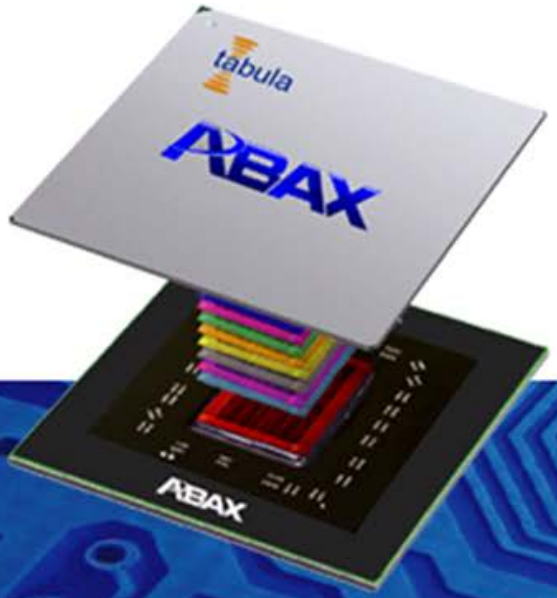
tabula.

Going beyond the FPGA with Spacetime

FPL 2012

Oslo, Norway



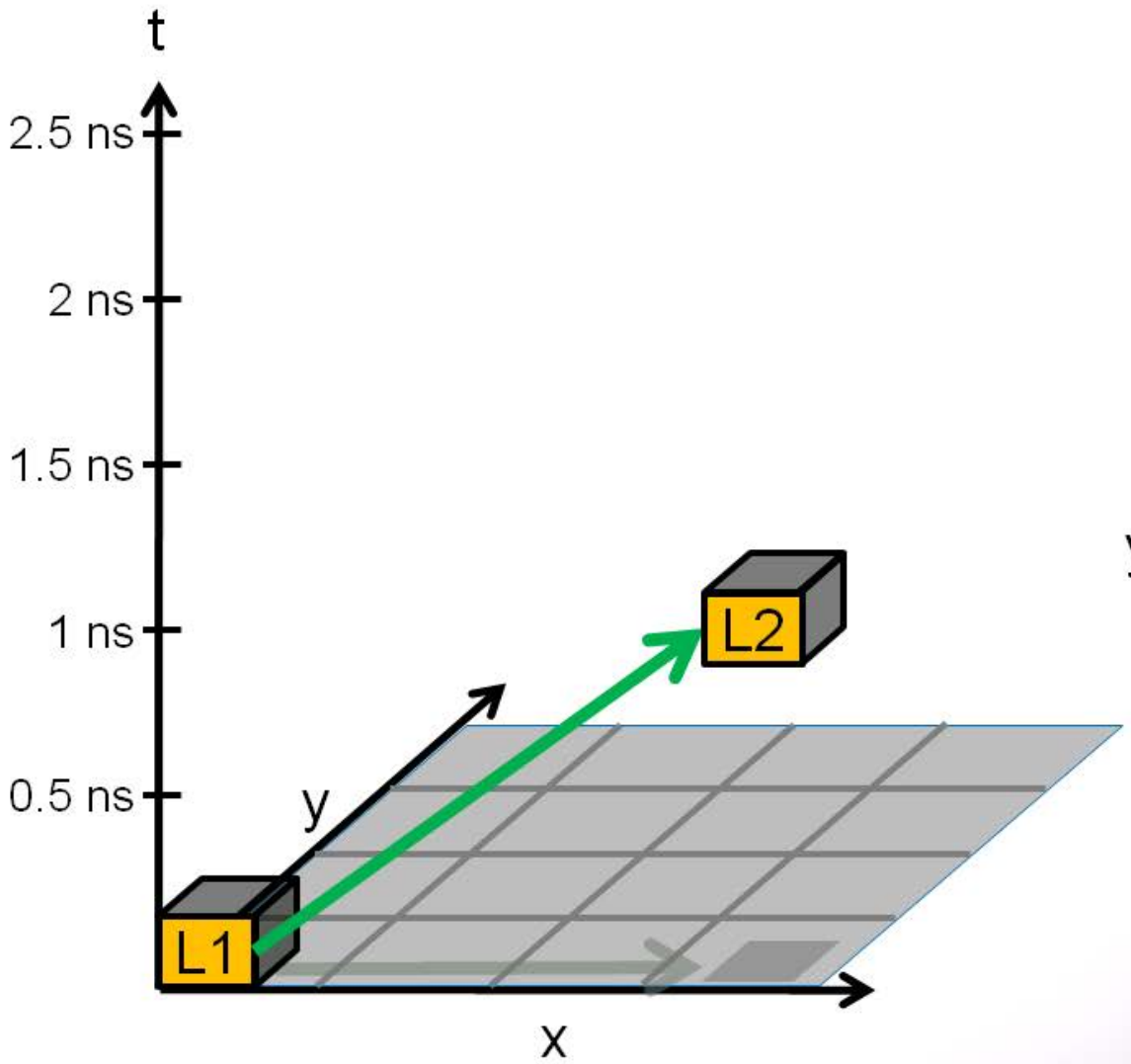
August 31, 2012

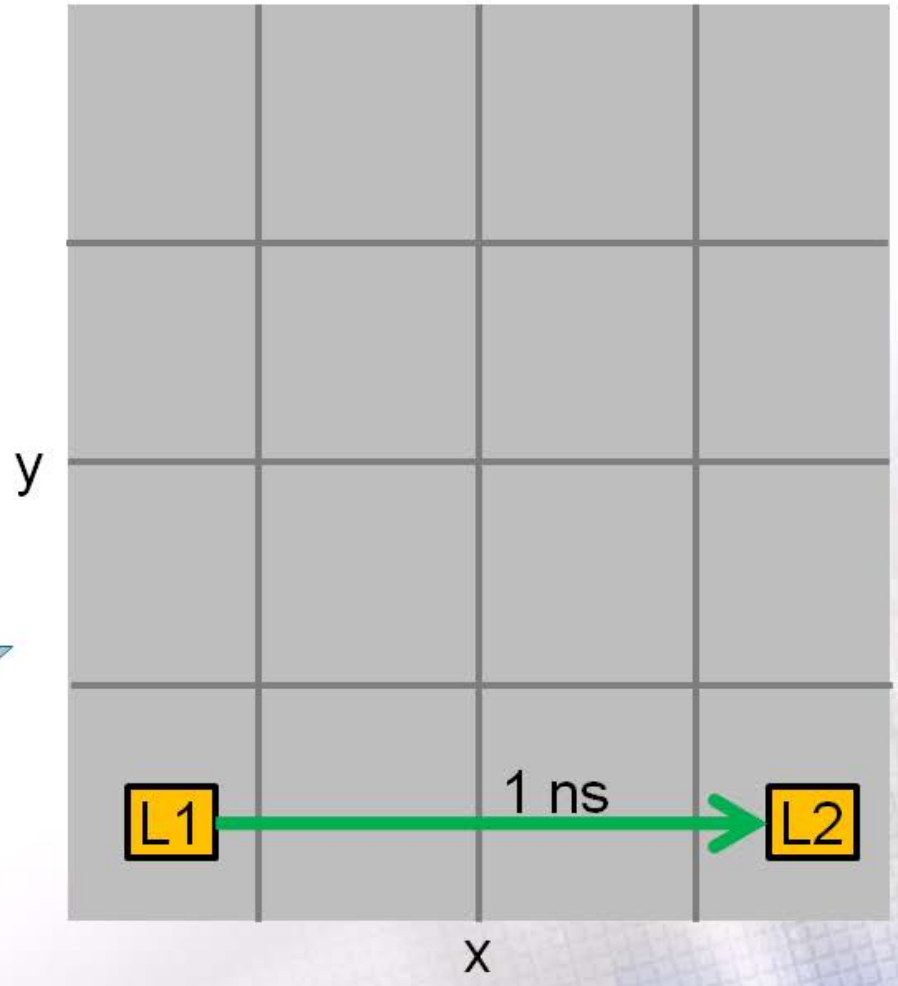
THE BEST PATH FROM IDEAS TO PRODUCTION SILICON ®

Programmable devices: good news and bad news

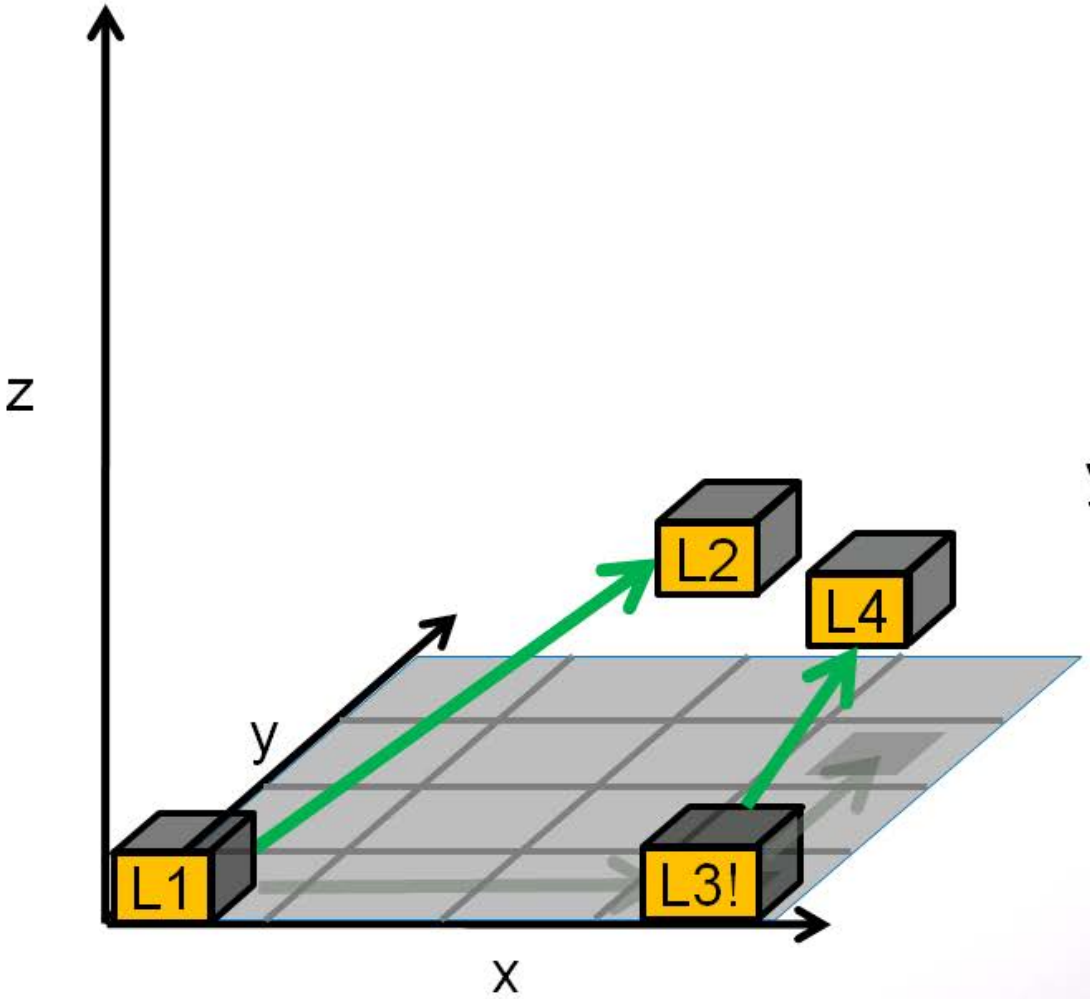
- The good news
  - Programmable!
  - On leading-edge process node
- The bad news
  - Very expensive!
  - Very large!
  - Very slow!
  - Very power-hungry!
  - Very hard to program (vs. CPU)!
  - Controlled by a duopoly
- How can we build something better than an FPGA?

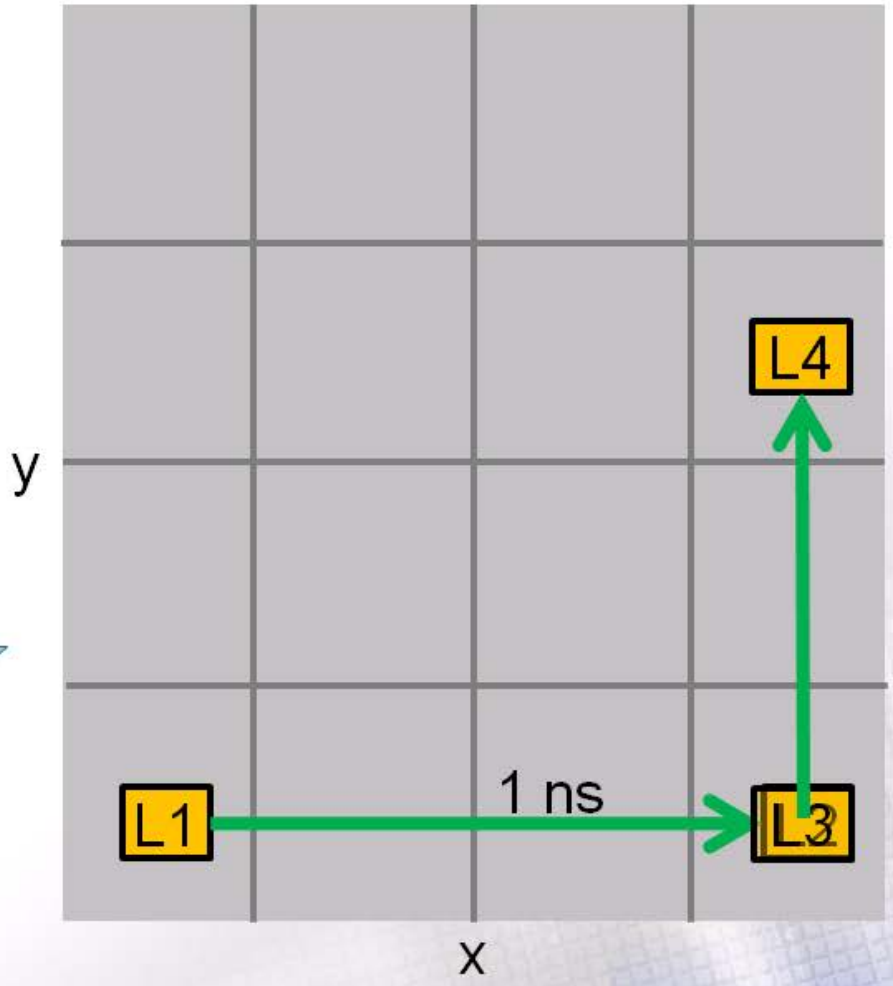
FPGA → 3PLD



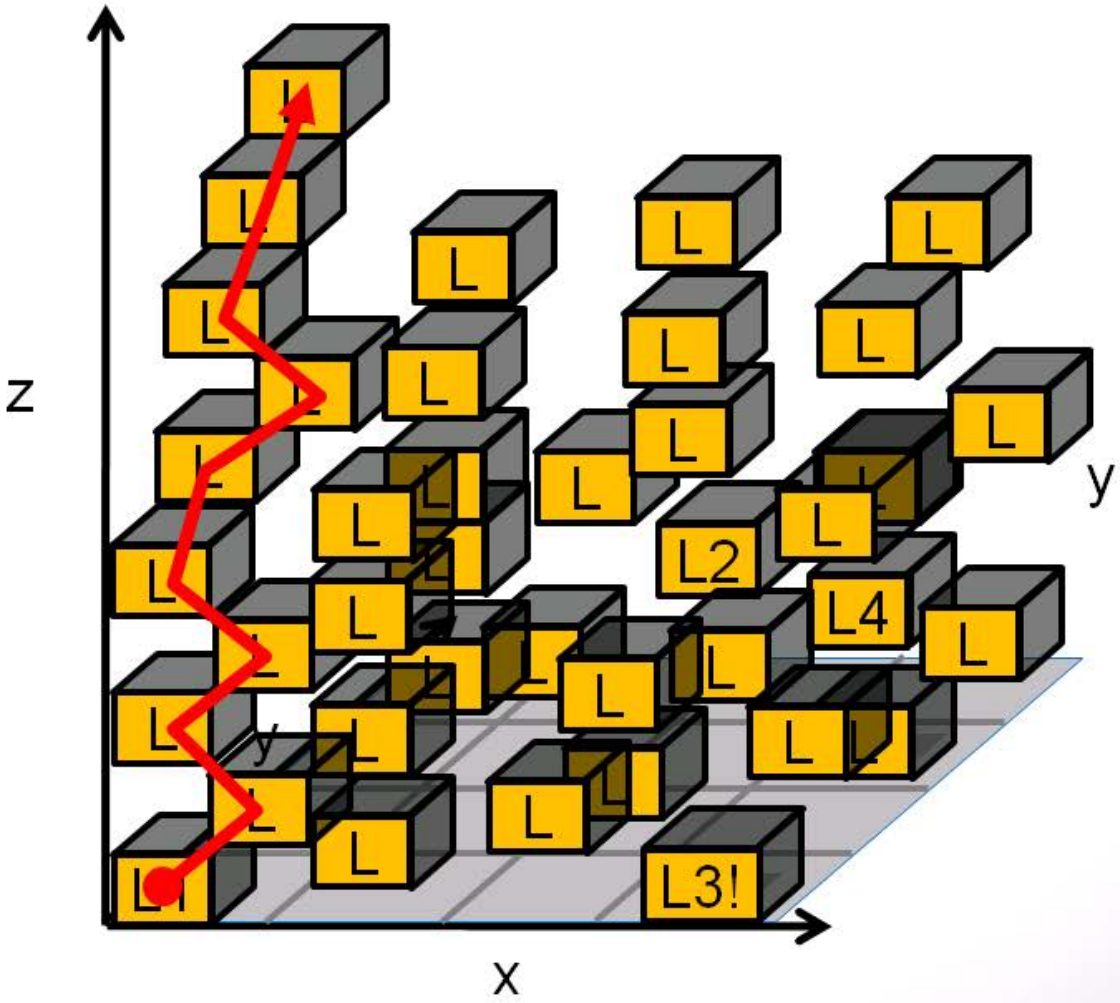


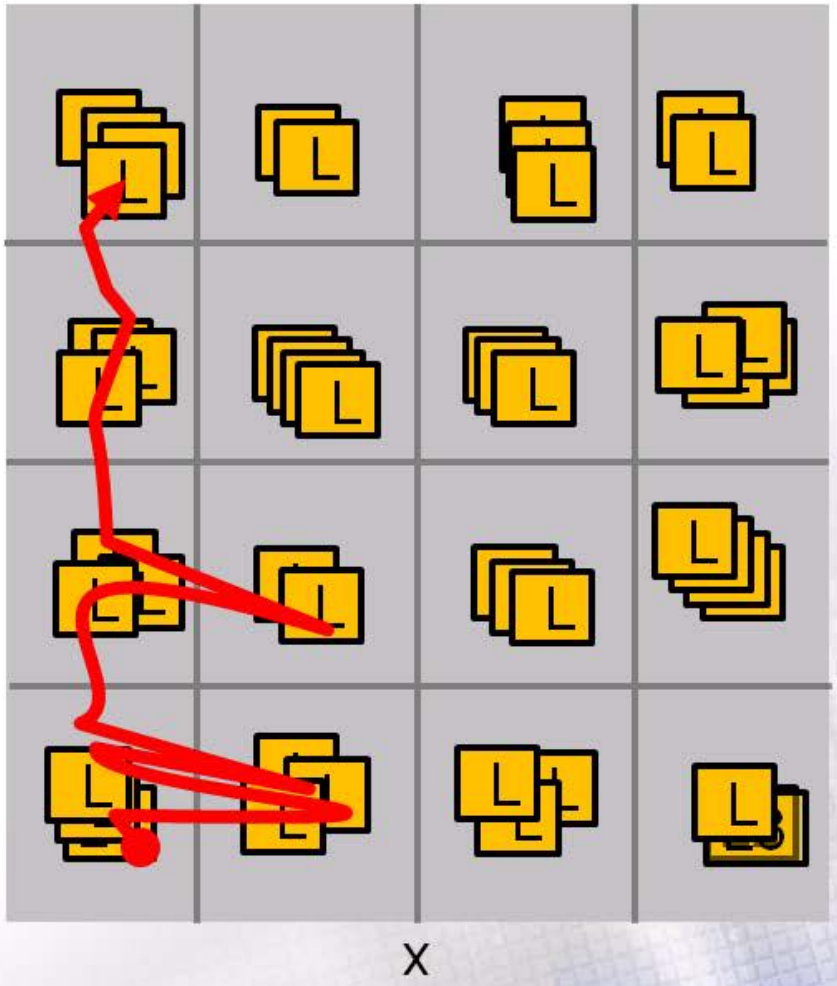
FPGA → 3PLD





FPGA → 3PLD





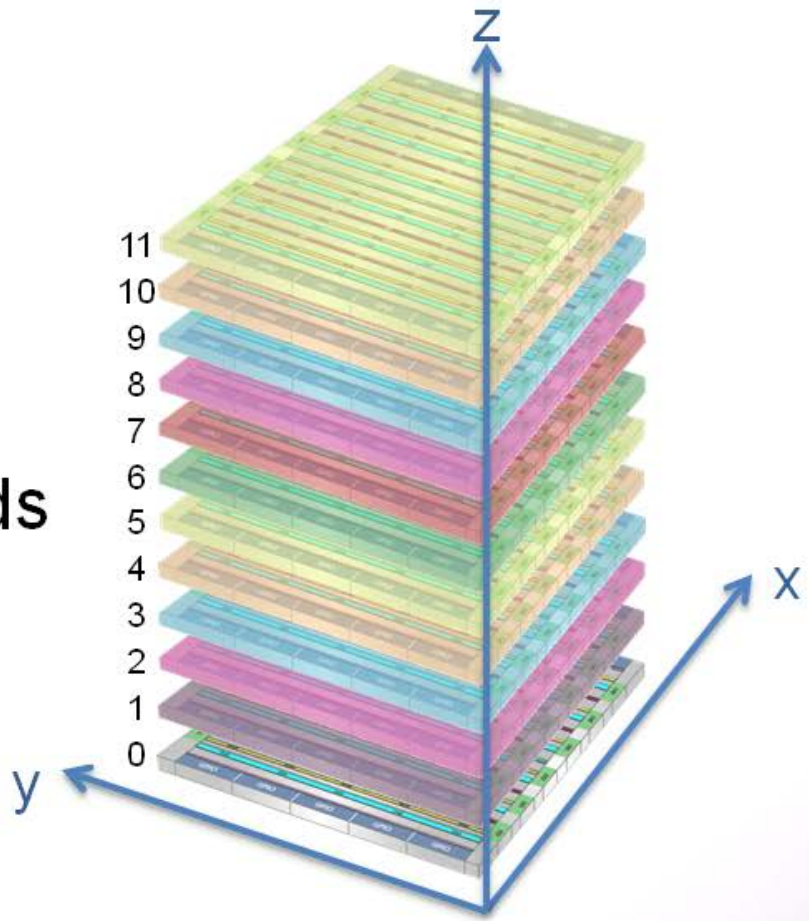
Dimension is a well-defined mathematical term

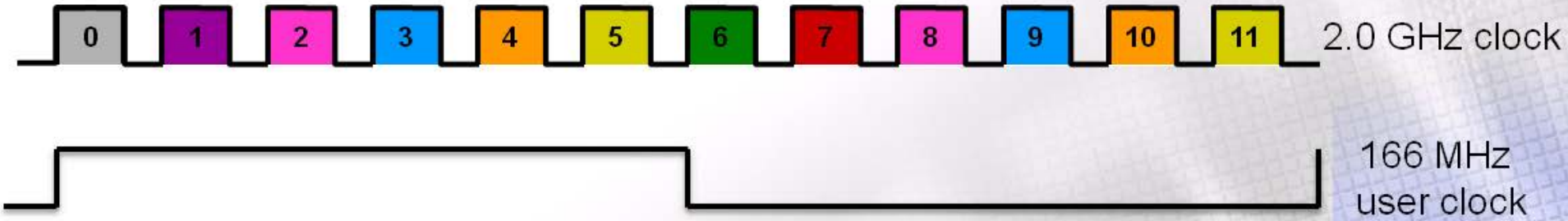
- Wikipedia: "the **dimension** of a space or object is […] the minimum number of coordinates needed to specify any point within it."
- Each LUT in Spacetime requires 3 coordinates to identify uniquely
  - I.e., (x,y,t), which can also be represented as (x,y,z)
- Thus, Spacetime is 3-dimensional – literally
  - Not "virtually"!

- FPGAs have only one layer of LUTs: they are 2-D
  - Like ASICs, they have many layers of interconnect
- Each LUT in an interposer-enhanced FPGA can still be uniquely identified by two coordinates: (x,y)
  - The interposer just adds more layers of interconnect
- Thus, interposer-enhanced FPGAs are also 2-dimensional

Implementing the third dimension in Spacetime

12 folds





A very small sample of the interesting problems

- How can you close timing?
- Can you achieve high performance?
- Doesn't the time to reconfigure kill you?

- Where does the state go during reconfiguration?
- How do you map designs to this fabric?
- Do you have to design differently for such an architecture?

- Interaction between asynchronous domains

- How do memories work?

- Isn't the power awful?

- Can you build an analytical placer in Spacetime?

- Interaction between related clock domains

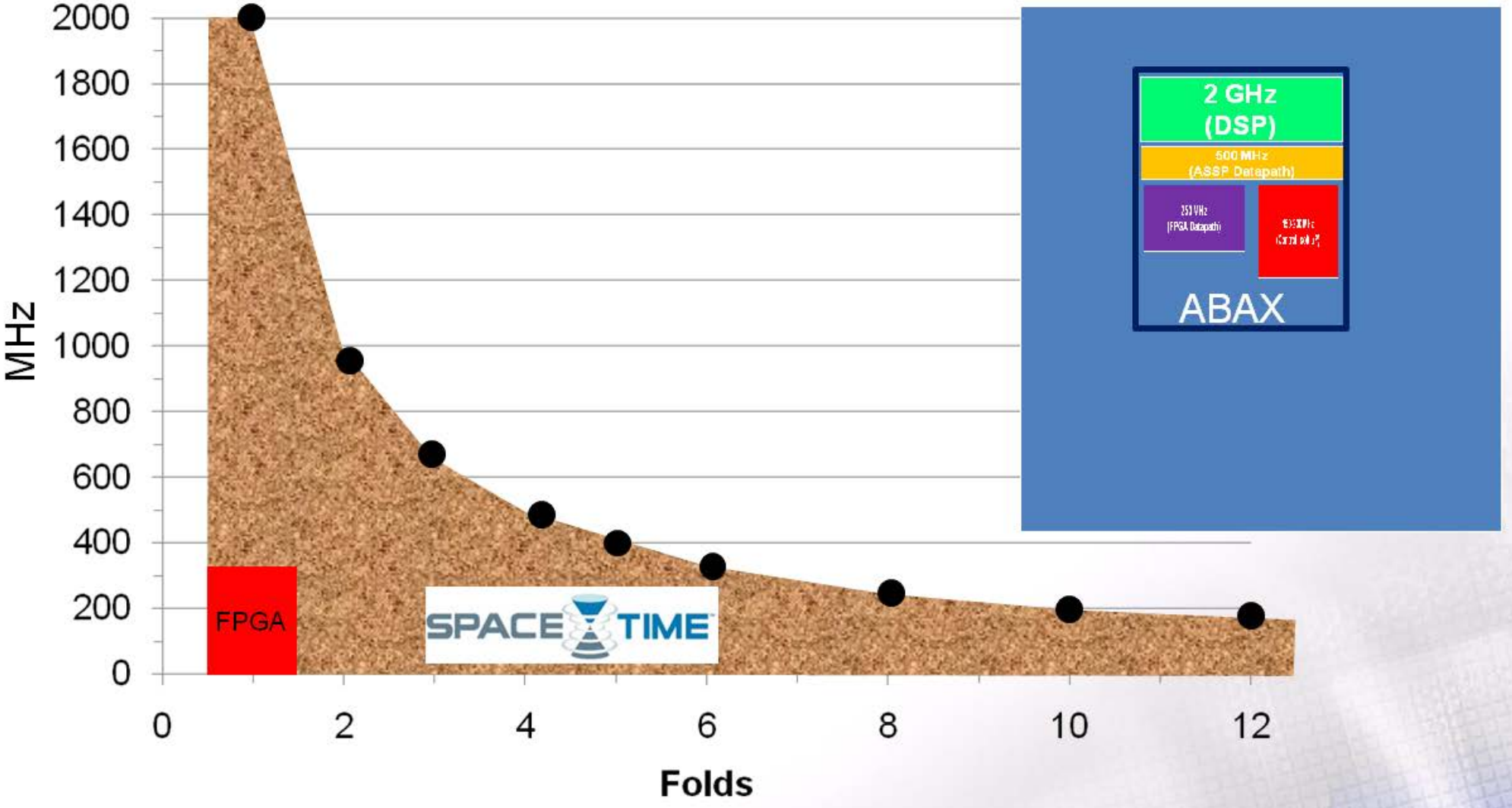
- What does the router look like?
- How do you compute minimal spanning trees?
- How do you compute Steiner trees?

- How do you account for the reconfiguration time in timing-driven P&R?
- How is logic synthesis affected by Spacetime?

- What does clock gating look like in Spacetime?

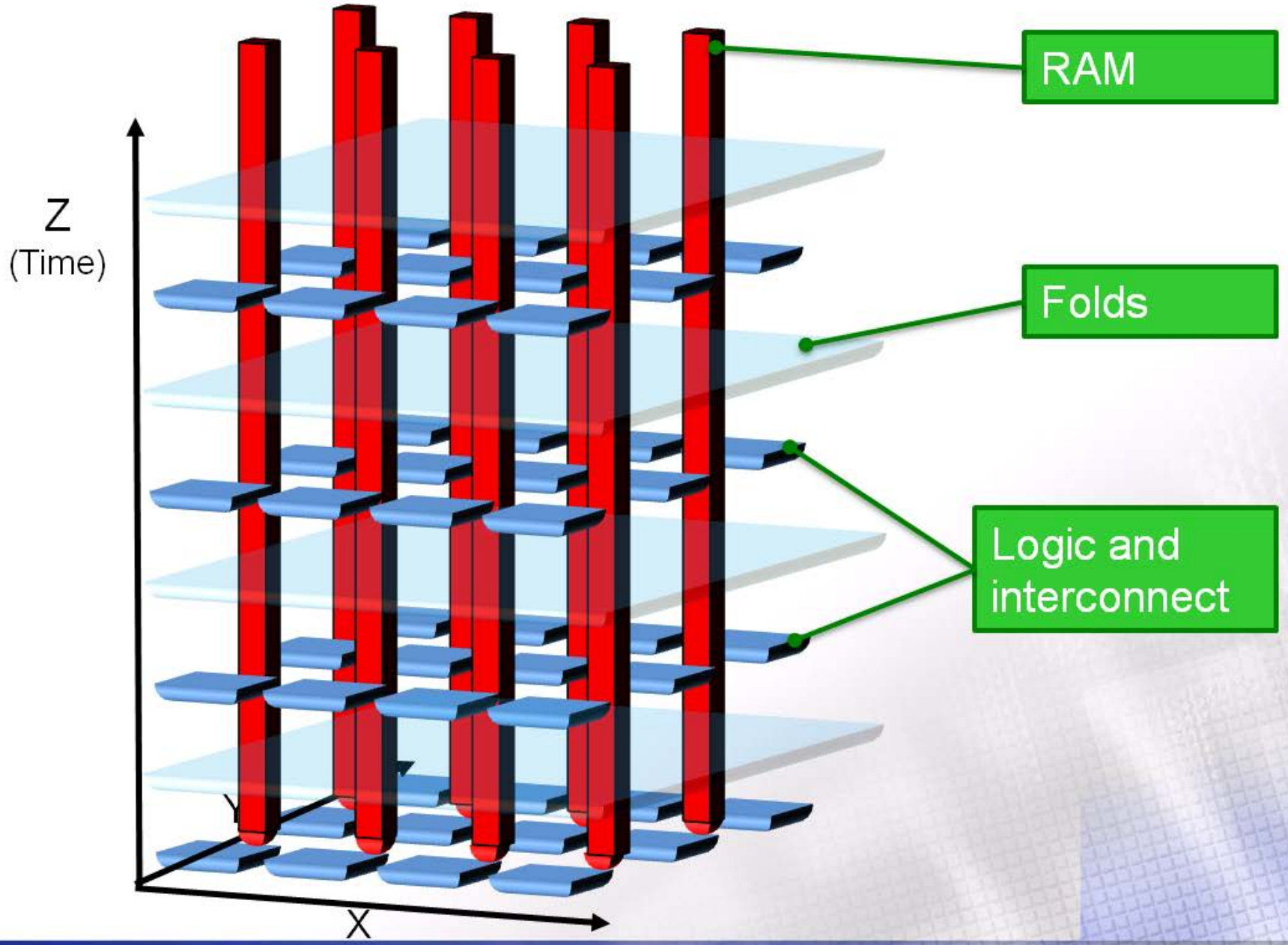
- How does DSP work?

Some logic must be fast, but most logic can be slow

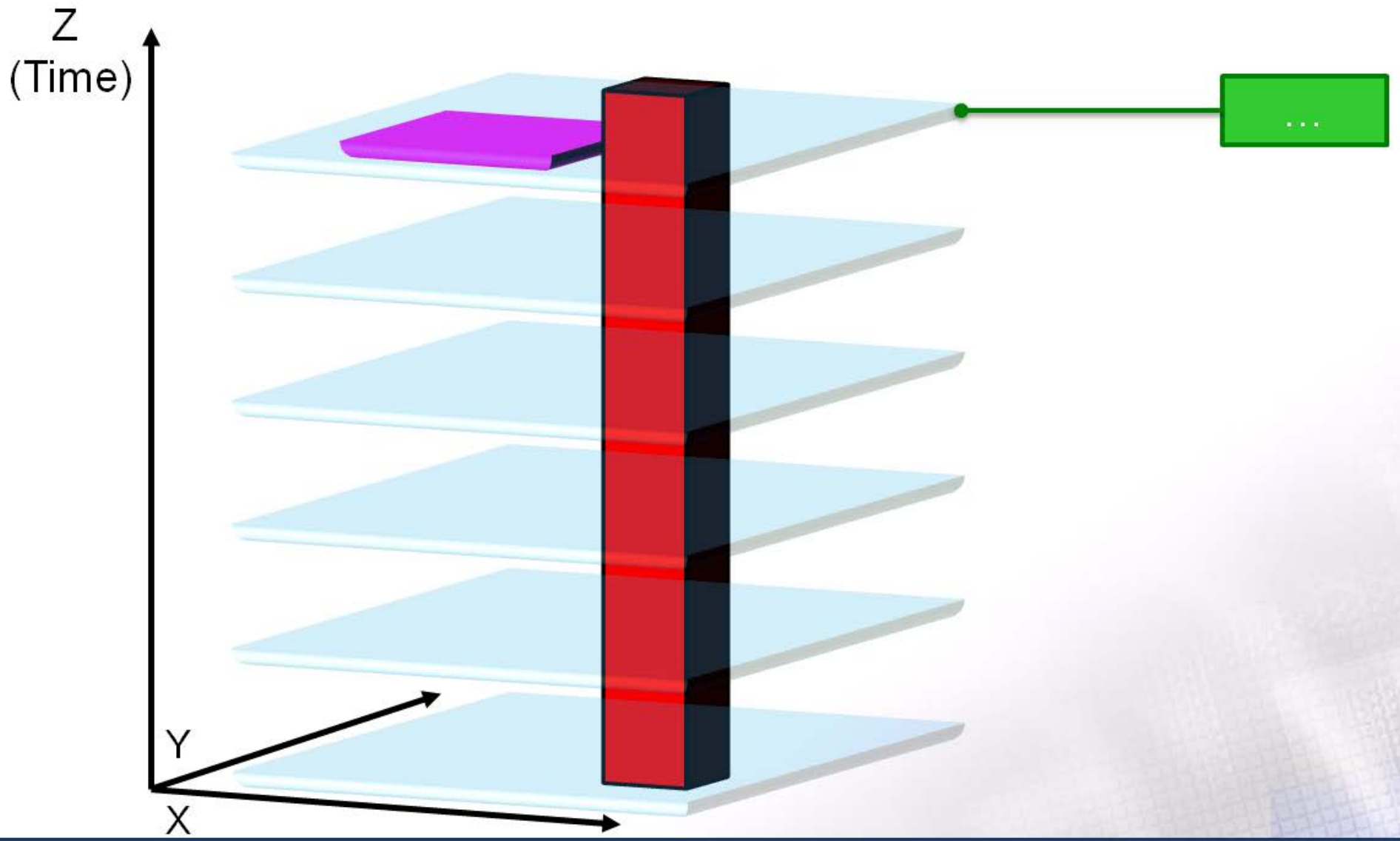


Stylus™ software automatically co-optimizes space and time

Spacetime memories



Spacetime memories are fast and have lots of ports

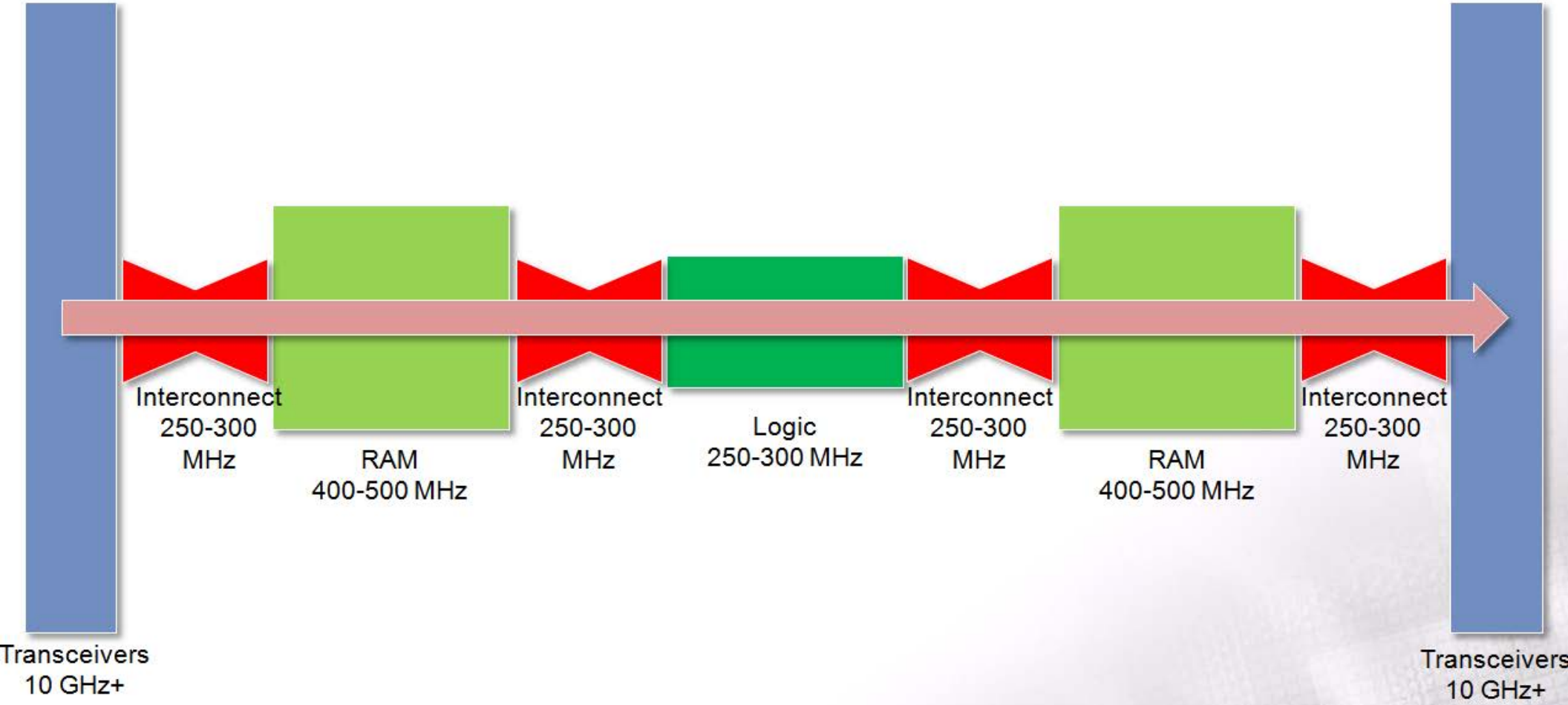


Stylus infers multi-ported memories automatically from RTL

Spacetime memories

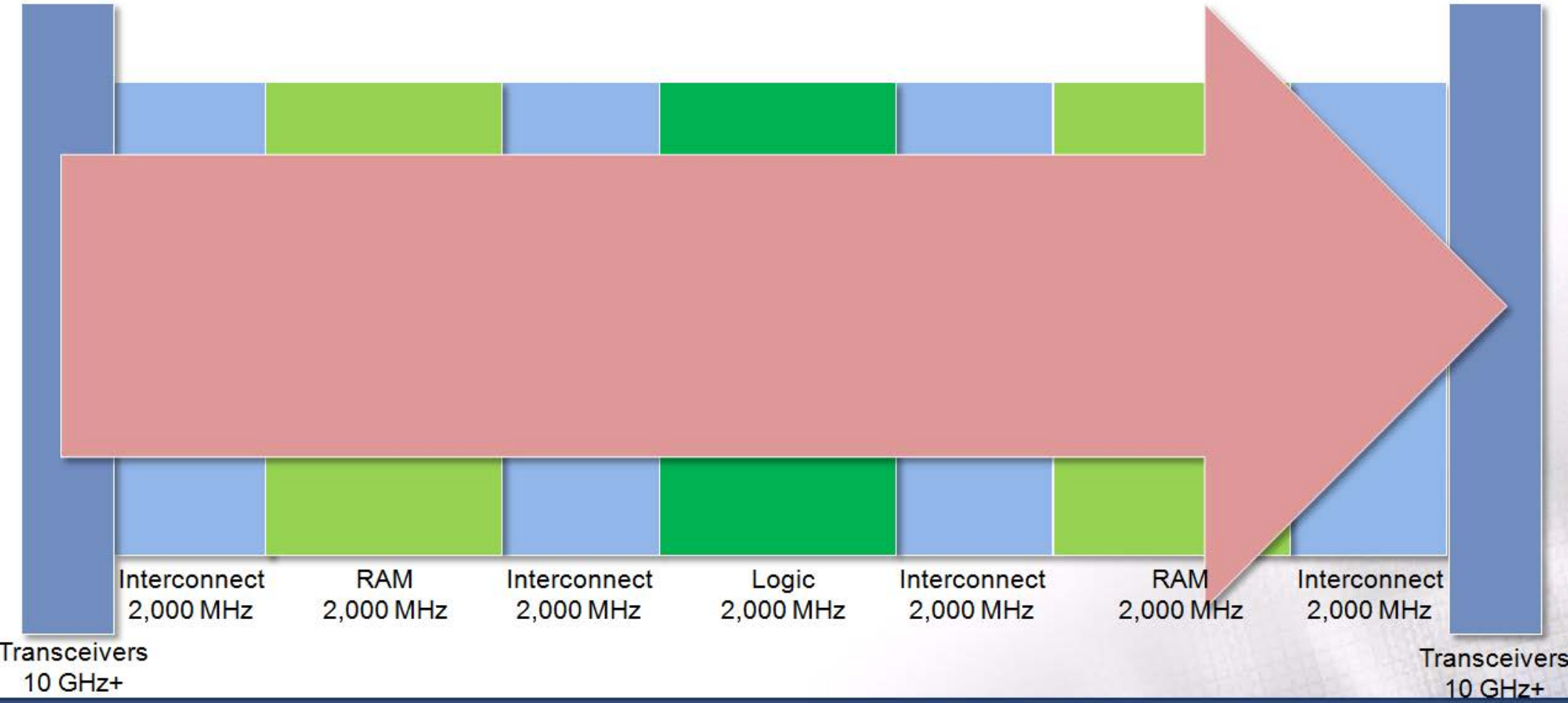
- Spacetime memories can use single-ported RAM cells
  - Half the size of the dual-ported RAM cells in FPGAs
- 2 GHz throughput – faster than most ASICs
  - 2 GHz in practice → >3x faster than dual-ported FPGA memory
  - Great for 100 Gbps networking
- 12 different pieces of logic can be adjacent to the same memory!
  - Very low latency
- DSPs can be fed at 2 GHz
  - ➔ DSPs can run at 2 GHz in practice
- Multiple user memories can be "folded" into one Spacetime memory
  - Non-overlapping bits in space
  - Non-overlapping ports in time
- So, multiple pieces of logic can be simultaneously adjacent to multiple memories!

Programmable devices are about interconnect



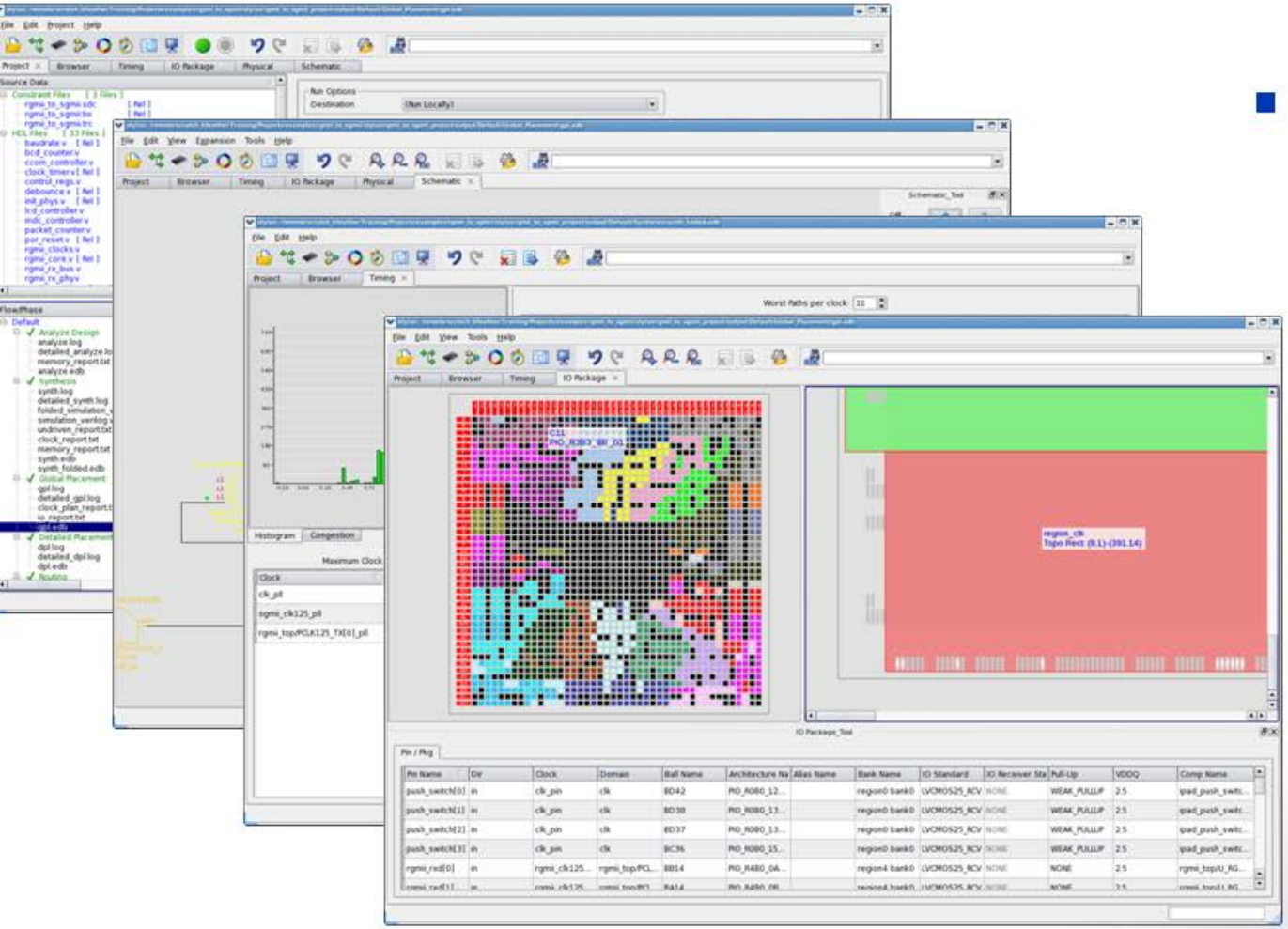
FPGA performance bogged down by long wires (= slow interconnect)

Spacetime addresses the interconnect bottleneck



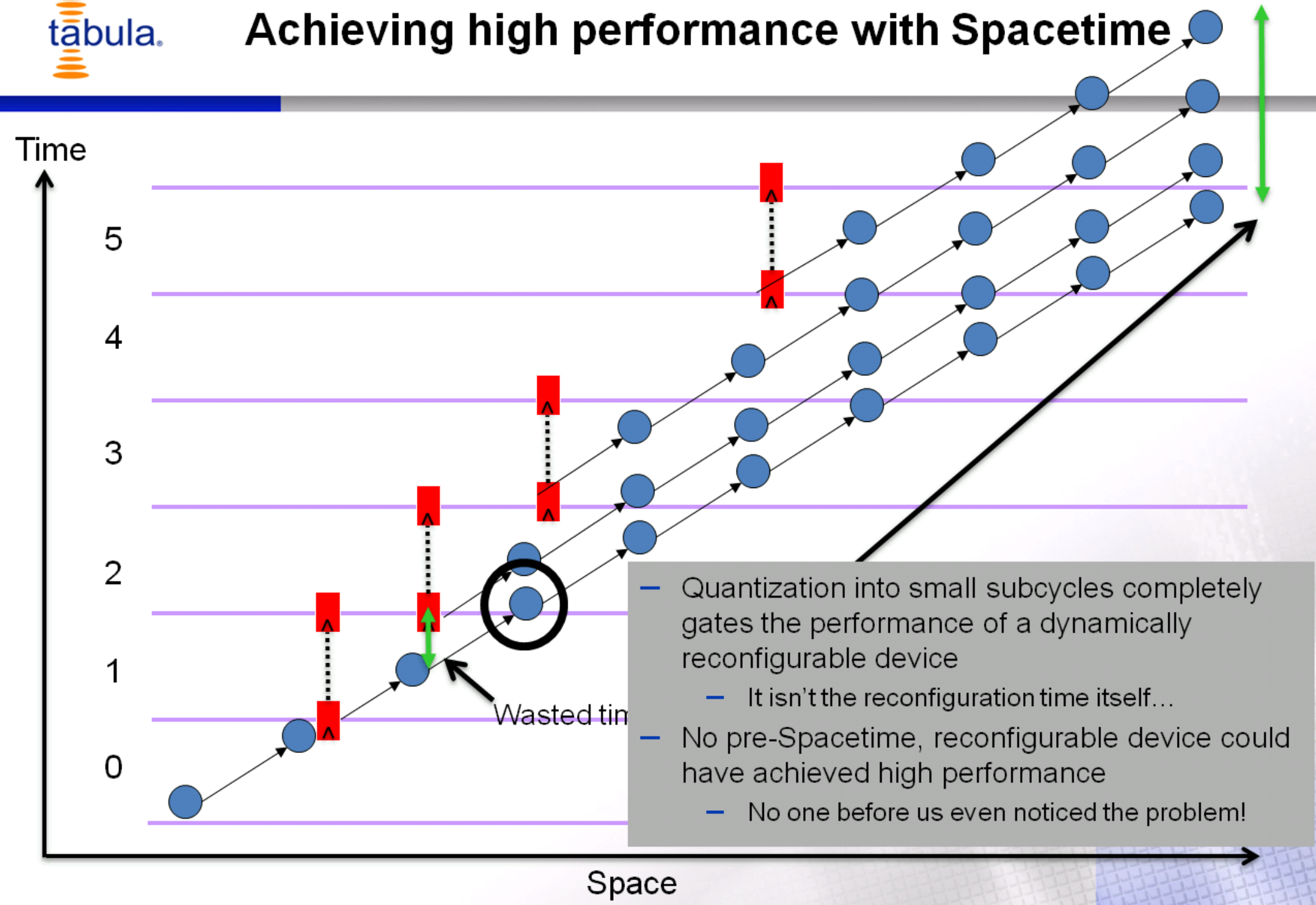
Everything can run at 2 GHz on our chips

Stylus: integrated design system



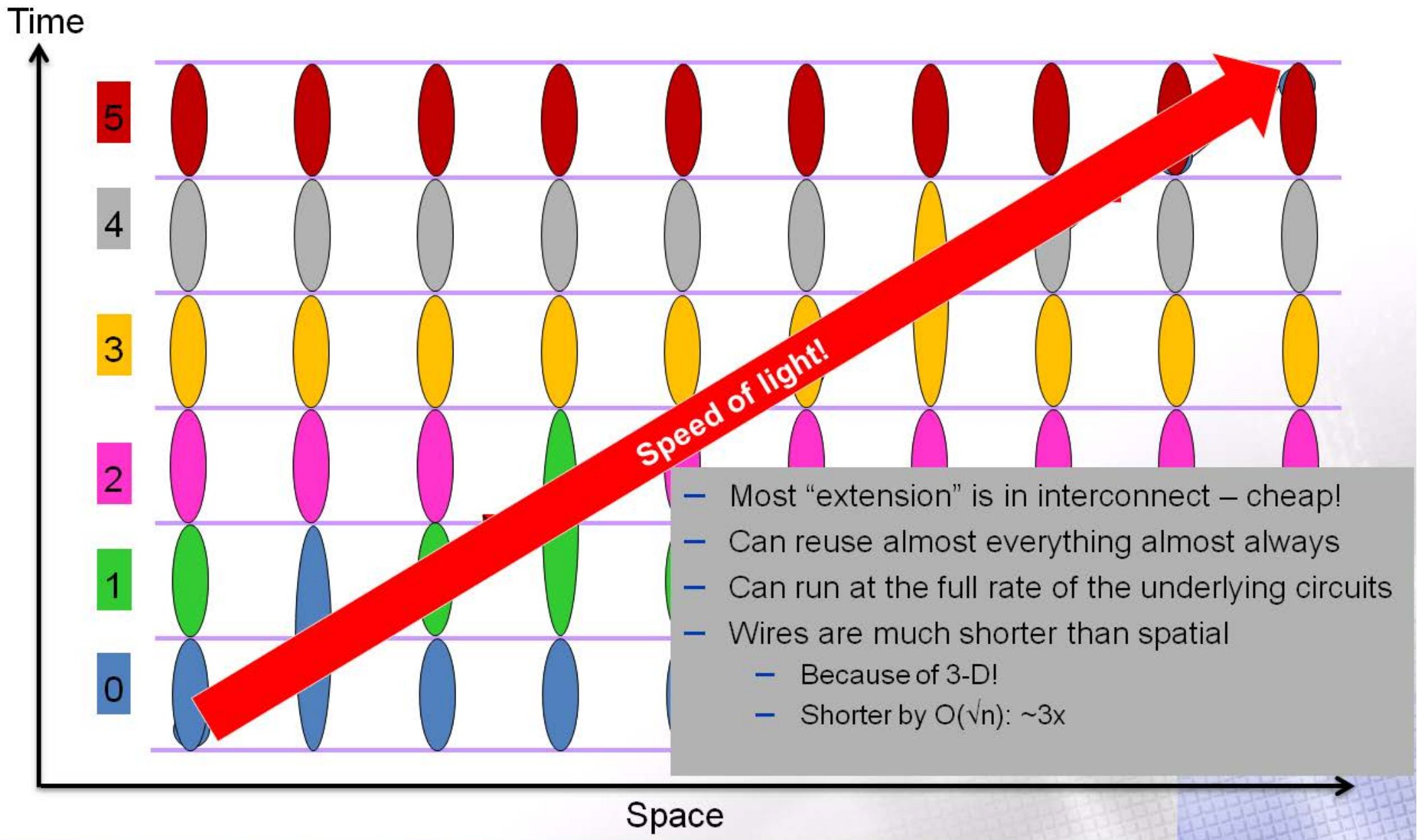
- Completely integrated design environment
  - Project management
  - HDL and Schematic browsing
  - Timing analysis and correlation back to HDL
  - I/O Pin and Board planning
  - …

Achieving high performance with Spacetime



- Quantization into small subcycles completely gates the performance of a dynamically reconfigurable device
  - It isn't the reconfiguration time itself…
- No pre-Spacetime, reconfigurable device could have achieved high performance
  - No one before us even noticed the problem!

Achieving high performance with Spacetime

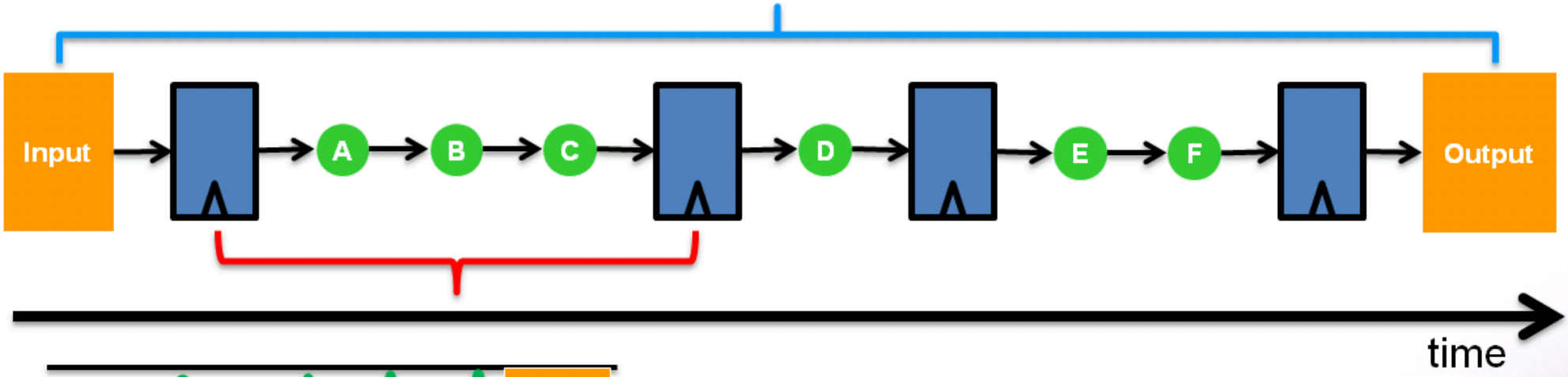


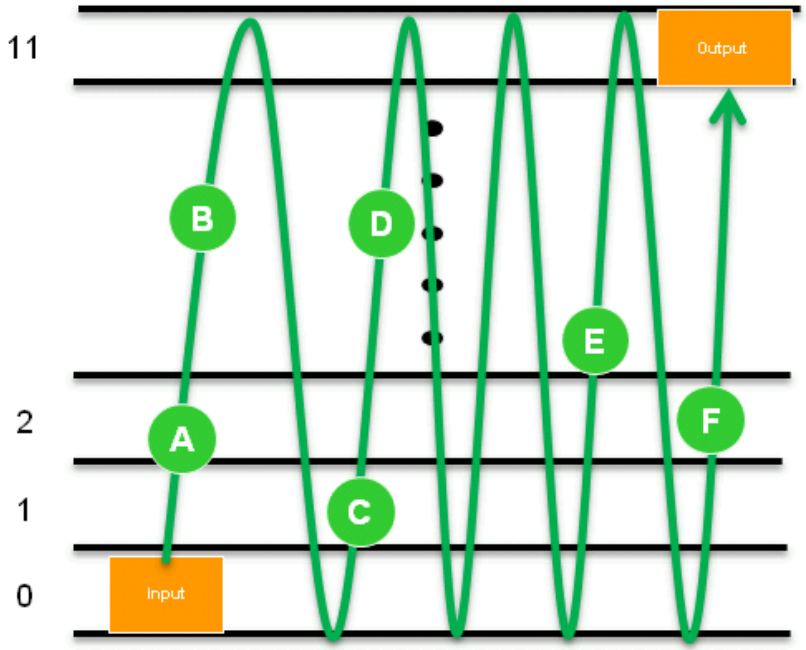
- Most "extension" is in interconnect – cheap!
- Can reuse almost everything almost always
- Can run at the full rate of the underlying circuits
- Wires are much shorter than spatial
  - Because of 3-D!
  - Shorter by O(√n): ~3x

Advanced Spacetime technique: *extension*

- Used pervasively by Spacetime P&R to optimize timing
  - Hides subcycle boundaries; treats time as continuous
  - Local reduction in foldedness of design
  - Removes quantization effects and reconfiguration timing 'tax'
- Allows design to run at maximum foldedness
  - More folds is always better
- Vital for high-performance, reconfigurable computing

Sequential timing: much easier timing closure





- ❑ Makes it much easier for Stylus to close timing
- ❑ Tells the user where the real timing problems are
- ❑ Makes it much easier for the user to close timing

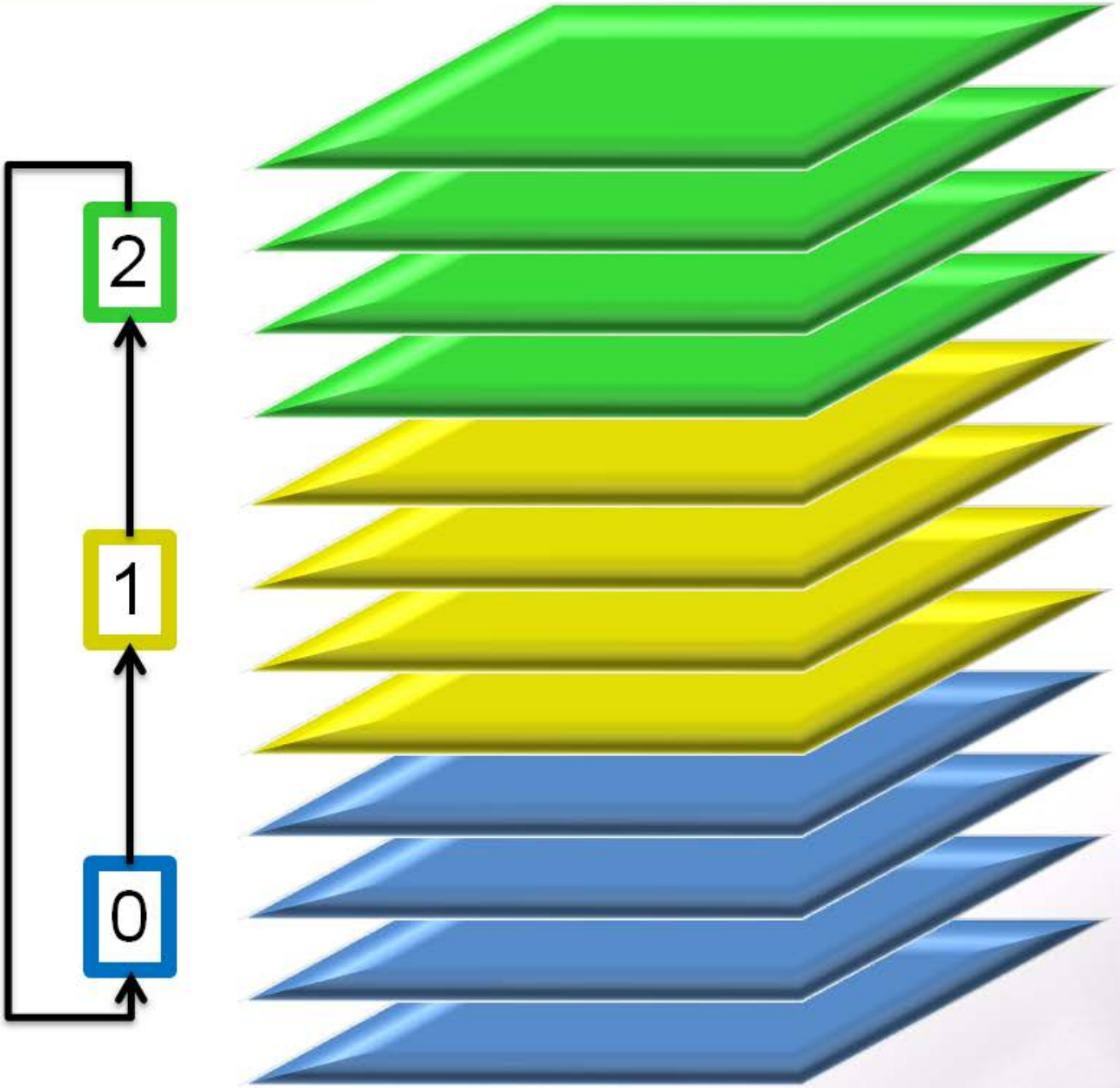
Sequential timing and rescheduling

- User state elements are simply a convenient way to name values
  - E.g., "the one that showed up here last time" vs. "… this time"
- Their meaning is just "a wire in Z" in Spacetime
- Must still meet externally visible timing requirements, but…
- Have tremendous flexibility to *reschedule* operations
  - To optimize area, performance, and power
- Spacetime enables very fine resolution of time
  - E.g., 500 ps folds vs. 5 ns cycles

- Rescheduling is as powerful as retiming, but it is not retiming
- Does not change netlist or ordering: preserves simulation behavior!
  - Can preserve user's notion of state (and names!)

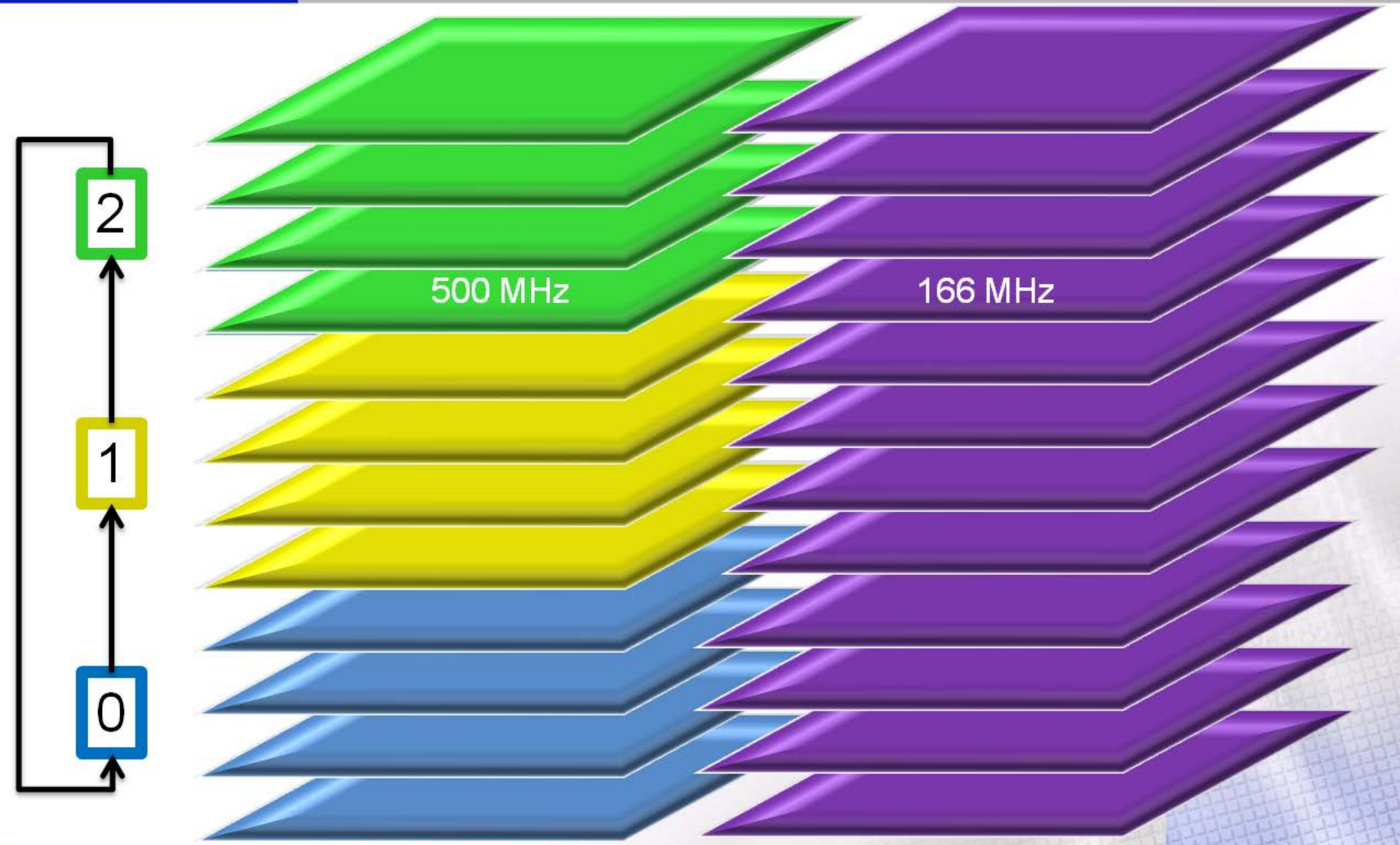
- P.S., "parallel" computing is not about parallelism….
- It's about rescheduling computations while preserving correctness

Achieving really high performance



- Reduce folds per cycle, and *rescale*
- Z resource increases quadratically with rescaling
- **Easier timing closure**
- **Improved routability**

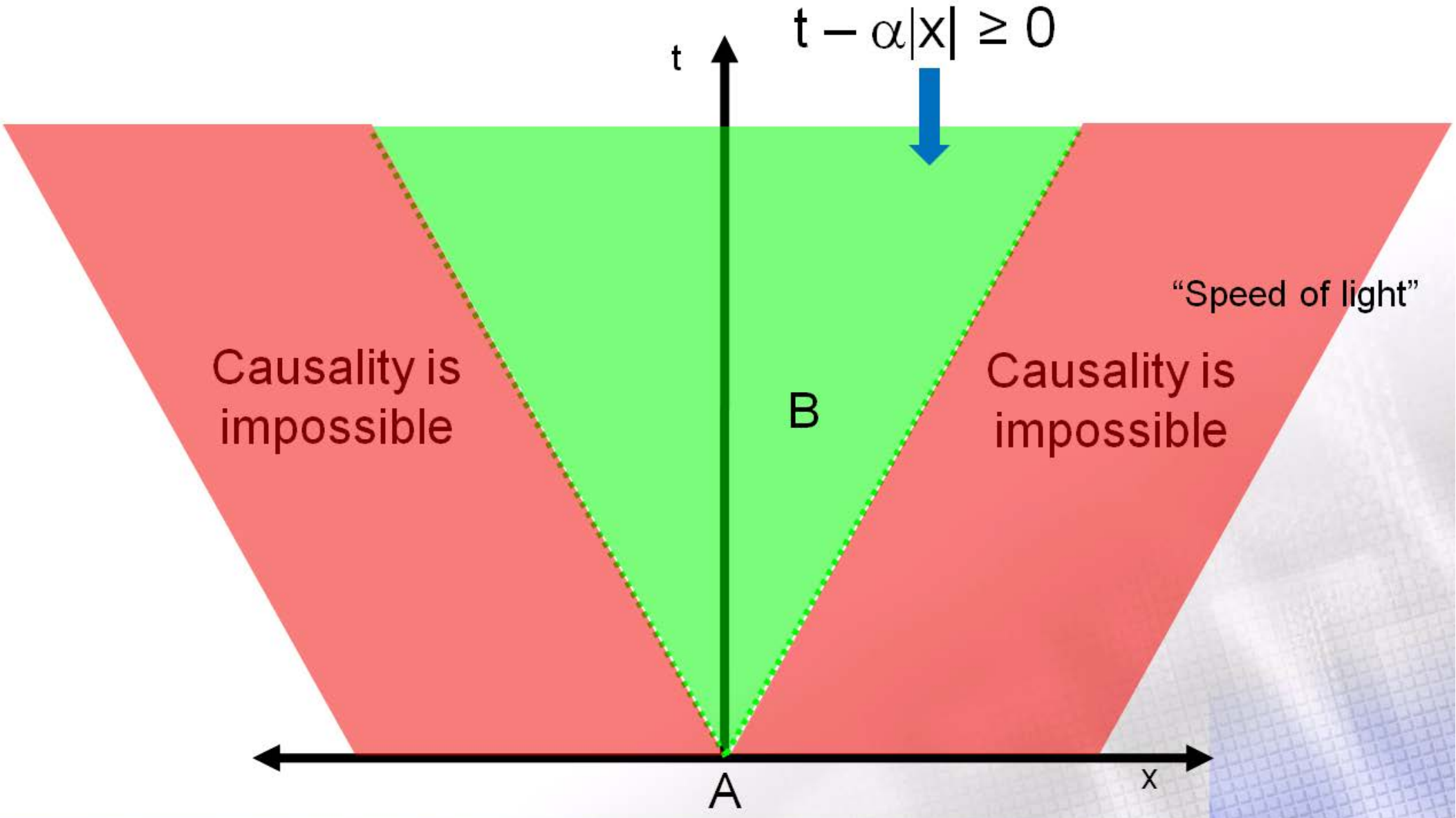
…and merging domains for better P&R



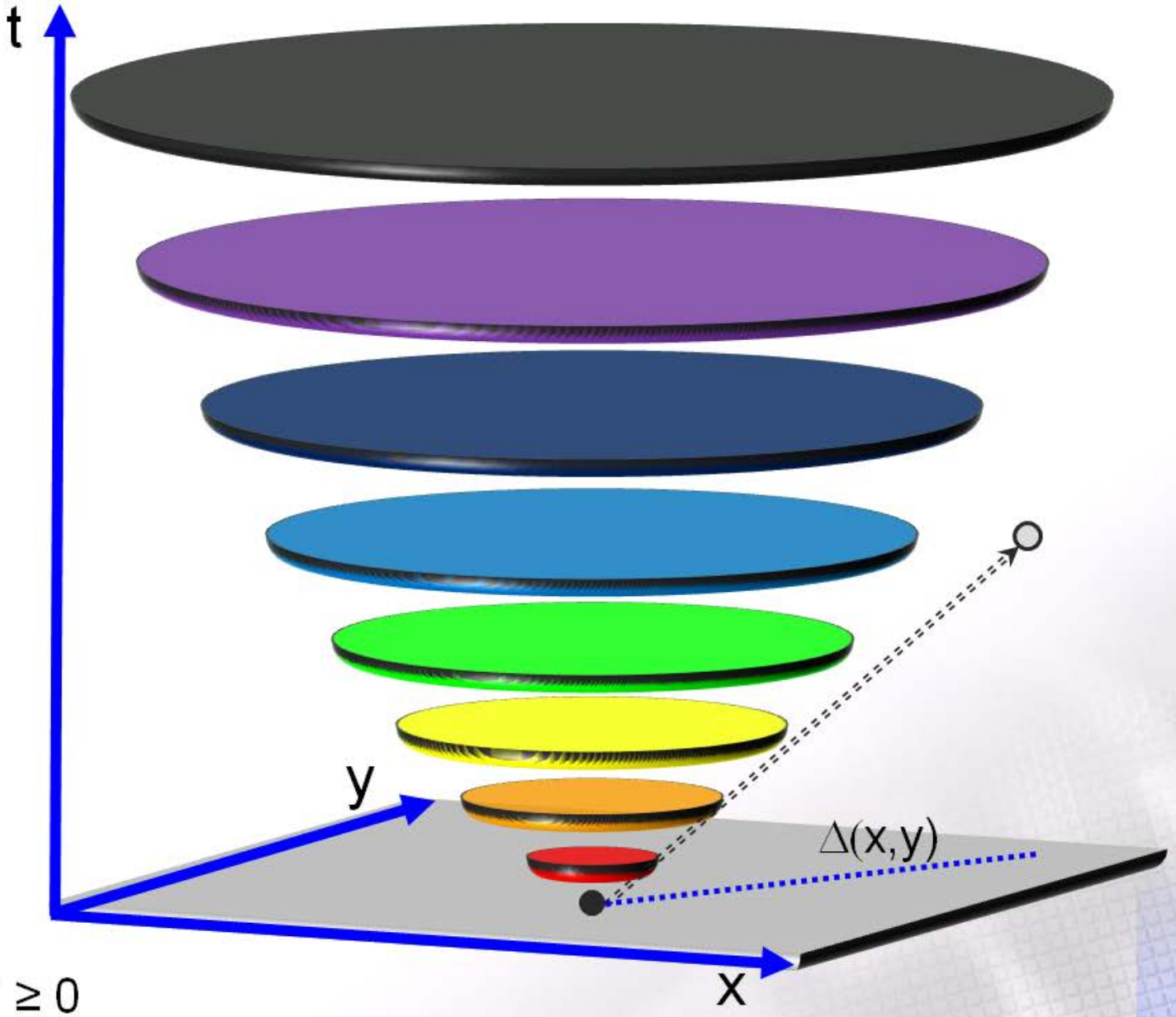
Mapping designs to Spacetime

- Objective: “Hide the Revolution”
  - Support traditional design methodology with RTL and SDC
  - Automatically map the designs to Spacetime (as though to ASIC or FPGA)
- Key observation: view the chip as 3-D!
  - Do not separate spatial placement from temporal scheduling
  - Do timing-driven placement in 3-D
- Spacetime geometry is Minkowski – not Euclidean or Manhattan
  - Component’s sinks must be in its *light cone*

Minkowski's spacetime in two dimensions

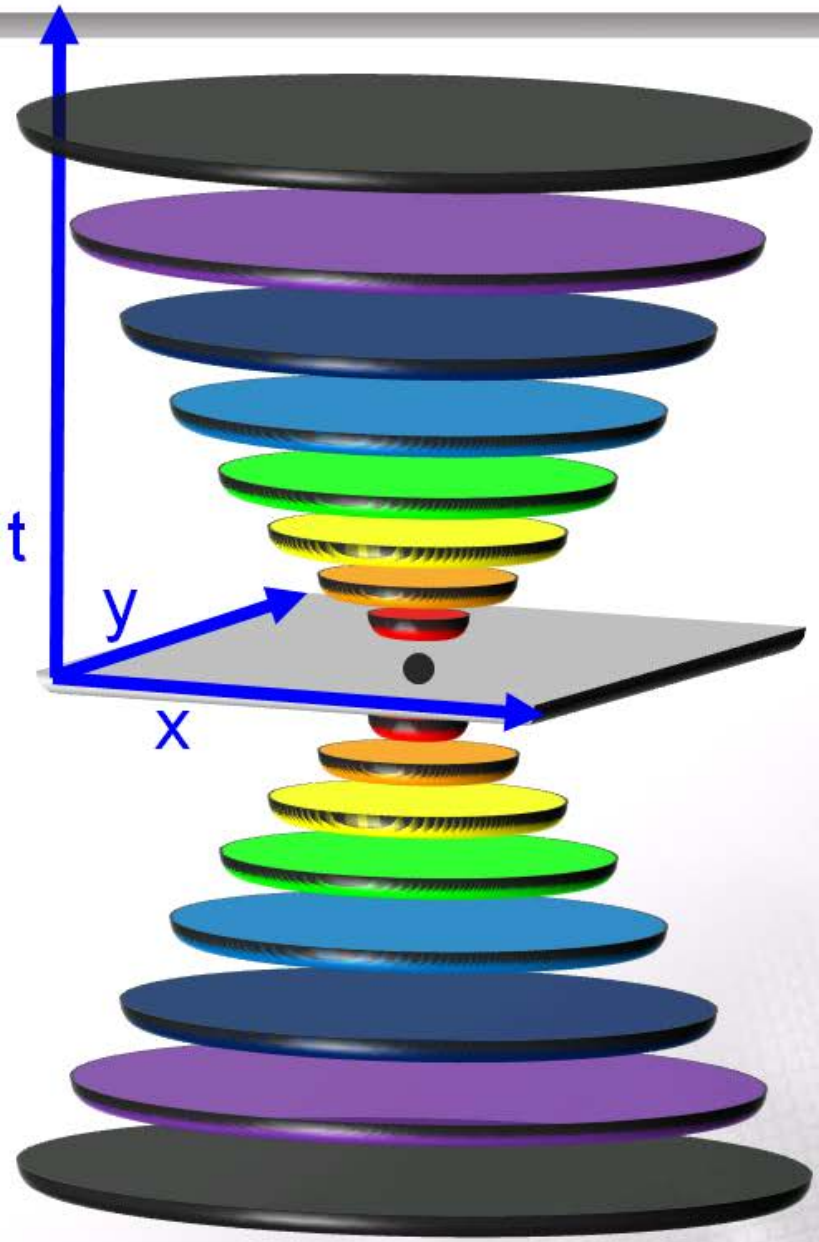


Minkowski spacetime in three dimensions

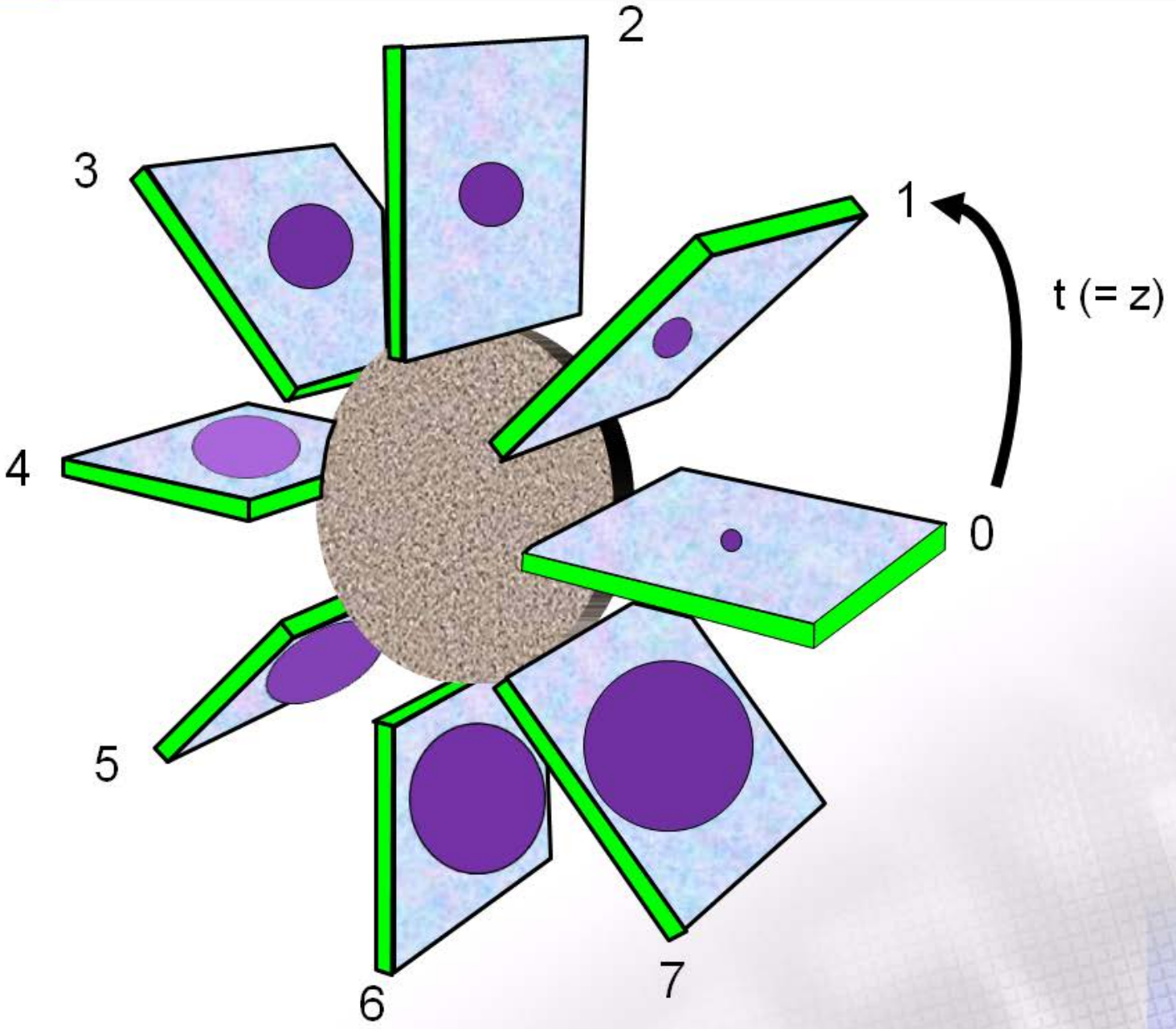


$t - \alpha|x| - \beta|y| \geq 0$

Minkowski spacetime in three dimensions, cont.



Spacetime cones live on a torus



Managing asynchronous clock domains

- Designs with multiple, unrelated clock domains are ubiquitous
  - I/O standards
  - IP
  - Debug/monitoring module
- Need a way to transmit data between domains
- Problem: metastability!
  - Unavoidable
- Mitigation strategy: time
  - Longer wait time → reduced probability of metastability

- Three common implementation strategies in non-Spacetime parts
  - Asynchronous FIFO
  - Dual-clocked RAM
  - Chain of state elements (e.g., flops)

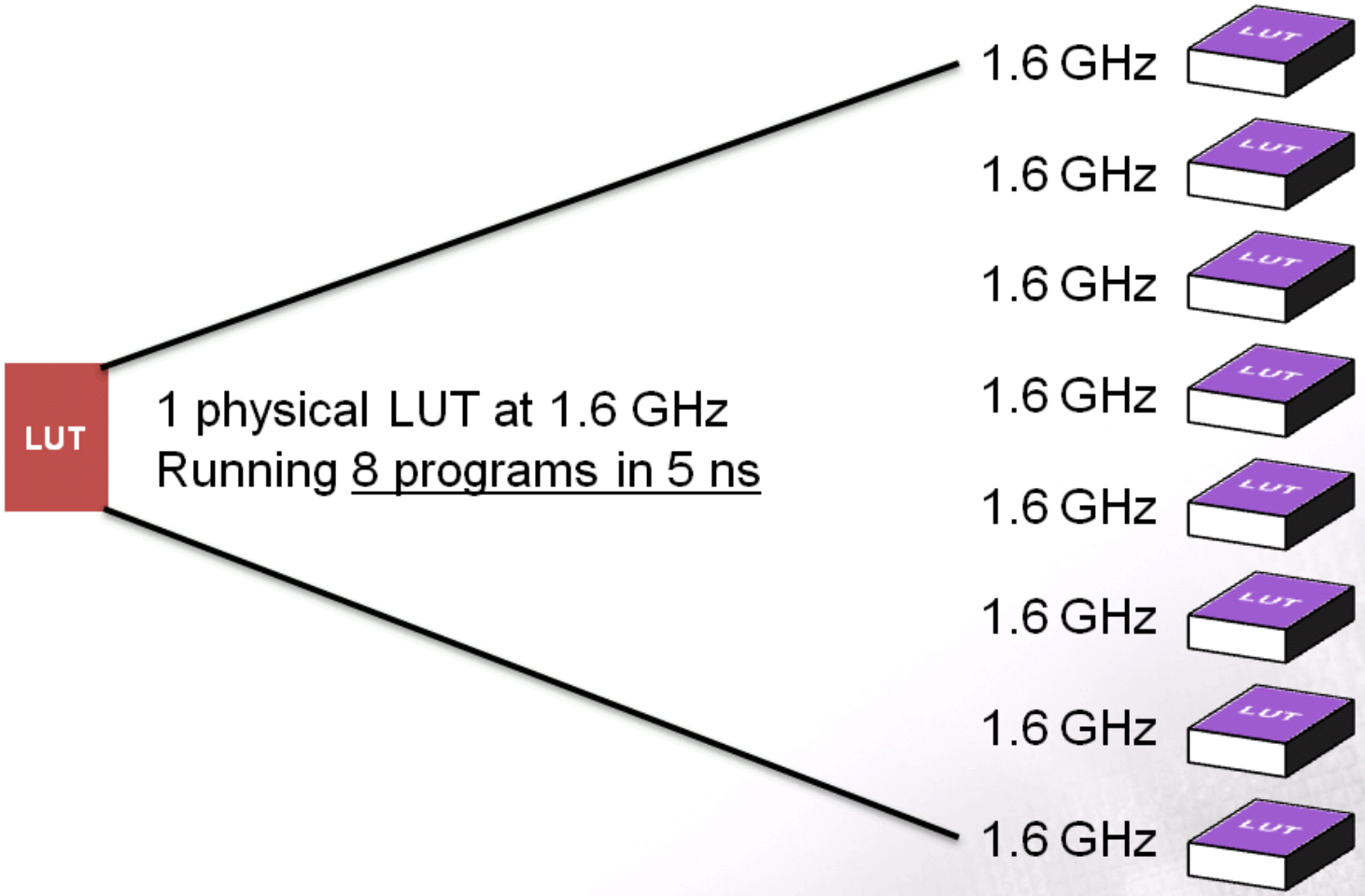
Managing asynchronous domains in Spacetime

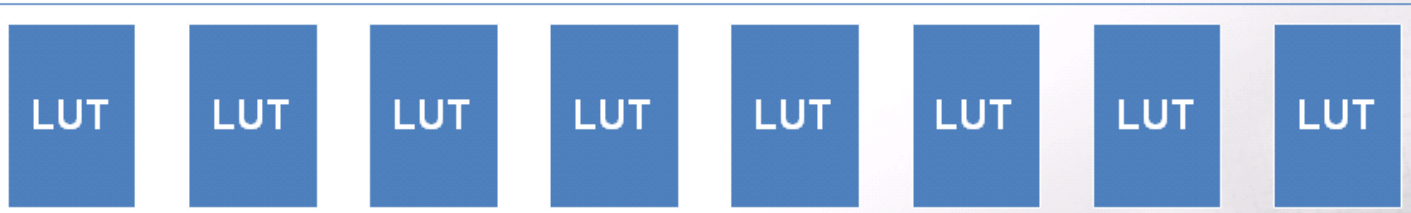
- Asynchronous FIFO
  - Hard asynchronous FIFO in every SerDes and every Parallel I/O
  - Pushes much asynchrony "to the boundary"
- Dual-clocked RAM
  - Many dual-clocked RAMs
  - When single-clocked, act as dual-ported per subcycle
- Chain of state elements (e.g., flops)
  - Need to wait for τ ns
  - On a spatial part, wait time is in quanta of user cycles (e.g., 3.5 – 6 ns)
  - In Spacetime, quantum is duration of fold – e.g., 500 ps
- Thus, latency of asynchronous communication can be much lower in Spacetime
- Advanced: ratiochronous domains can often implement asynchronous domains

Power

- There are four components to Spacetime power
  - Static / Leakage
  - Dynamic / “User”
  - I/O
  - Reconfiguration
- Spacetime can offer much lower leakage power
  - ~1/3 the die area of FPGA for a given function
  - Leakage is becoming dominant at advanced process nodes
- Spacetime can offer somewhat lower dynamic power!

Dynamic power is lower with Spacetime

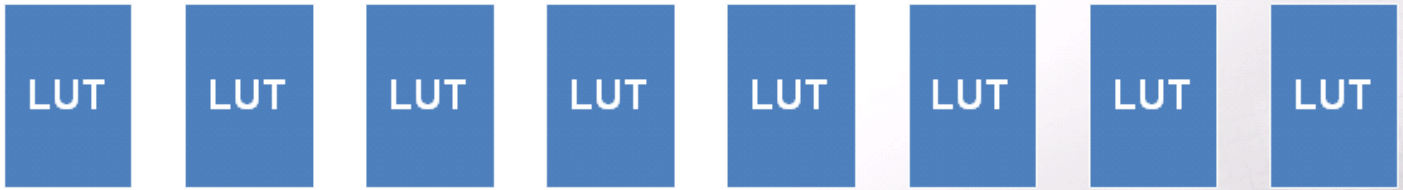




Xitera FPGA:
8 LUTs at 200 MHz
8 programs in 5 ns

Dynamic power is lower with Spacetime

- Same amount of work in the same amount of time
- Same amount of dynamic power
- But…
- Our wires are much shorter: factor of $O(\sqrt{n})$
- Thus, Spacetime dynamic power can be lower than the FPGA's ☺



Xitera FPGA:
8 LUTs at 200 MHz
8 programs in 5 ns

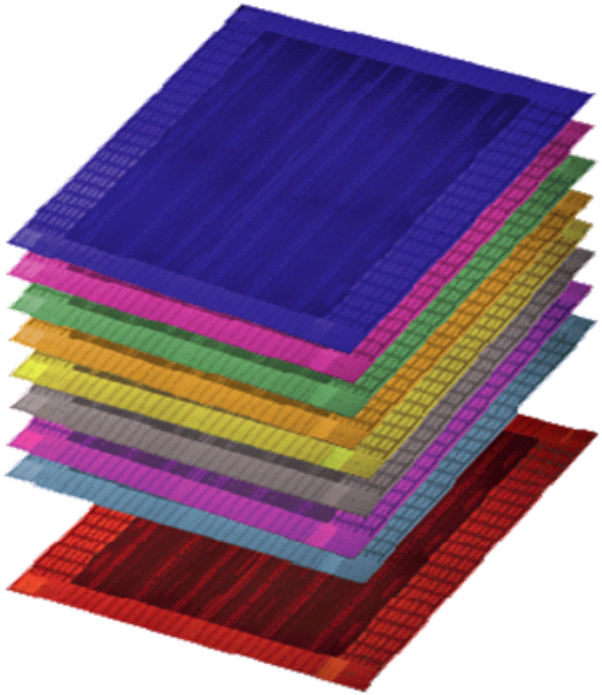
Power

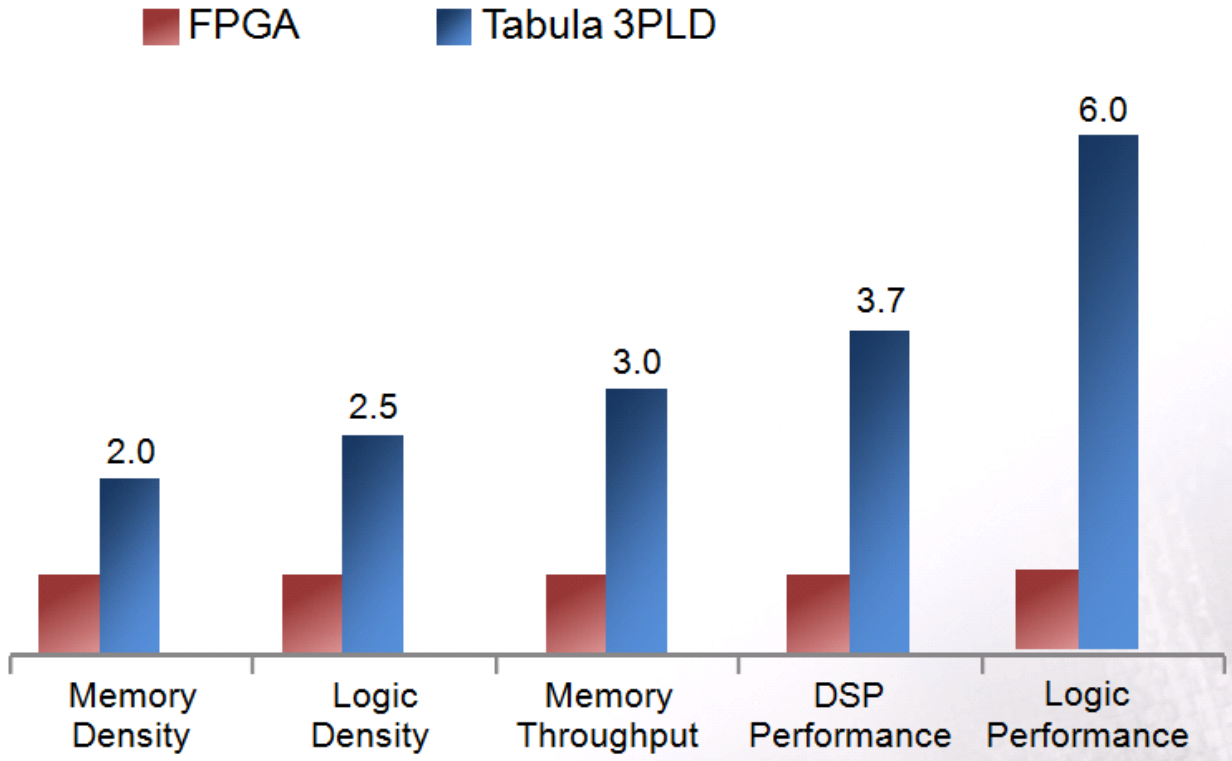
- There are four components to Spacetime power
  - Static / Leakage
  - Dynamic / "User"
  - I/O
  - Reconfiguration
- Spacetime can offer much lower leakage power
  - ~1/3 the die area of FPGA for a given function
  - Leakage is becoming dominant at advanced process nodes
- Spacetime can offer somewhat lower dynamic power!

- Spacetime offers roughly equal I/O power
- Spacetime must (uniquely) pay a reconfiguration tax

- Horse race between leakage + dynamic and reconfiguration

A new category of programmable device





- **2.5x LOGIC DENSITY**
- **2x MEMORY DENSITY**
- **3x MEMORY PORTS**
- **4x DSP PERFORMANCE**
- **6x LOGIC PERFORMANCE**

global sources EE Times Asia "…capability unmatched by traditional FPGAs or CPLDs."

"…unmatched capability and affordability." EE Times europe

global sources EE Times India "…can surpass performance of FPGAs or CPLDs."

Summary

- Spacetime dramatically improves density, performance, memory, DSP
  - Need not cost extra power intrinsically
- Map designs into 3D using the mathematics of special relativity
  - 2 spatial dimensions and 1 time dimension = 3D Minkowski spacetime
- Dynamic reconfiguration is just the first step on a long, fascinating path
  - Significant inventions all along the way
- Many advanced Spacetime techniques
  - Transparent latches in interconnect, extension, rescaling, rescheduling, state as Z wiring, P&R in 3D Minkowski geometry, etc.

- Static scheduling and 3D view opens a new door in computing
- Powerful, alternative perspective from "parallel computing" to exploit massive parallelism of the hardware

- Wait 'til you see what happens next!