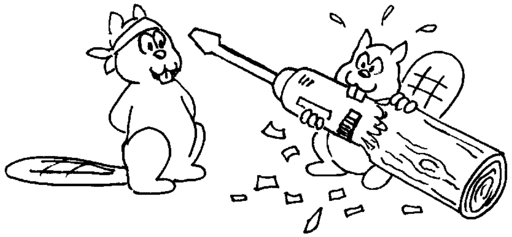
Dynamically Programmable Gate Arrays: A Step Toward Increased Computational Density

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.

Goal

Goal for a *general-purpose* computing device:

- implement desired functionality
- perform at required throughput or latency
- require minimal cost (*e.g.* silicon area)

How big does my computation need to be?

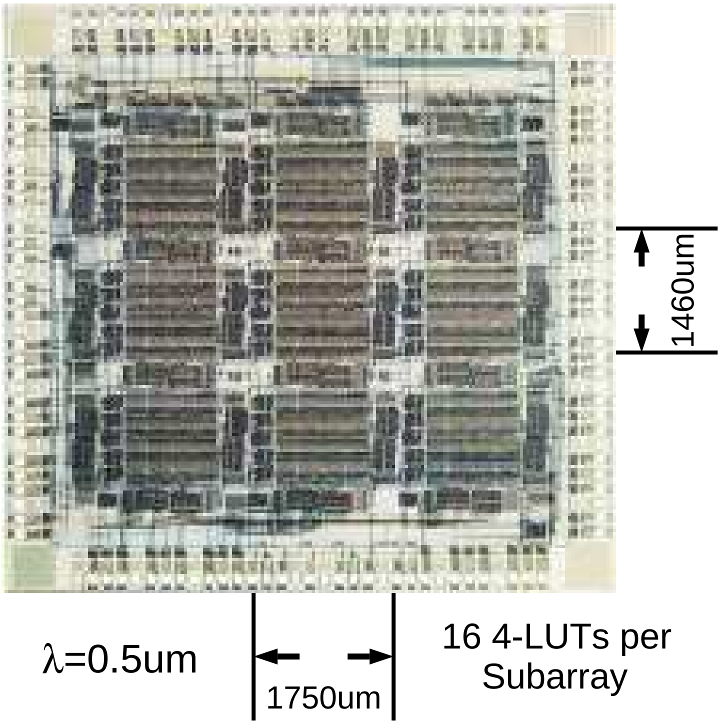
Outline

- How big is a computation?
- DPGA – multicontext FPGAs
- Example: ASCII Hex→Binary conversion
- iDPGA – (adding input registers)
- Area versus Throughput
- Summary

How Big?

How Big Is My Computation?

Area (4-LUT) $\approx$ 600-800Kλ^2



(*e.g.* 640Kλ^2)

$$A_{compute} = N_{LUT} \times A_{LUT}$$

MIT AI Lab

How Big Is My Computation?

...but, computational descriptions are more compact.

Configuration bits/4-LUT:

$$50 \rightarrow 200$$

Area/Configuration bit:

$$\text{DRAM} \approx 300\lambda^2$$
$$\text{SRAM} \approx 1200\lambda^2$$

Area/4-LUT Configuration:

$$200 \text{ bits @ } 1200\lambda^2 \Rightarrow 240\text{K}\lambda^2$$
$$50 \text{ bits @ } 300\lambda^2 \Rightarrow 15\text{K}\lambda^2$$

How Big Is My Computation?

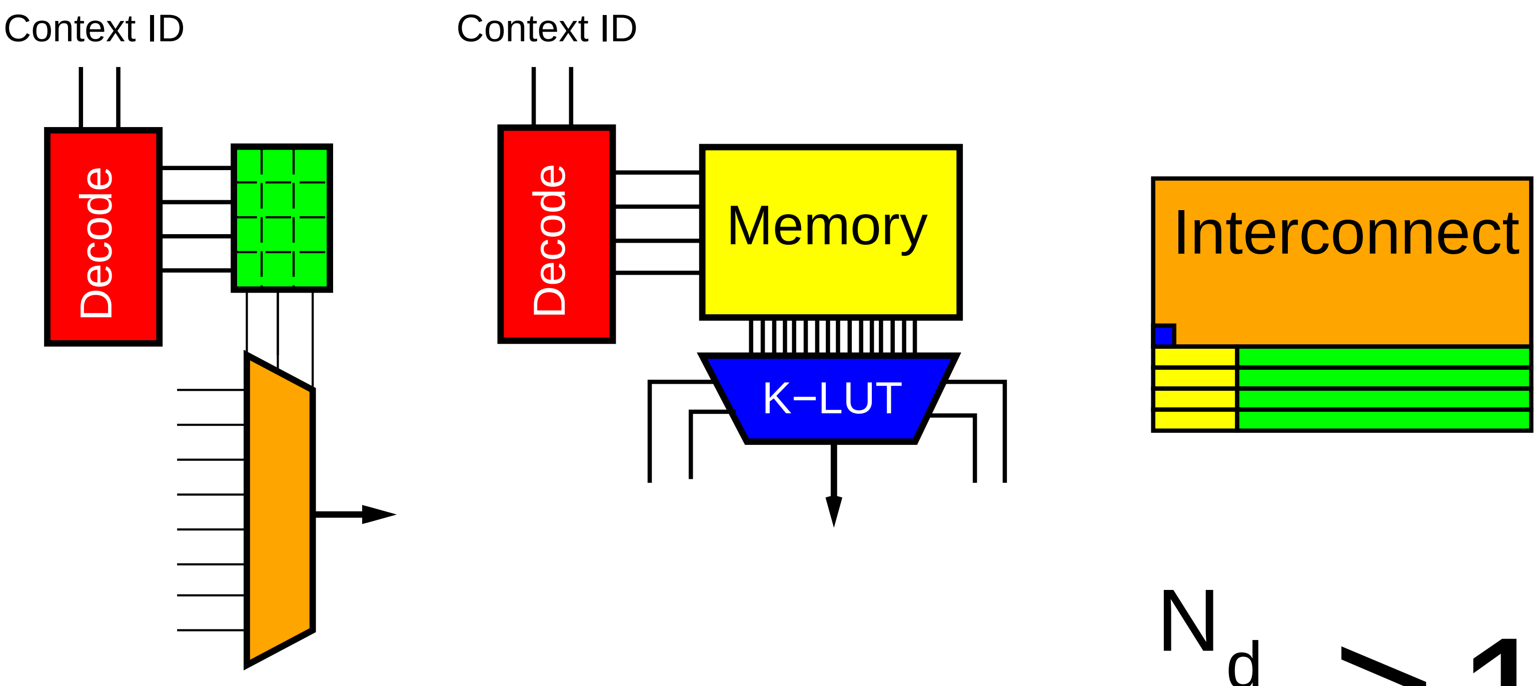
N_d – number of design LUTs (dicated by design)

N_a – number of physical/active LUTs (dictated by performance target)

$$A_{compute} = N_a \times A_{LUT} + N_d \times A_{config}$$

DPGA

Dynamically Programmable Gate Arrays



$$\frac{N_d}{N_a} > 1$$

Prototype:

$$A_{LUT} \approx 560\mathrm{K}\lambda^2$$
$$A_{config} \approx 20\mathrm{K}\lambda^2$$

MIT AI Lab

Example

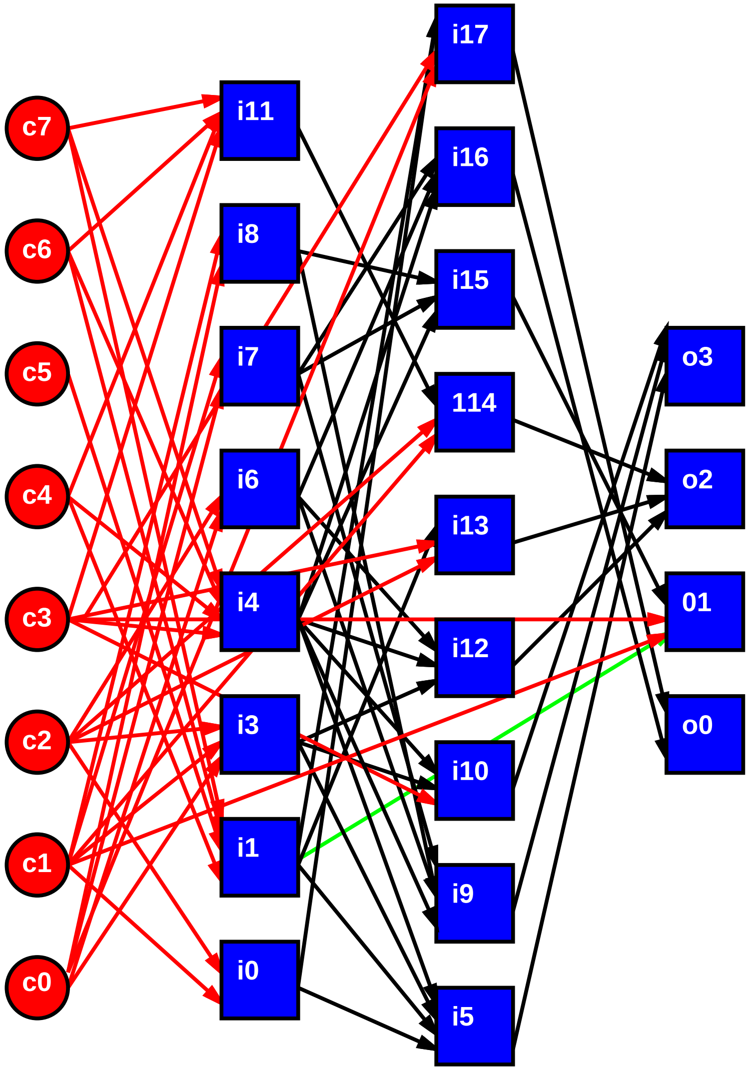
ASCII HEX→Binary Conversion

e.g. 'A' $\equiv$ 0x41 $\Rightarrow$ 0x0A (0d10)

```
if (c >= 0x30 && c <= 0x39)
    res = c-0x30;
else if (c >= 0x40 && c <= 0x46)
    res = c - 0x40 + 10;
else if (c >= 0x60 && c <= 0x66)
    res = c - 0x60 + 10;
else
    res = 0;
```

ASCII HEX→Binary Conversion

INORDER = C<7> C<6> C<5> C<4> C<3> C<2> C<1> C<0> ;
OUTORDER = O<3> O<2> O<1> O<0> ;
# stage 1 – 8 LUTs (C<3:0> pass through)
i0 = !C<1> * !C<2> ;
i1 = C<4> * C<5> * !C<6> * !C<7> ;
i3 = C<0> * C<1> * !C<2> ;
i4 = !C<3> * !C<4> * C<6> * !C<7> ;
i6 = !C<0> * C<2> ;
i7 = !C<0> * C<1> ;
i8 = C<0> * !C<1> ;
i11 = !C<7> * C<6> * !C<4> * !C<3> ;
# stage 2 – 9 LUTs (i1,C<3>,C<1> pass through)
i5 = i0 * i1 + i3 * i4 ;
i9 = i6 * i4 + i7 * i4 + i8 * i4 ;
i10 = C<3> + i3 * i4 ;
i12 = i3 * i4 + i6 * i4;
i13 = i1 * !C<3> * C<2> ;
i14 = C<2> * !C<1> * i11 ;
i15 = i8 * i4 + i7 * i4 ;
i16 = i7 * i4 + i6 * i4 ;
i17 = i1 * !C<3> * C<0> + C<0> * i0 * i1 ;
# stage 3 – 4 LUTs
O<3> = (i10+i9)*(i5+i9);
O<2> = i12 + i13 + i14 ;
O<1> = i1 * !C<3> * C<1> + i15 ;
O<0> = i16 + i17 ;



MIT AI Lab

ASCII HEX→Binary Conversion

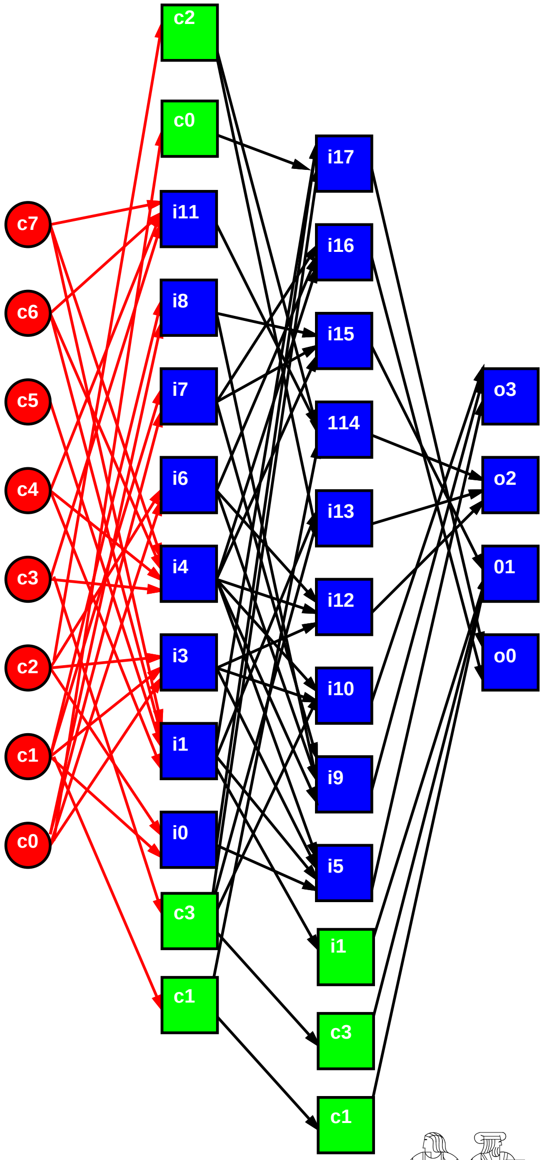
7 Additional LUTs for retiming

$$A_{pipe} = \Big(28 \cdot 560\text{K}\lambda^2 \\ +28 \cdot 20\text{K}\lambda^2\Big) \\ 16.2\text{M}\lambda^2$$

at ≈7 ns/LUT

140M chars/sec

Spatial Pipelining (FPGA)



ASCII HEX→Binary Conversion

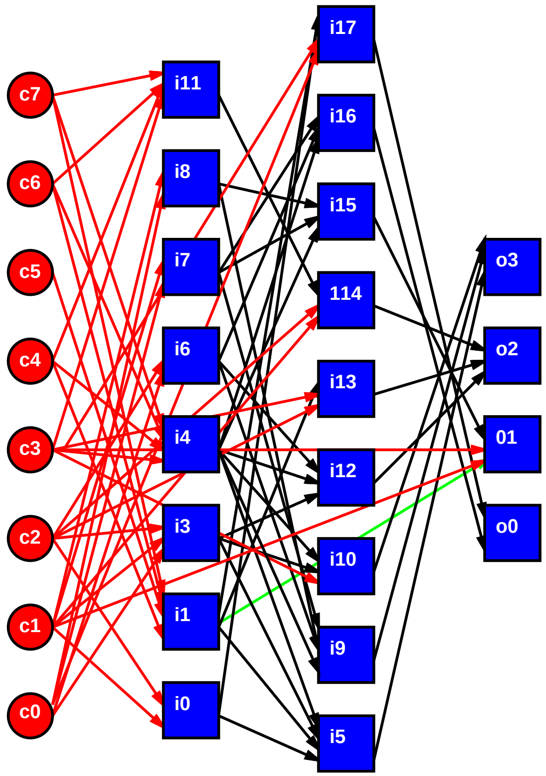
21 LUTs total

$$A_{nopipe} = \left(21 \cdot 560\mathrm{K}\lambda^2 + 21 \cdot 20\mathrm{K}\lambda^2\right)$$
$$12.2\mathrm{M}\lambda^2$$

at ≈7 ns/LUT

48M chars/sec

No Pipelining (FPGA)



MIT AI Lab

ASCII HEX→Binary Conversion

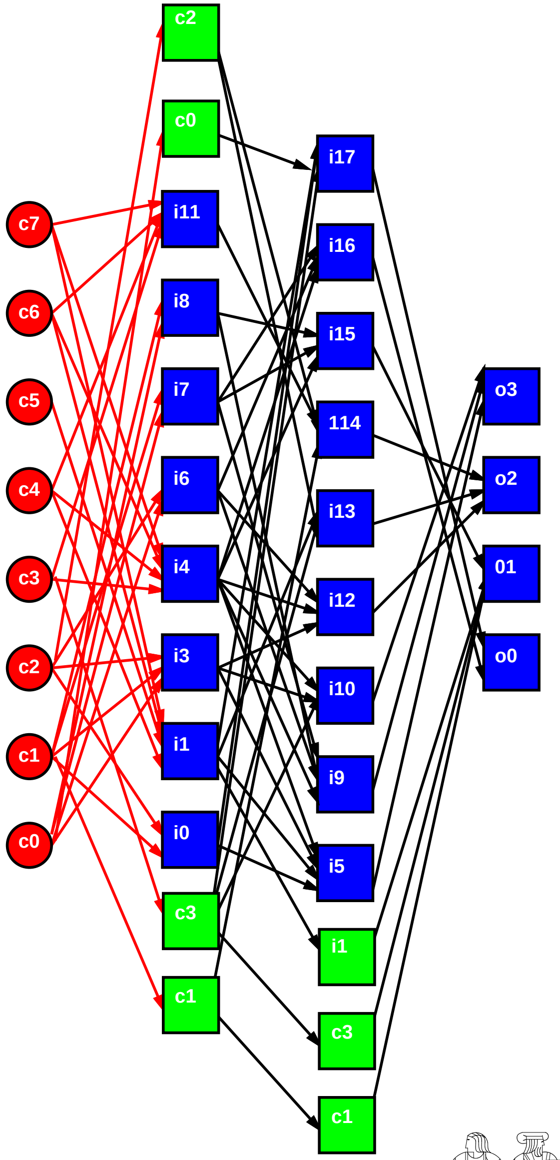
12 LUTs in largest stage

$$A_{time-pipe} = \left(12 \cdot 560\text{K}\lambda^2 + 36 \cdot 20\text{K}\lambda^2\right)$$
$$7.4\text{M}\lambda^2$$

at ≈7-9.5 ns/LUT

35-48M chars/sec

Temporal Pipelining (DPGA)



MIT AI Lab

ASCII HEX→Binary Conversion

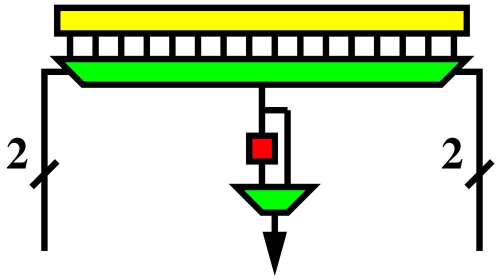
Can we temporally pipeline further?

For this example and architecture, **no**

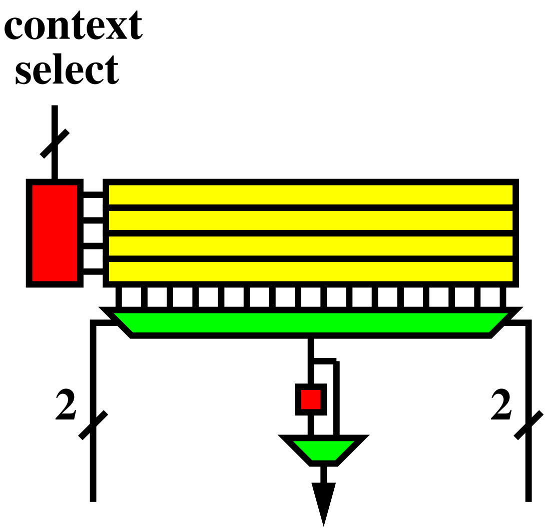
Signal retiming requirements prevent sequentialization.

N.B. Can share logic in time sliced manner.

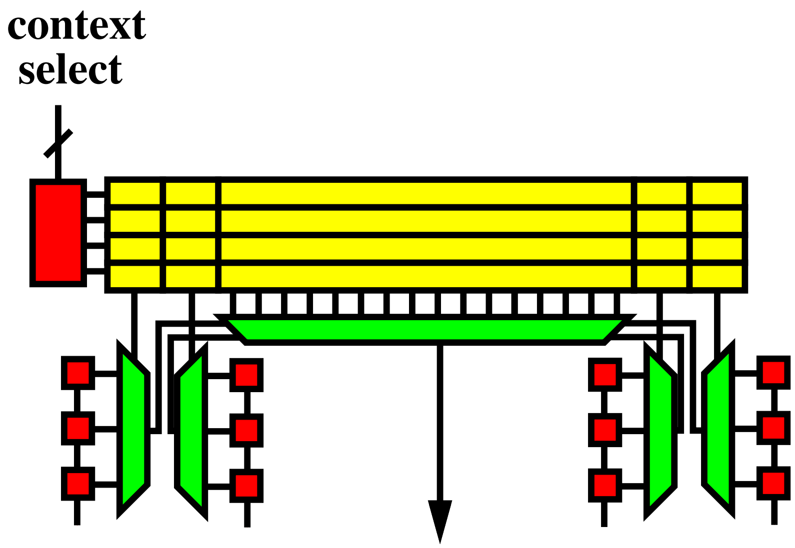
iDPGA – Adding Input Registers



FPGA
Array Element



DPGA
Array Element



iDPGA
Array Element

ASCII HEX→Binary Conversion

4 LUTs in largest stage

$$A_{time-pipe} = \left(4 \cdot 500\text{K}\lambda^2 + 24 \cdot 40\text{K}\lambda^2\right) + 16 \cdot 90\text{K}\lambda^2\Big) \\ 4.4\text{M}\lambda^2$$

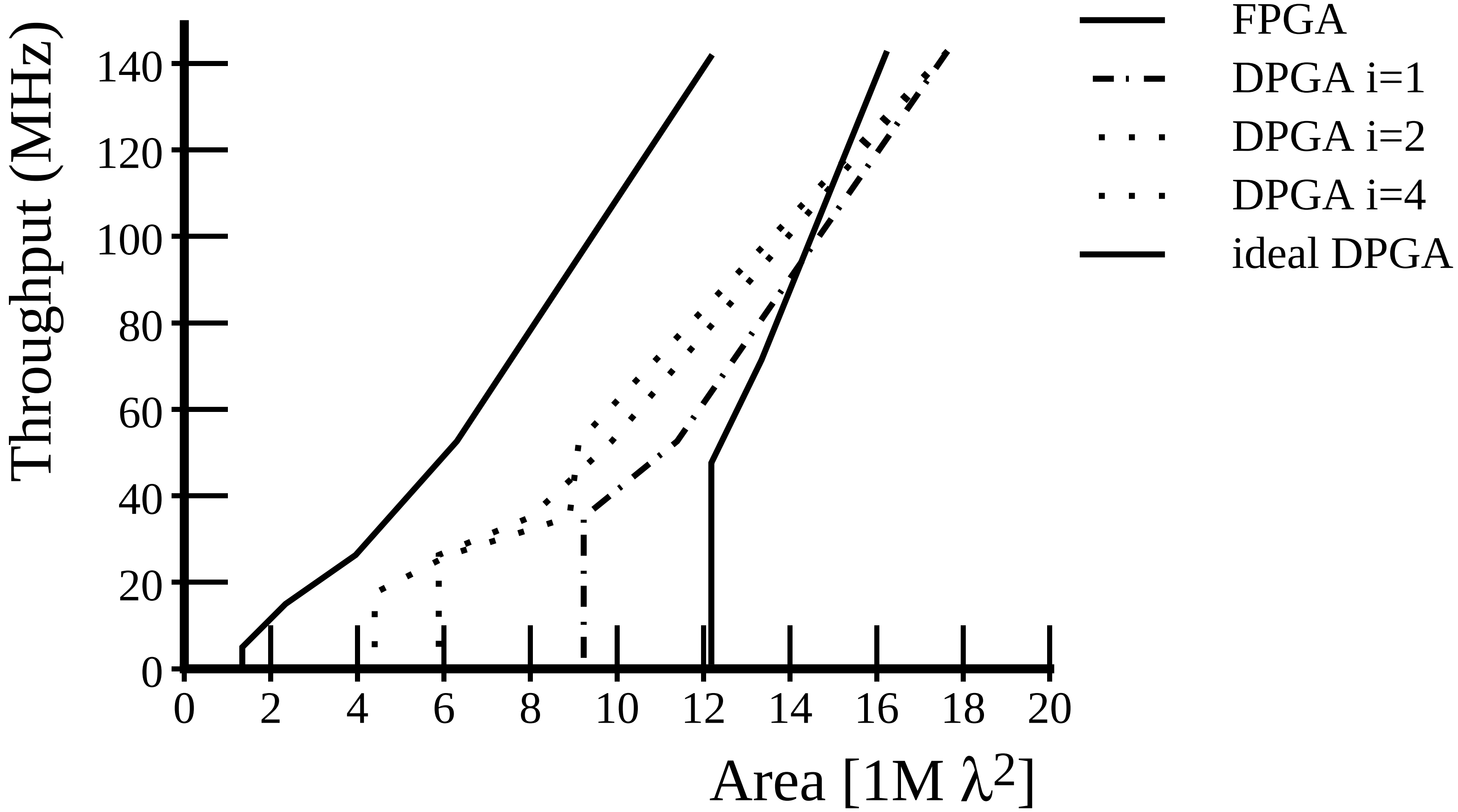
$$A_{base} = 500\text{K}\lambda^2$$
$$A_{config} = 40\text{K}\lambda^2$$
$$A_{inputs} = 90\text{K}\lambda^2$$

i=4, 6 timesteps (contexts)
at ≈7-9.5 ns/LUT
17-24M chars/sec

Temporal Pipelining (iDPGA)

Area vs Throughput

ASCII HEX→Binary Conversion



MIT AI Lab

Summary

Summary

- Computational area determined by:
  - *descriptive complexity*
  - *computational throughput*
- FPGAs ($N_a = N_d$)
  - efficient only for highest throughput requirements
- DPGAs ($N_d > N_a$)
  - efficient when throughput requirements limited
- Serialization
  - Can run into retiming limits
  - Solve by:
    - time-slicing or input retiming registers

Additional Information

- Full Technical Report in HTML
  - <http://www.ai.mit.edu/projects/transit/tn136/tn136.html>
- Full Technical Report in PostScript
  - <ftp://transit.ai.mit.edu/transit-notes/tn136.ps.Z>
- FPD'96 paper in PostScript
  - <ftp://transit.ai.mit.edu/paper/dpga-fpd96.ps.Z>
- PostScript for these slides
  - <ftp://transit.ai.mit.edu/slides/dpga-fpd96.ps.Z>
- **Reinventing Computing Home Page**
  - <http://www.ai.mit.edu/projects/transit/rc_home_page.html>
- **DPGA Information**
  - <http://www.ai.mit.edu/projects/transit/dpga_prototype_documents.html>