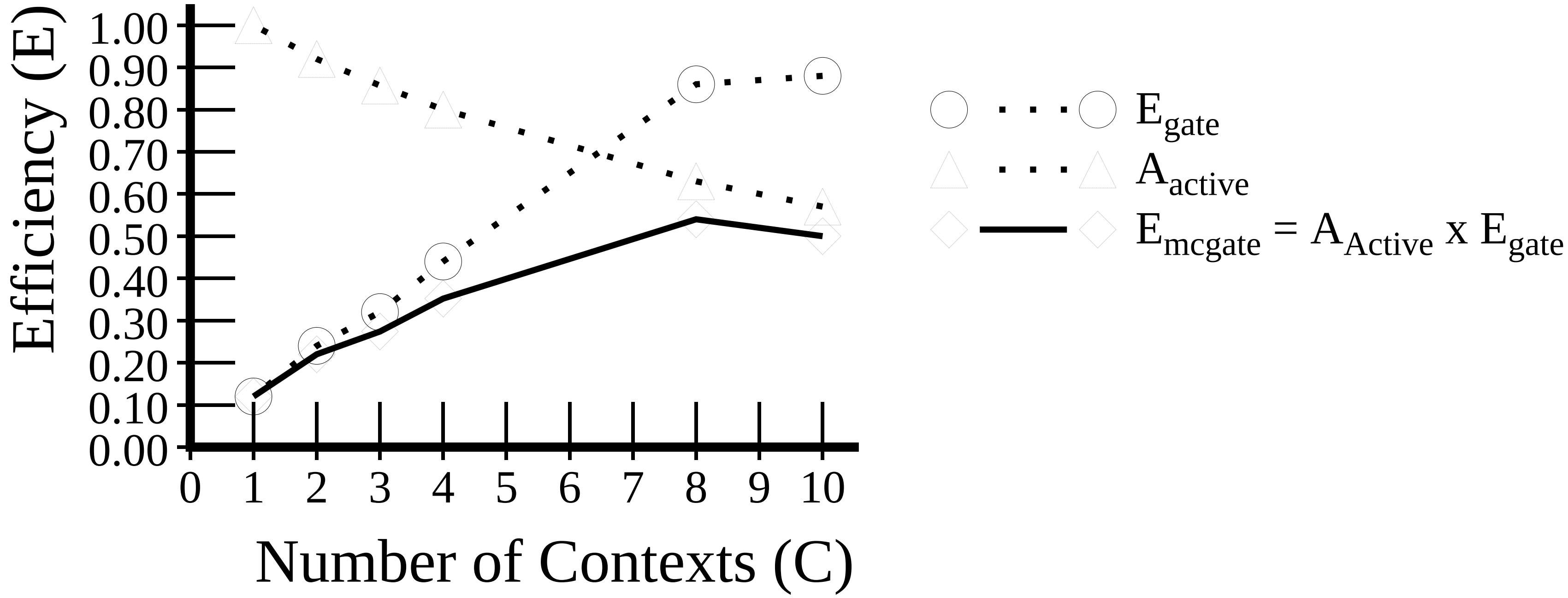
$$E = \frac{U}{C_{peak}}$$

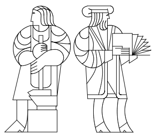


Levelized Logic Example

DES (MCNC Benchmark)

Critical Path – 8 gate delays – $E_{single} = 0.13$



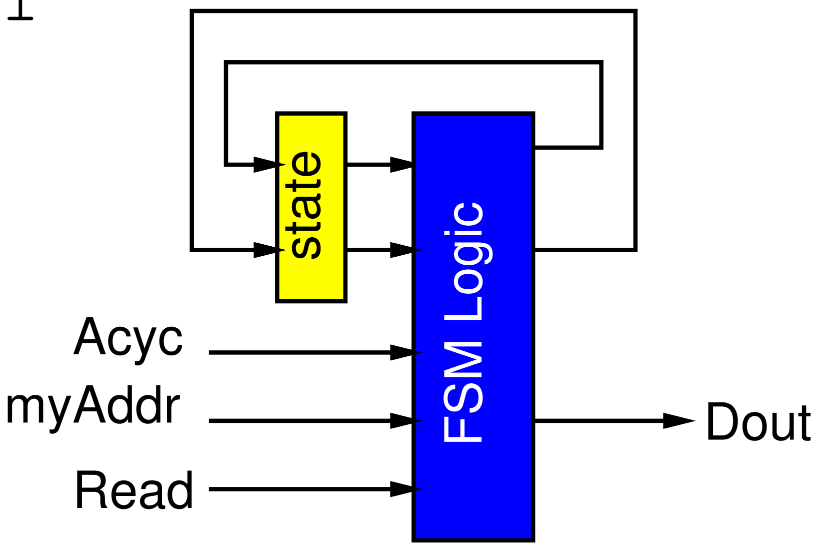


MIT AI Lab

Application: FSMs

Finite State Machines

```
Idle (00):
          if (Acyc & myAddr & Read)
             goto Wait1
          else
             goto Idle
Wait1 (01): goto Data
Data (10): Assert Dout
           goto Wait2
Wait2 (11): goto Idle
```



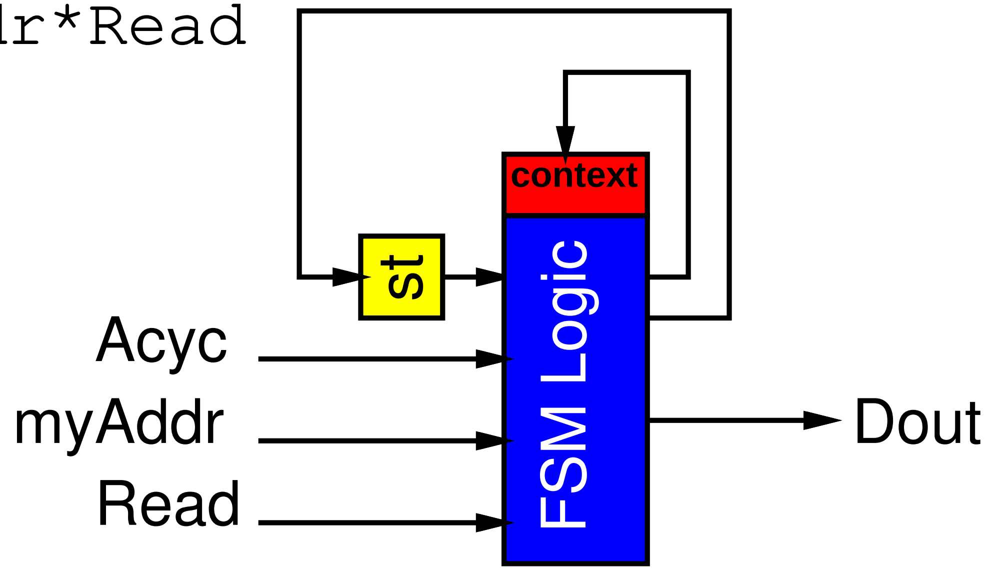
```
Dout = S1*/S0
NS0 =  /S1*/S0*Acyc*myAddr*Read + S1*/S0
NS1 = /S1*S0 + S1*/S0
```

Finite State Machines

Reduce Logic – Factoring by States

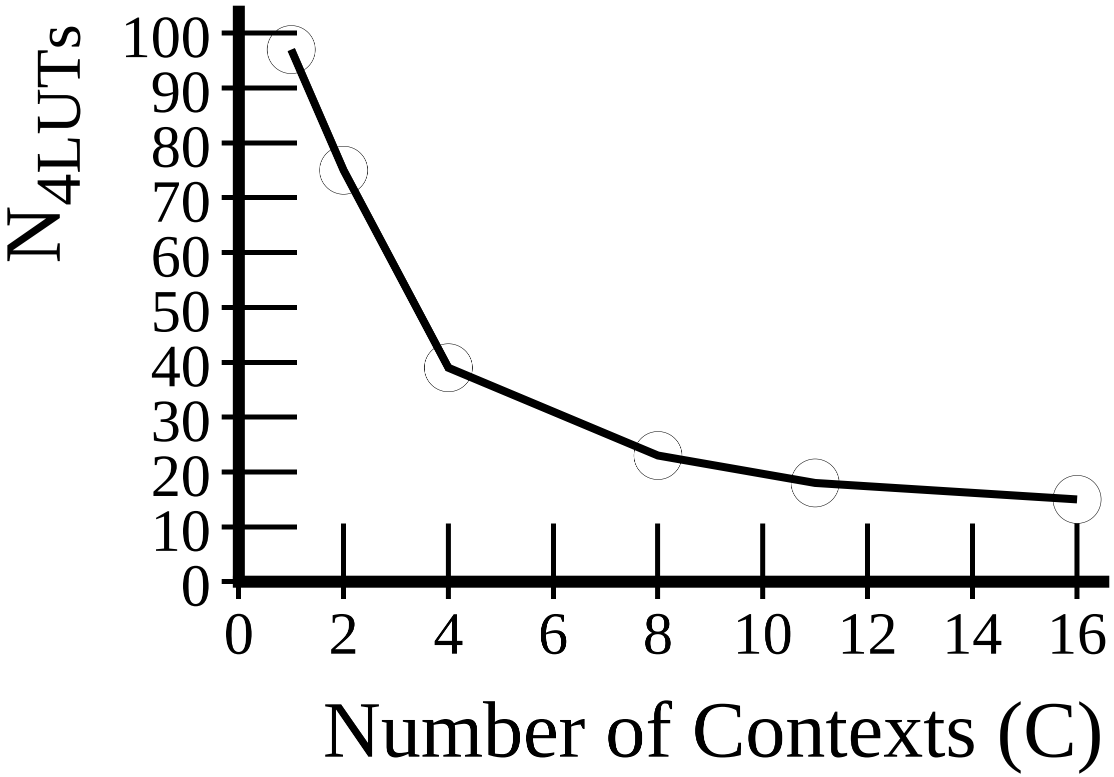
```
Context 0 (S1=0)
Dout = 0
NS0 = /S0*Acyc*myAddr*Read
NS1 = S0

Context 1 (S1=1)
Dout = 0
NS0 = /S0
NS1 = /S0
```



Finite State Machines

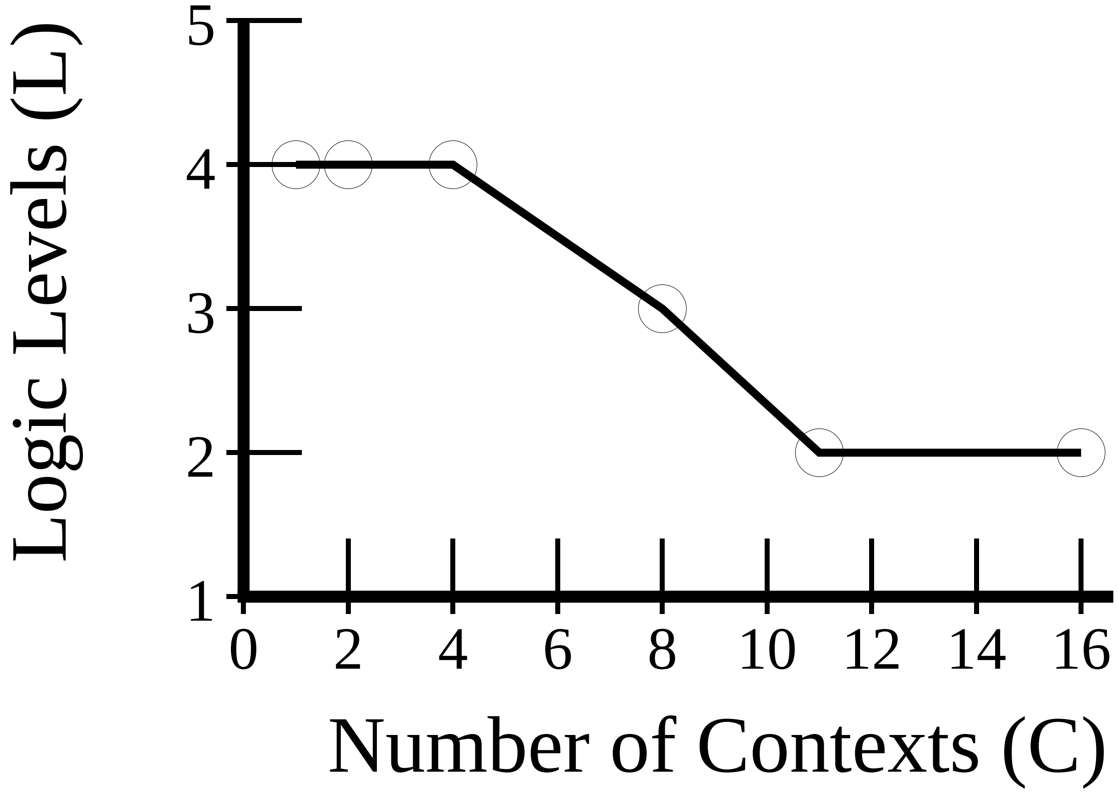
CSE (MCNC FSM Benchmark) – 16 States



MIT
AI
Lab

Finite State Machines

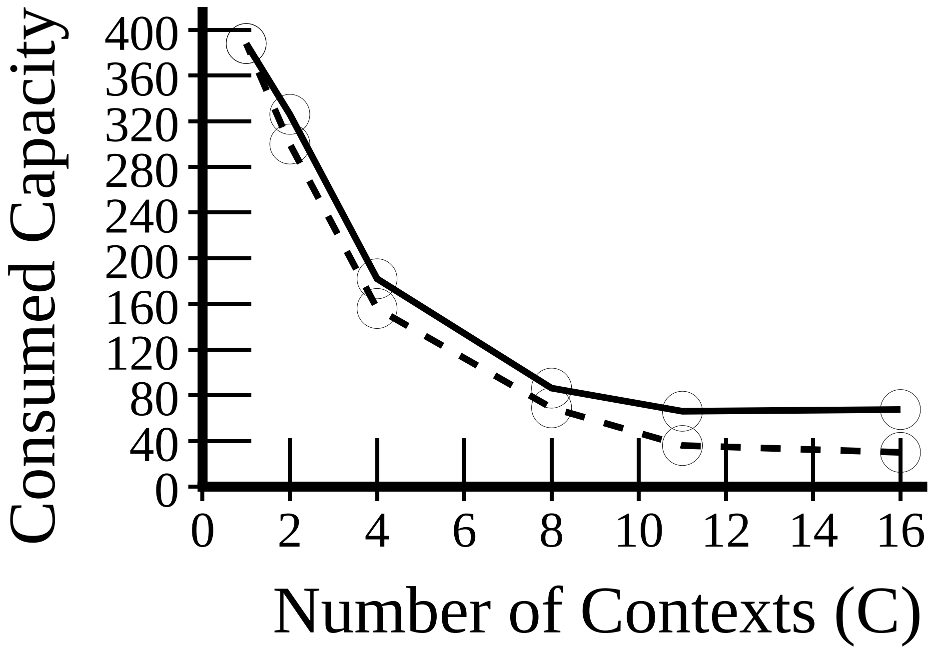
CSE (MCNC FSM Benchmark) – 16 States



MIT
AI
Lab

Finite State Machines

CSE (MCNC FSM Benchmark) – 16 States



Consumed Capacity = $N_{4LUT} \times L$

Normalized Consumed Capacity = $(N_{4LUT} \times L)/A_{active}$

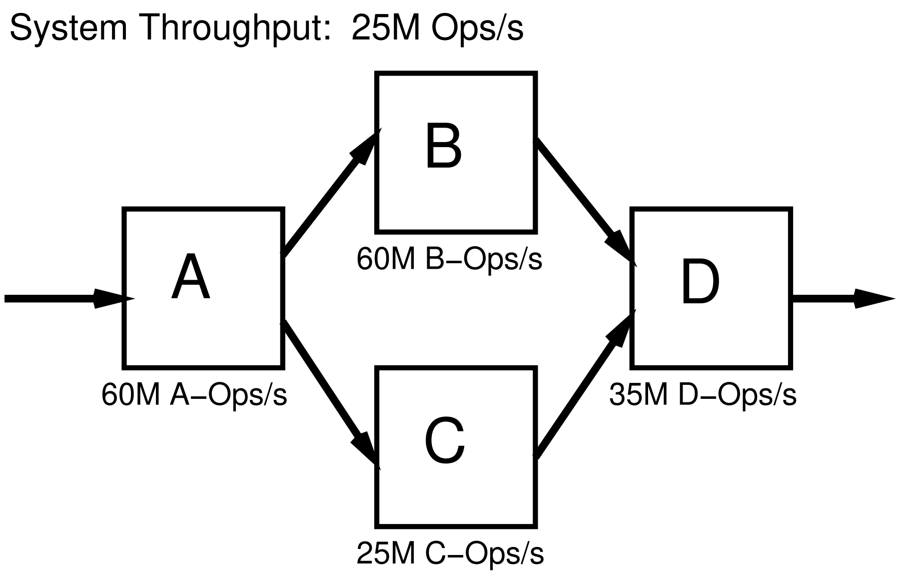
MIT AI Lab

Bottlenecks

Bottlenecks and Capacity

Why don't we fully, spatially, pipeline everything?

- Relative processing speeds
- Fixed functional requirements
- I/O Bandwidth bottleneck
- Latency limited designs



MIT AI Lab

Summary (1 of 2)

- Device Capacity: two components
  - spatial (how many?)
  - temporal (how quickly can I reuse them?)
- Conventional FPGAs
  - temporal capacity $\rightarrow$ more of the same
  - spatial capacity $\rightarrow$ diverse functions
- Multicontext FPGAs (*e.g.* DPGA)
  - temporal capacity $\rightarrow$ diverse functions

Summary (2 of 2)

- Applications
  - typically exhibit functionality limitations
  - effect increases as device speeds increase
- Styles for exploiting
  - Multifunction components
  - Utility functions
  - Temporal pipelining
    - levelized logic of circuits – 3-4$\times$ efficiency increase
  - Temporally varying logic

Additional Information

- Full Technical Report in HTML
  - <http://www.ai.mit.edu/projects/transit/tn129/tn129.html>
- Full Technical Report in PostScript
  - <ftp://transit.ai.mit.edu/transit-notes/tn129.ps.Z>
- FPGA'96 paper in PostScript
  - <ftp://transit.ai.mit.edu/paper/dpga-use-fpga96.ps.Z>
- PostScript for these slides
  - <ftp://transit.ai.mit.edu/slides/dpga-use-fpga96.ps.Z>
- Reinventing Computing Home Page
  - <http://www.ai.mit.edu/projects/transit/rc_home_page.html>