Imperial College London

AN INTRODUCTION TO

FIELD PROGRAMMABLE GATE ARRAYS

UK Advanced Instrumentation Course 2022

Andrew W. Rose, Imperial College London

awr01@imperial.ac.uk

WHAT THIS LECTURE IS (AND WHAT IT IS NOT)

- This lecture is a somewhat light-hearted introduction to what FPGAs are, and why they are both brilliant and horrible
- Unfortunately, 1 hour does not give time to go into any depth – several months of hands-on work would be more realistic

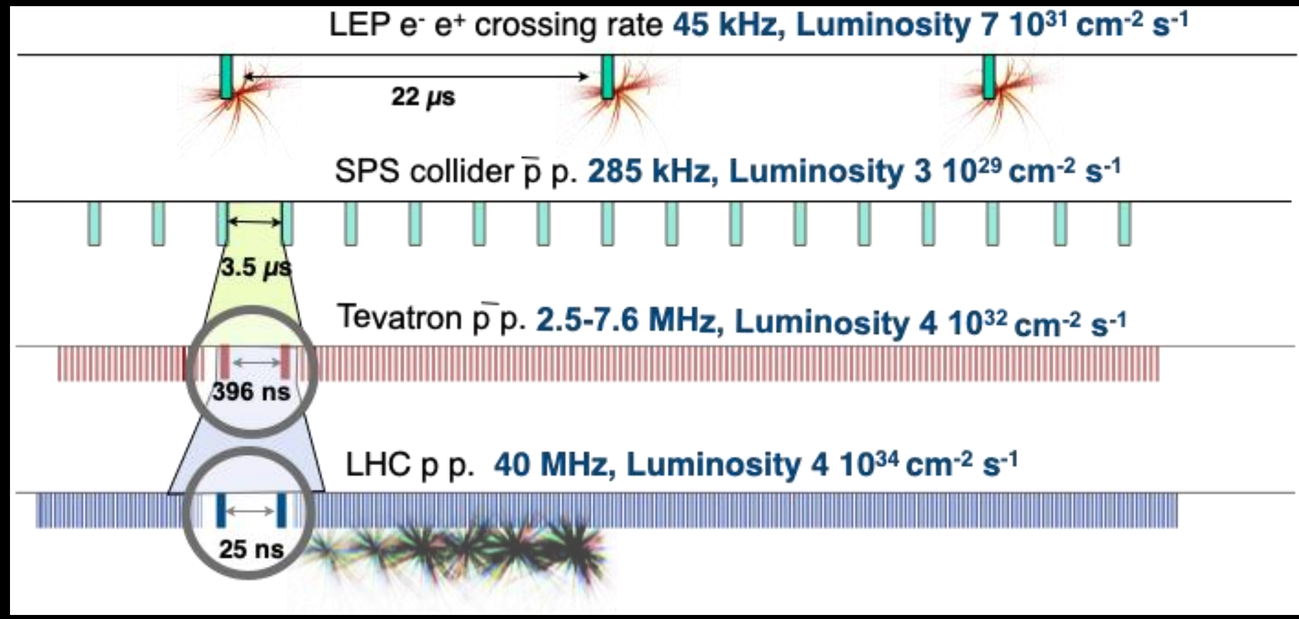
RECALL FROM TRIGGER & DAQ LECTURES

RECALL FROM TRIGGER & DAQ LECTURES

So… I should probably justify those statements…

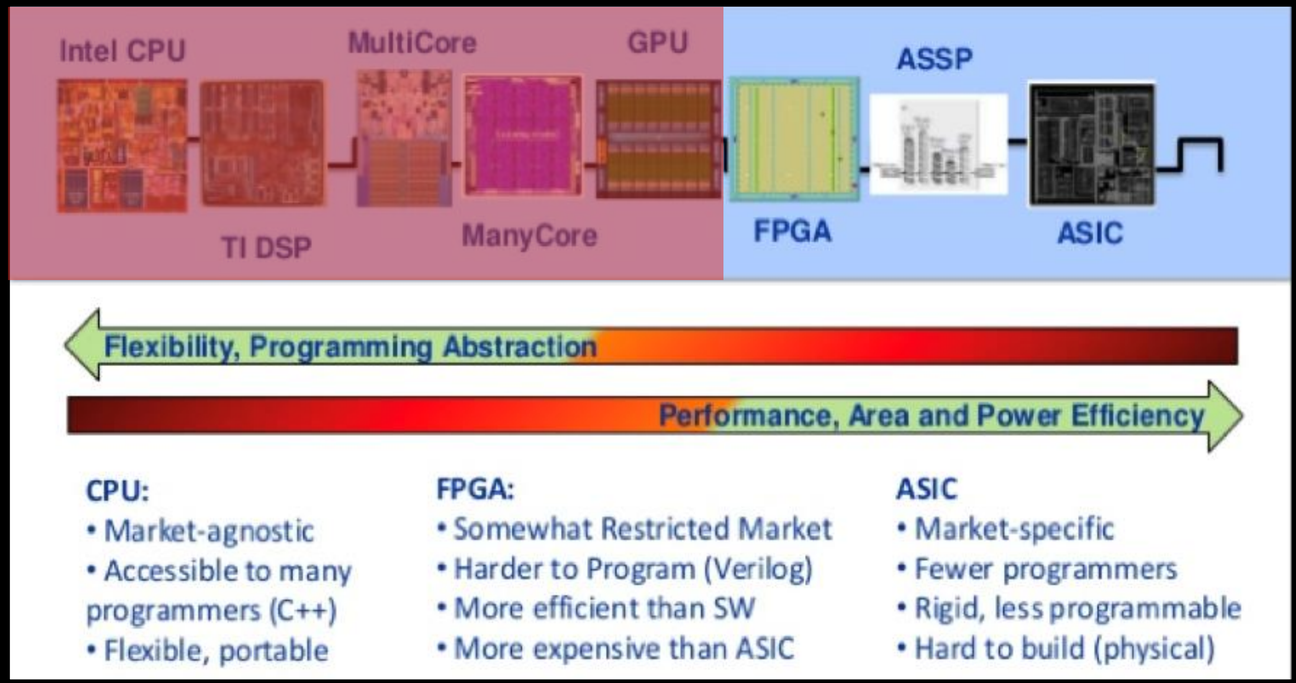
A NOTE ON TIMESCALES

- At 40MHz BX rate, a 4GHz CPU could perform 100 CPU operations (not enough to be useful) before having to pass to the next core
- Compare that to the O(10M) detector channels
- What technology can we use?



PROGRAMMABLE DEVICES

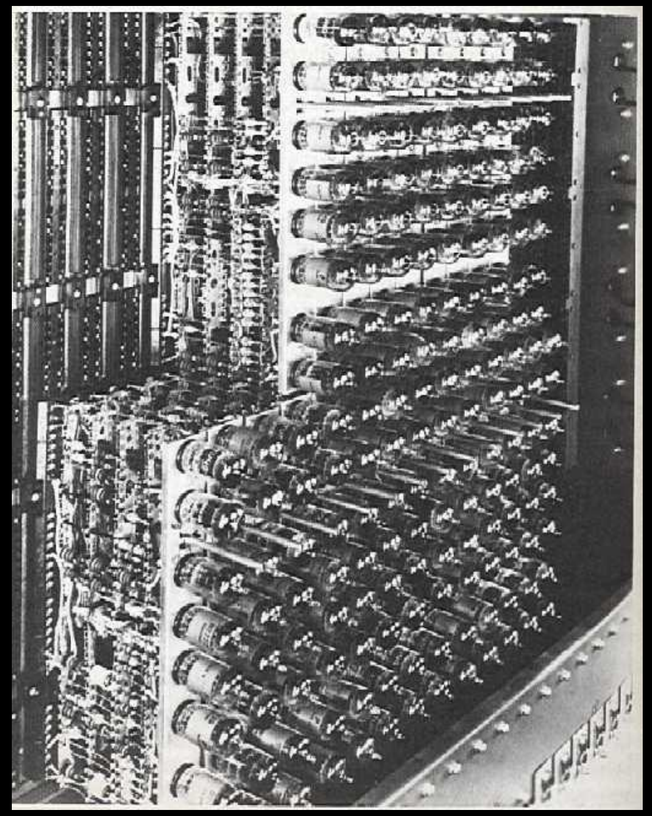
- Application-specific integrated circuits (ASICs): optimised for fast processing, design encoded into silicon
- "Programmable ASICS": Field-programmable gate arrays (FPGAs)



AN ASIDE: THE HISTORY OF ELECTRONICS

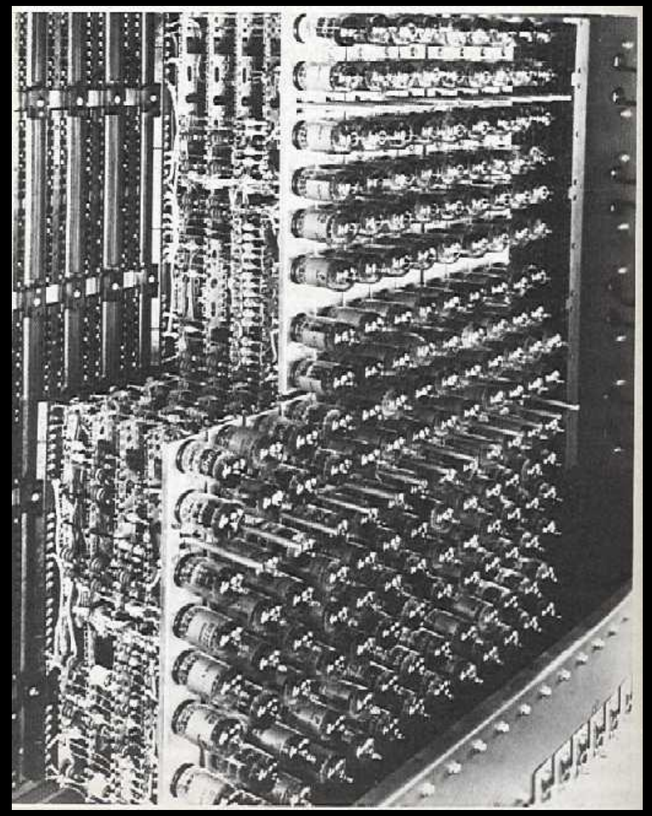
- Digital electronics really started with the advent of the thermionic valve (colloquially, the “vacuum tube”)

THE HISTORY OF ELECTRONICS



Valve
transistors

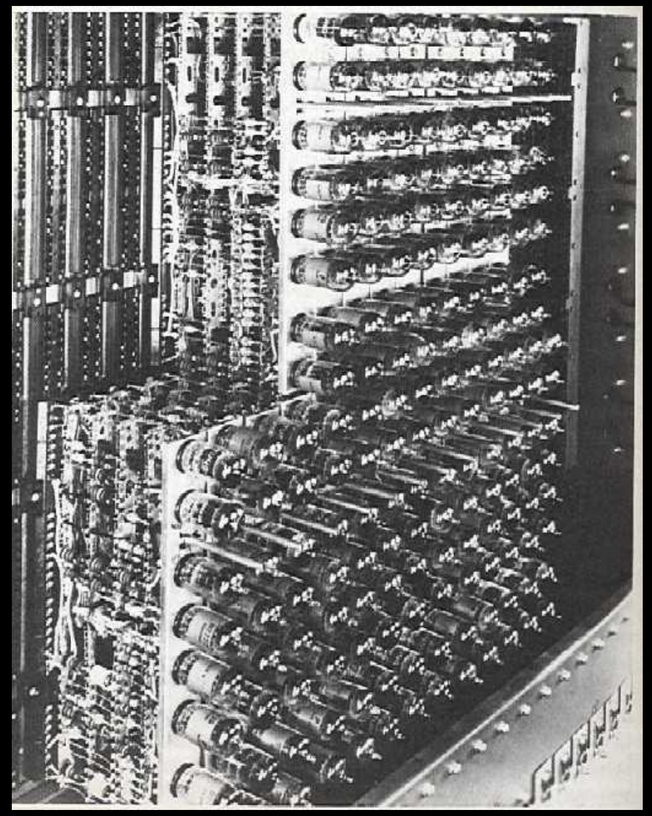
THE HISTORY OF ELECTRONICS



Valve transistors

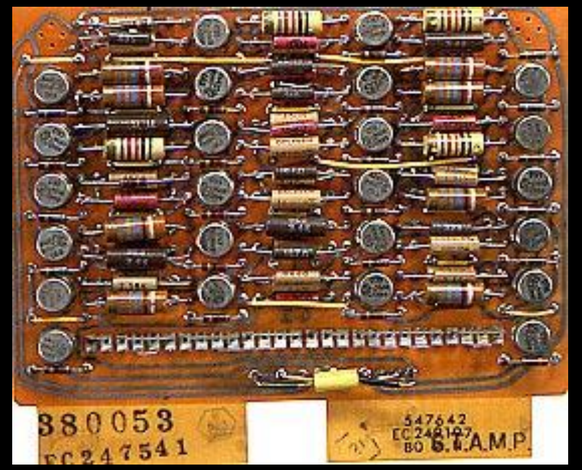
First solid-state transistors

THE HISTORY OF ELECTRONICS



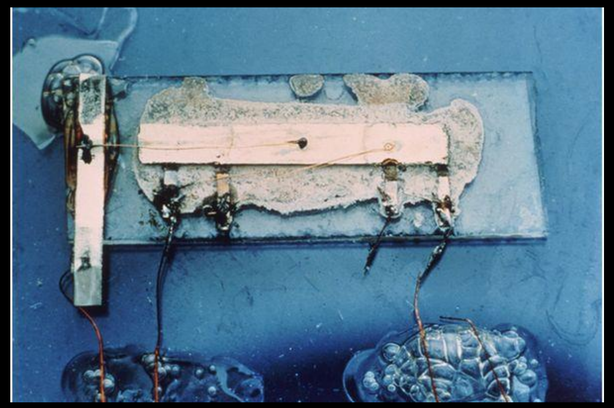
Valve transistors

First solid-state transistors



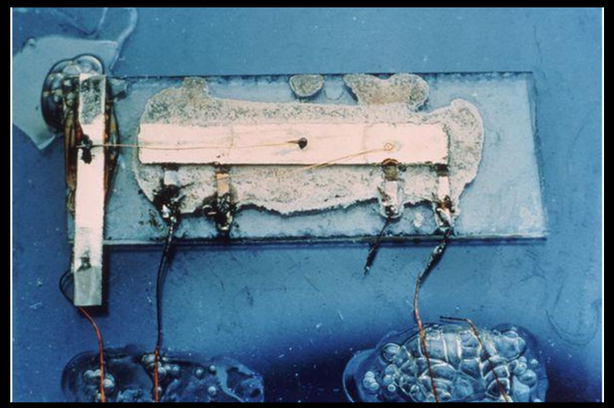
Solid-state transistors

THE HISTORY OF ELECTRONICS

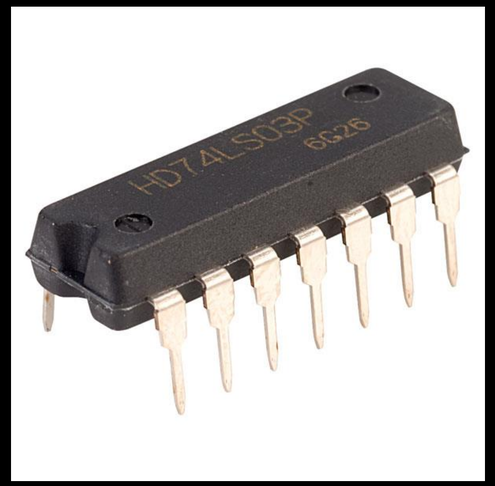


First multi-transistor silicon

THE HISTORY OF ELECTRONICS

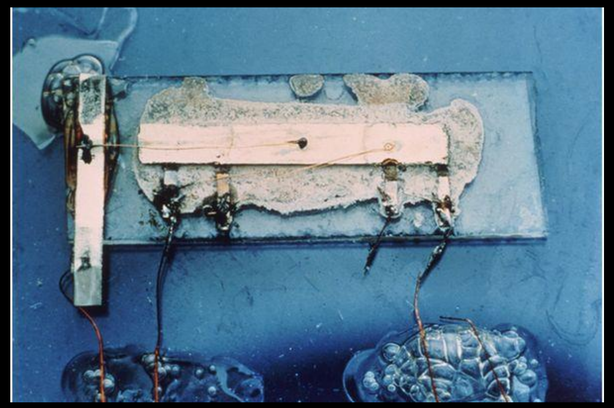


First multi-transistor silicon

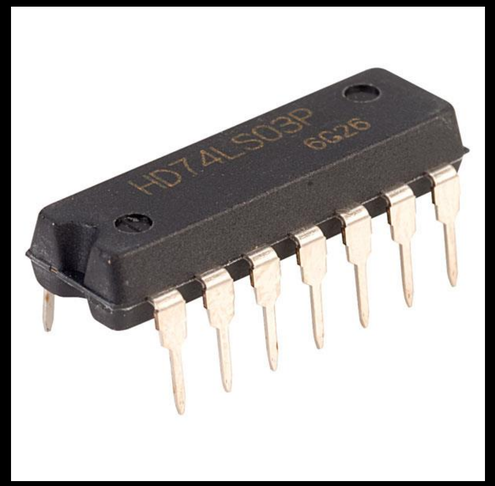


Packaged Logic

THE HISTORY OF ELECTRONICS



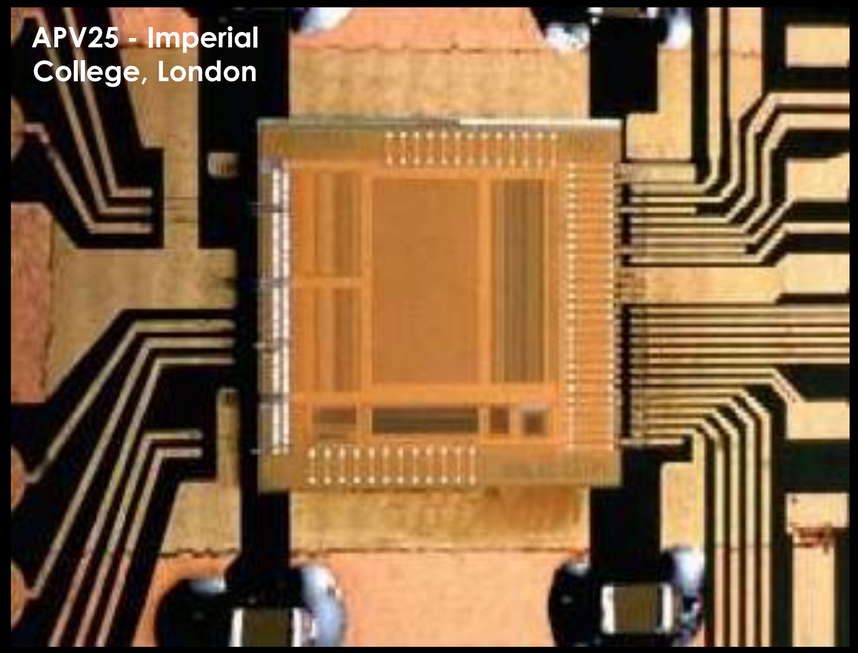
First multi-transistor silicon



Packaged Logic

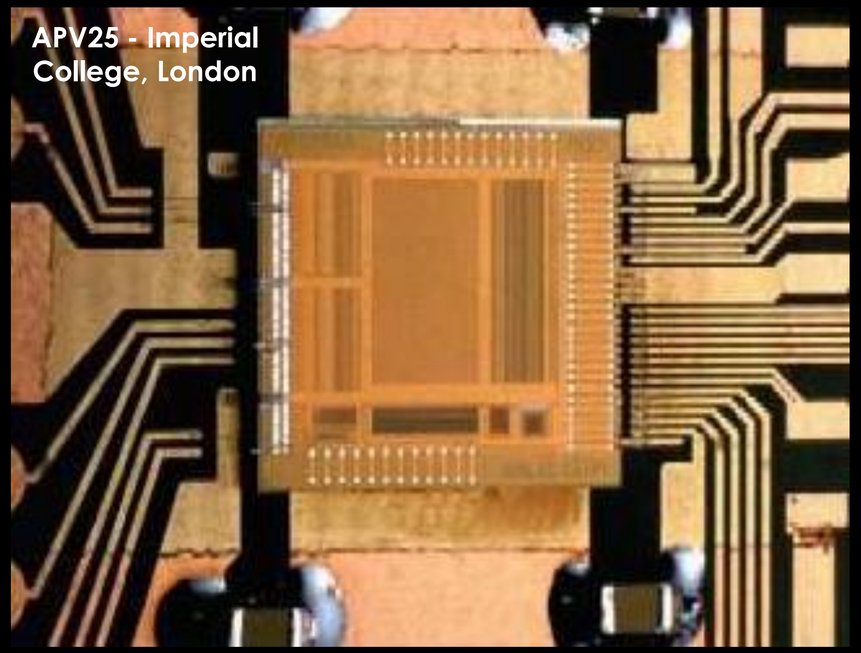
“Mini” processor board

ASICS

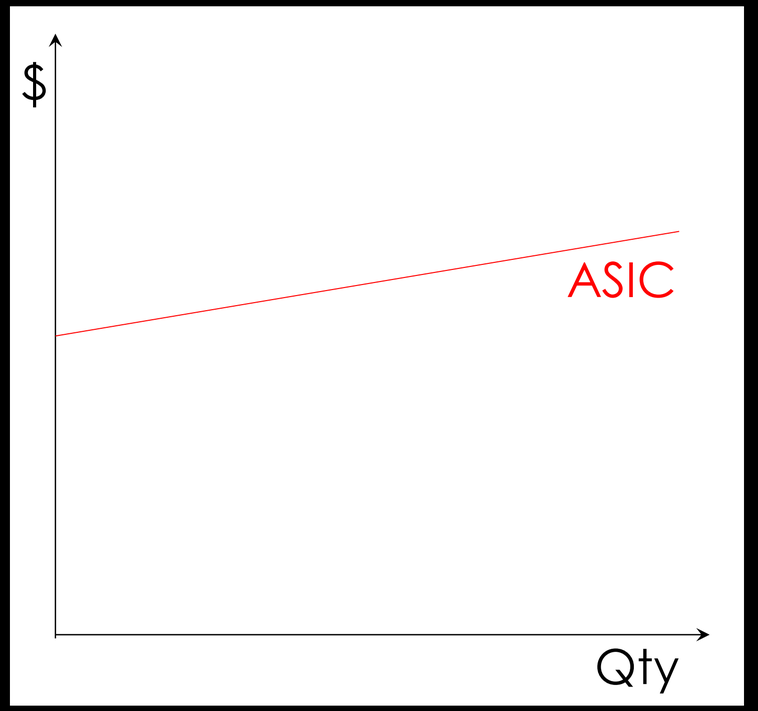


Application Specific Integrated Circuit (ASIC)

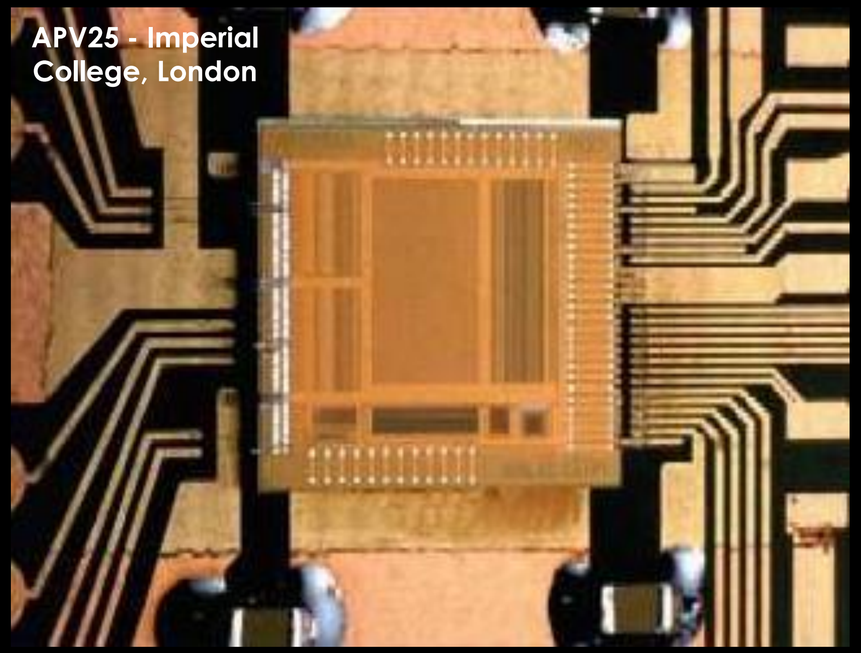
ASICS



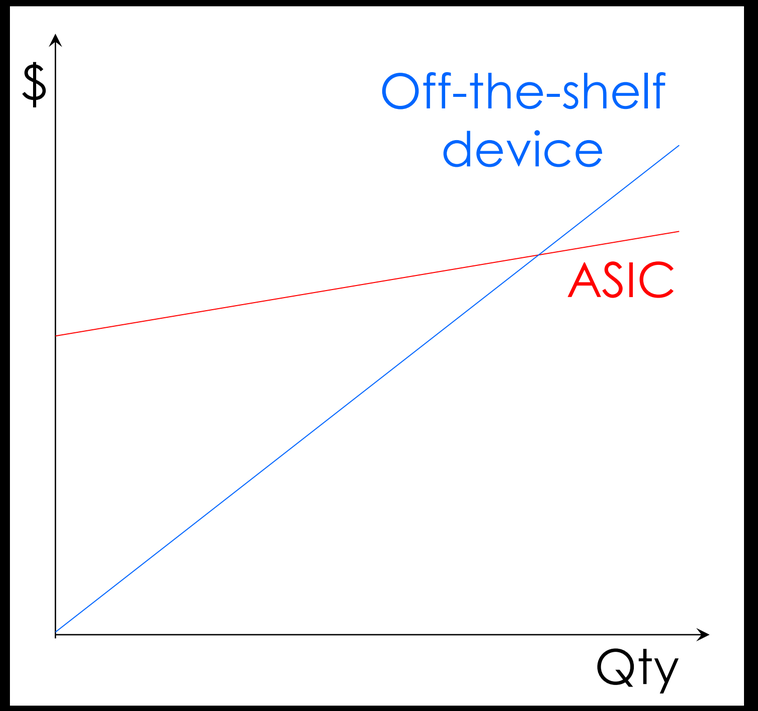
Application Specific Integrated Circuit (ASIC)



ASICS



Application Specific Integrated Circuit (ASIC)



TWO PHILOSOPHIES: SPACE VS. TIME

Have each operation performed by
dedicated logic
and
let signal propagate as a wave
through the logic

Combinatorial

TWO PHILOSOPHIES: SPACE VS. TIME

Have each operation performed by
dedicated logic
and
let signal propagate as a wave
through the logic

Combinatorial

Fast, but messy, hard to understand, not scalable and low throughput

TWO PHILOSOPHIES: SPACE VS. TIME

Have each operation performed by
dedicated logic
and
do that same operation
on every clock cycle

Slightly slower, but clean, easy to understand, scalable and high throughput

Pipelined/Parallel

TWO PHILOSOPHIES: SPACE VS. TIME

Have each operation performed by
dedicated logic
and
do that same operation
on every clock cycle

Pipelined/Parallel

Have each operation performed by
the same logic
performing
a different operation
on every clock cycle

Sequential

TWO PHILOSOPHIES: SPACE VS. TIME

Have each operation performed by
dedicated logic
and
do that same operation
on every clock cycle

Pipelined/Parallel

Have each operation performed by
the same logic
performing
a different operation
on every clock cycle

Sequential

A debate as old as
electronic computing itself

TWO PHILOSOPHIES: SPACE VS. TIME

Have each operation performed by
dedicated logic
and
do that same operation
on every clock cycle

Pipelined/Parallel

Have each operation performed by
the same logic
performing
a different operation
on every clock cycle

Sequential

"The parallel approach to computing does require that some original thinking be done about numerical analysis and data management in order to secure efficient use.

In an environment which has represented the absence of the need to think as the highest virtue, this is a decided disadvantage"

Daniel Slotnick, 1967

AND THE STORY DIVERGES…

Pipelined/Parallel

Programmable Array Logic

Pack entire logic circuits in a chip

Sequential

Microprocessor

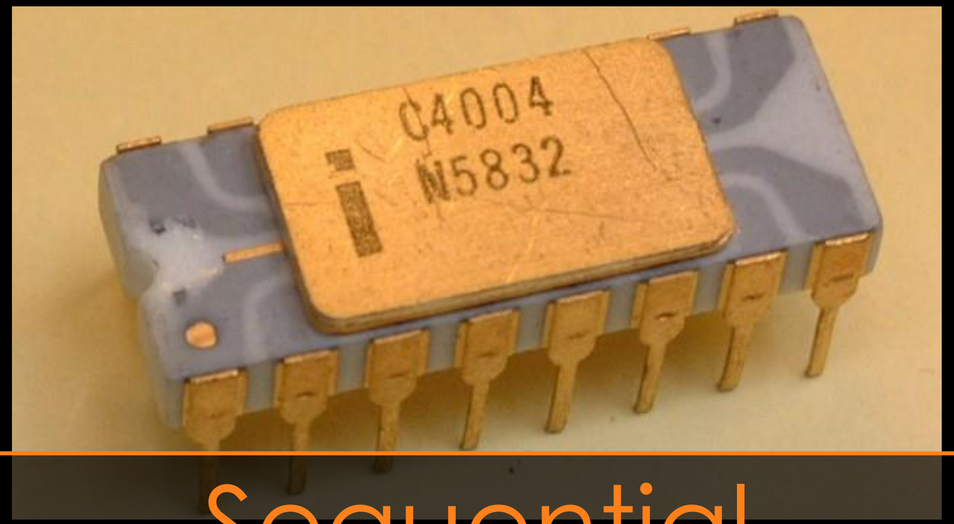
Perform all logical operations in one location, but sequentially

AND THE STORY DIVERGES…

Pipelined/Parallel

Programmable Array Logic

Pack entire logic circuits in a chip



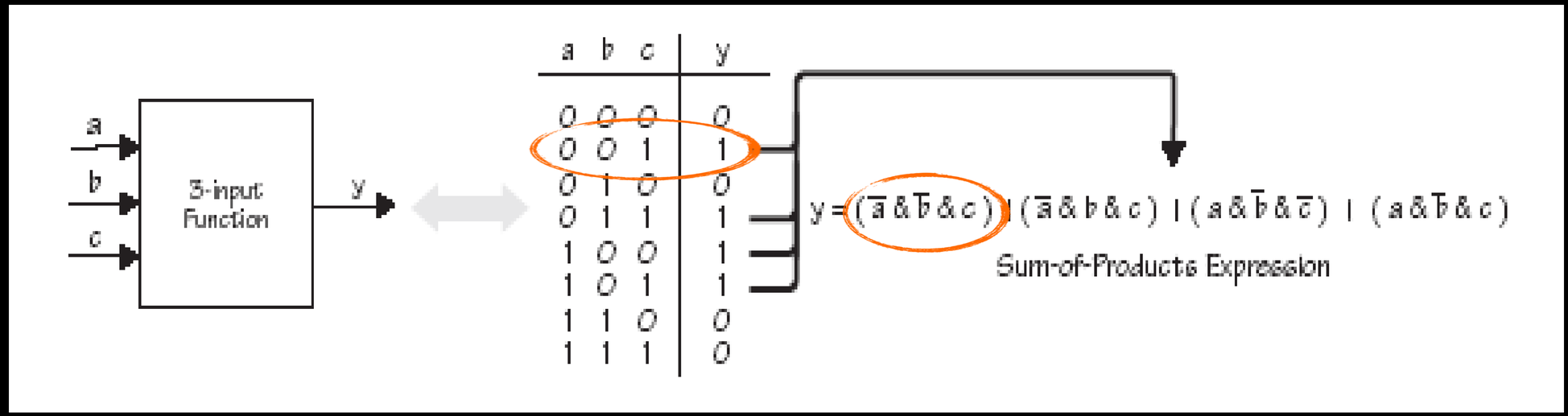
Sequential

Microprocessor

Perform all logical operations in one location, but sequentially

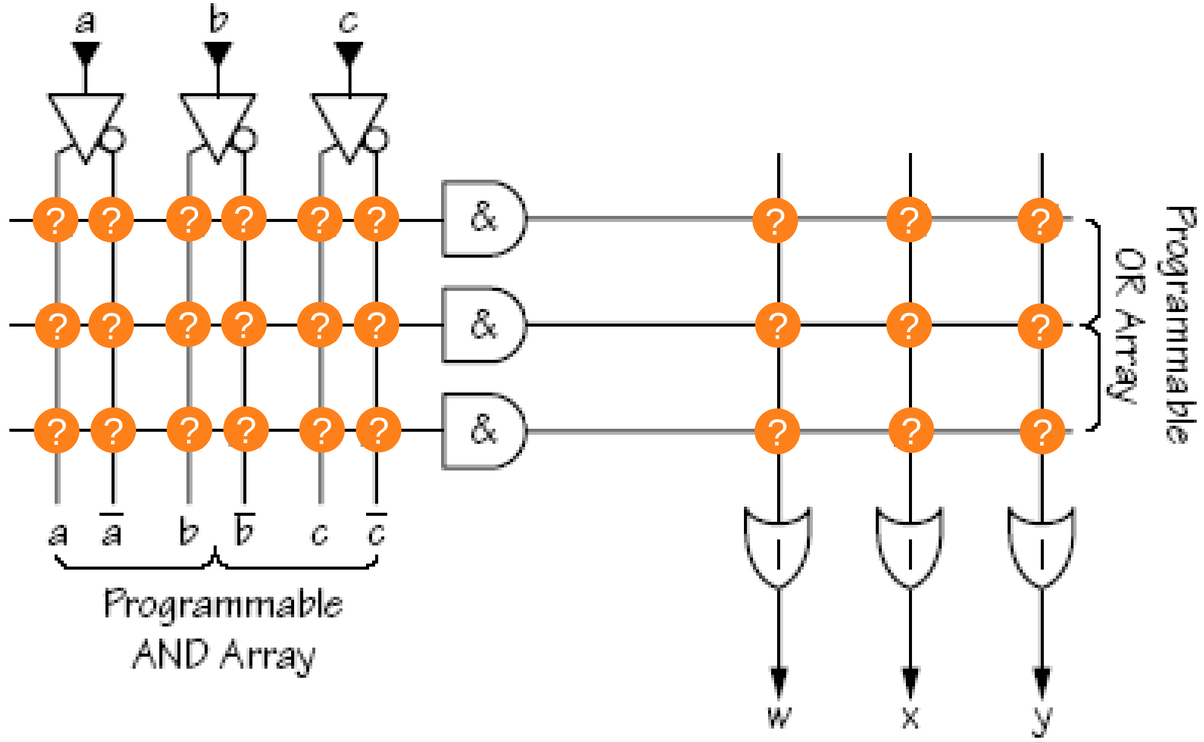
Limited further discussion of microprocessors

SUM-OF-PRODUCTS THEOREM



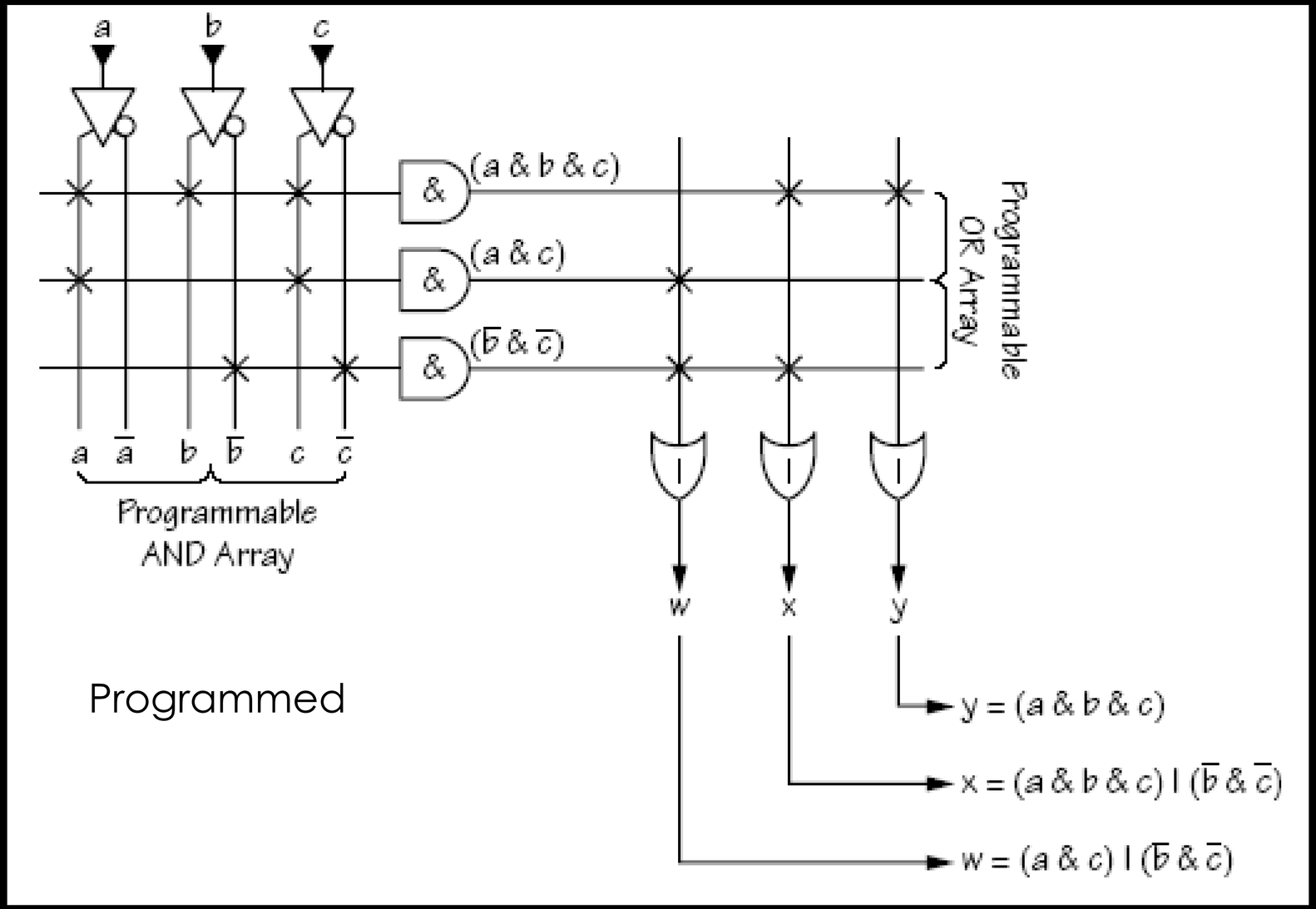
- Any Boolean operation may be expressed as
  the OR of AND operations (Sum of products form)
- Or
  the AND of OR operations (Product of sums form)

PROGRAMMABLE LOGIC DEVICES (PLDS)



Unprogrammed

PROGRAMMABLE LOGIC DEVICES (PLDS)

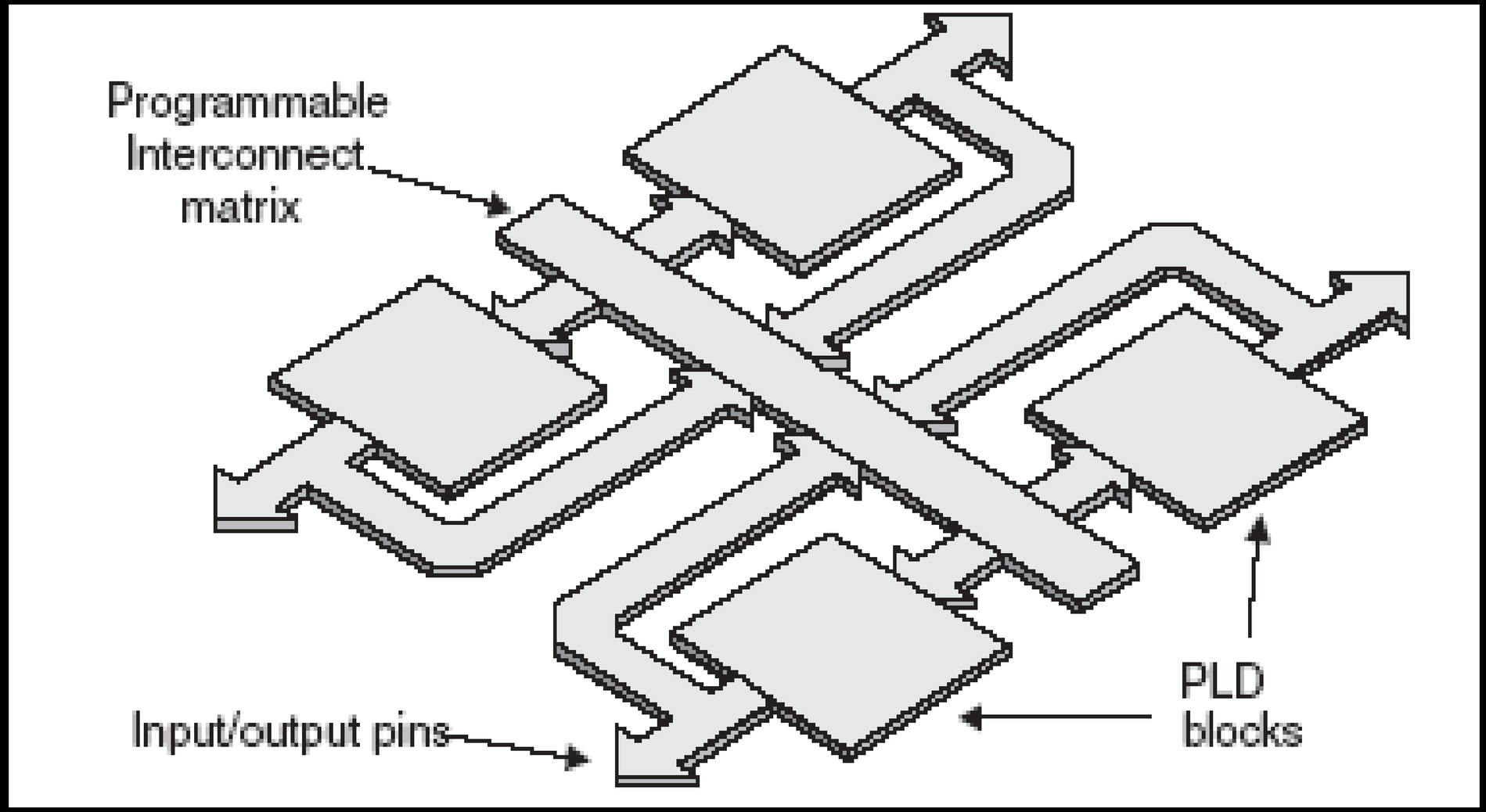


PROGRAMMABLE LOGIC DEVICES (PLDS)

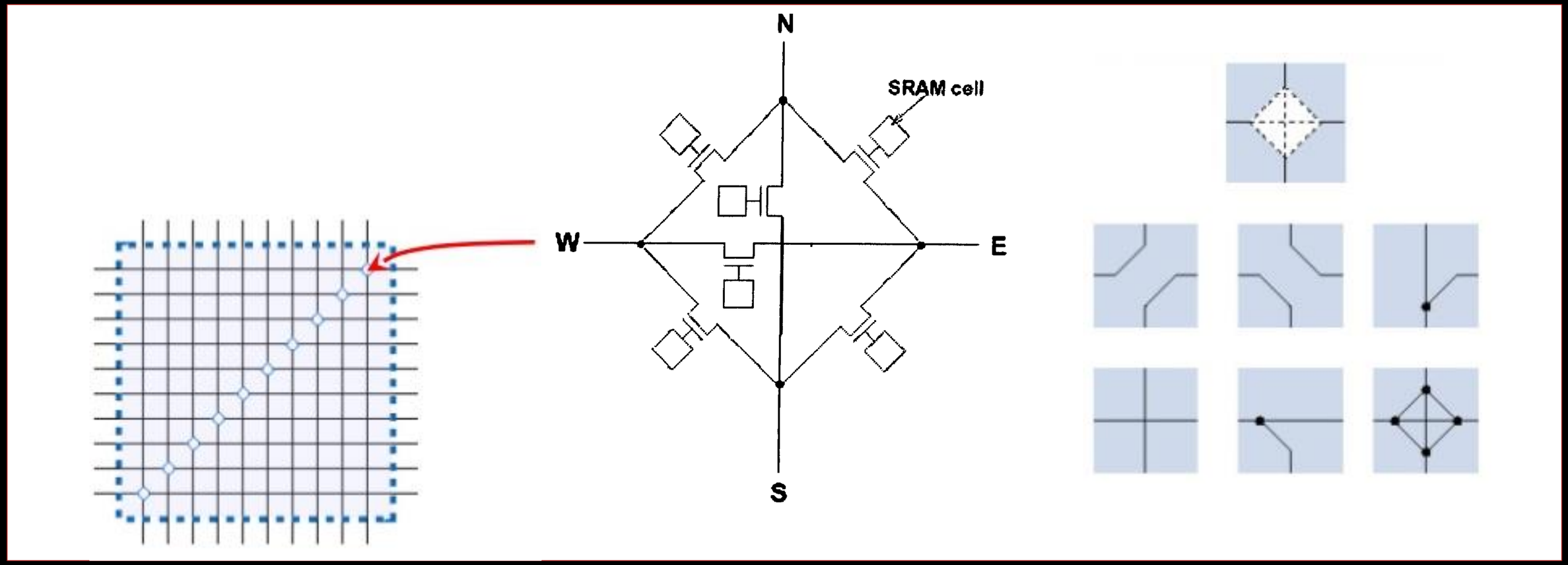
- Originally one-time programmable
- Later field reprogrammable

- What did people do? Build boards with many PLDs…

COMPLEX PLDS (CPLDS)



PROGRAMMABLE INTERCONNECT MATRIX



AN ALTERNATIVE APPROACH

- Why bother with the complexity of the PLD cell?

- Replace the PLD cell with a simple SRAM:
  - Data-in becomes the “address”
  - Outputs the preloaded value for a given input

AN ALTERNATIVE APPROACH

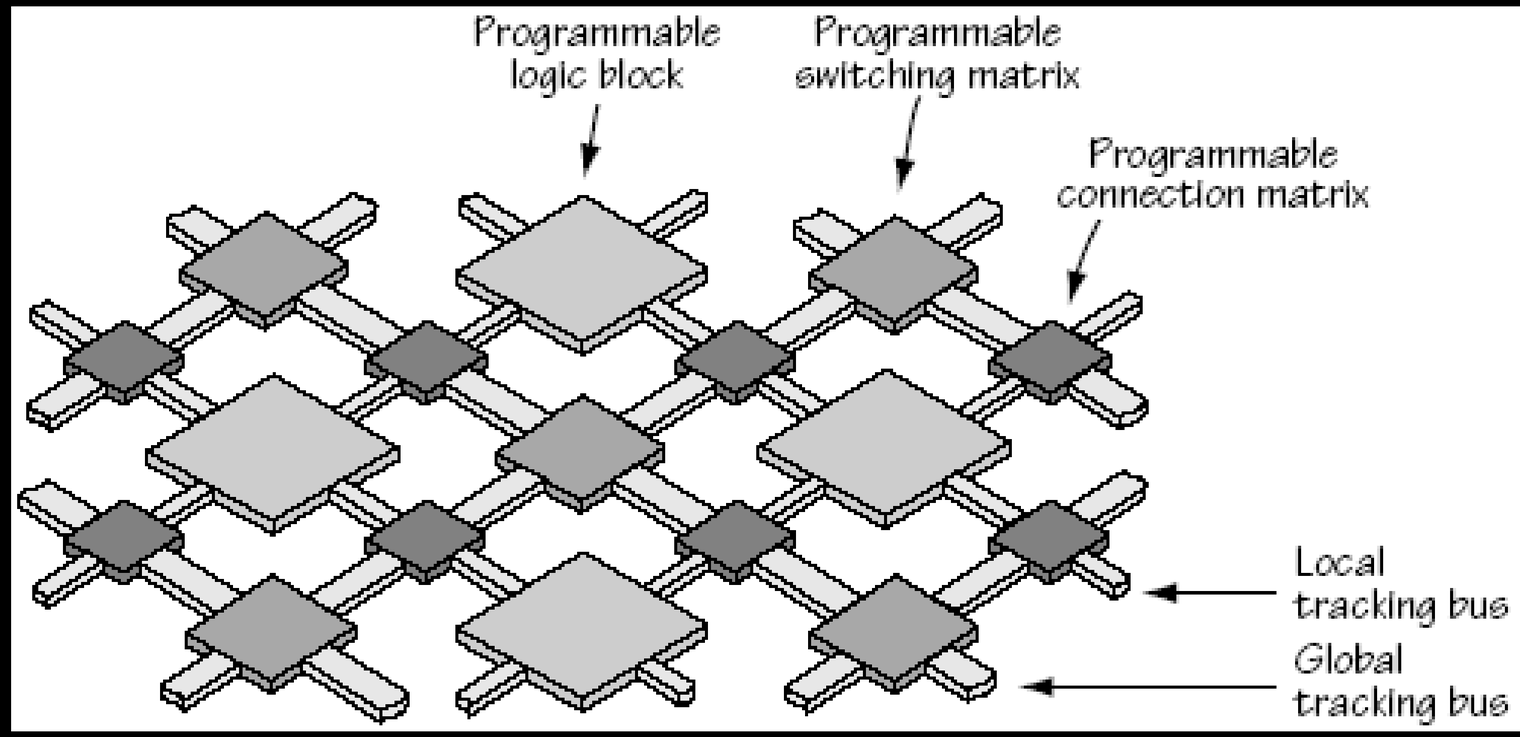
- Why bother with the complexity of the PLD cell?

- Replace the PLD cell with a simple SRAM:
  - Data-in becomes the “address”
  - Outputs the preloaded value for a given input

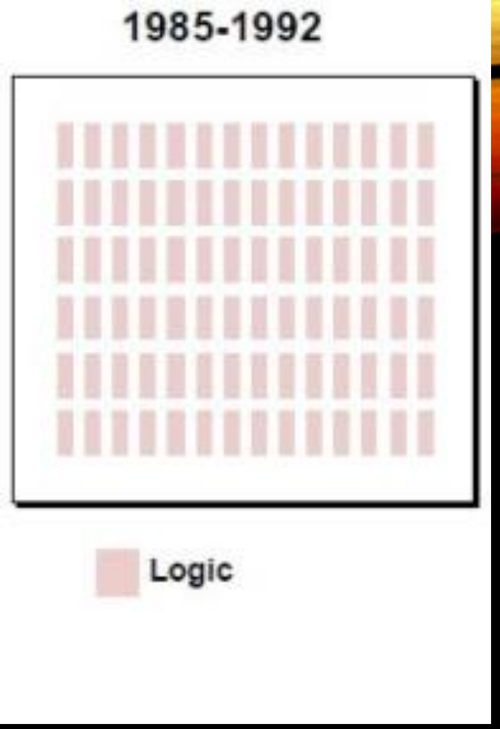
The Field Programmable Gate Array (FPGA)

FIELD PROGRAMMABLE GATE ARRAYS (FPGAS)

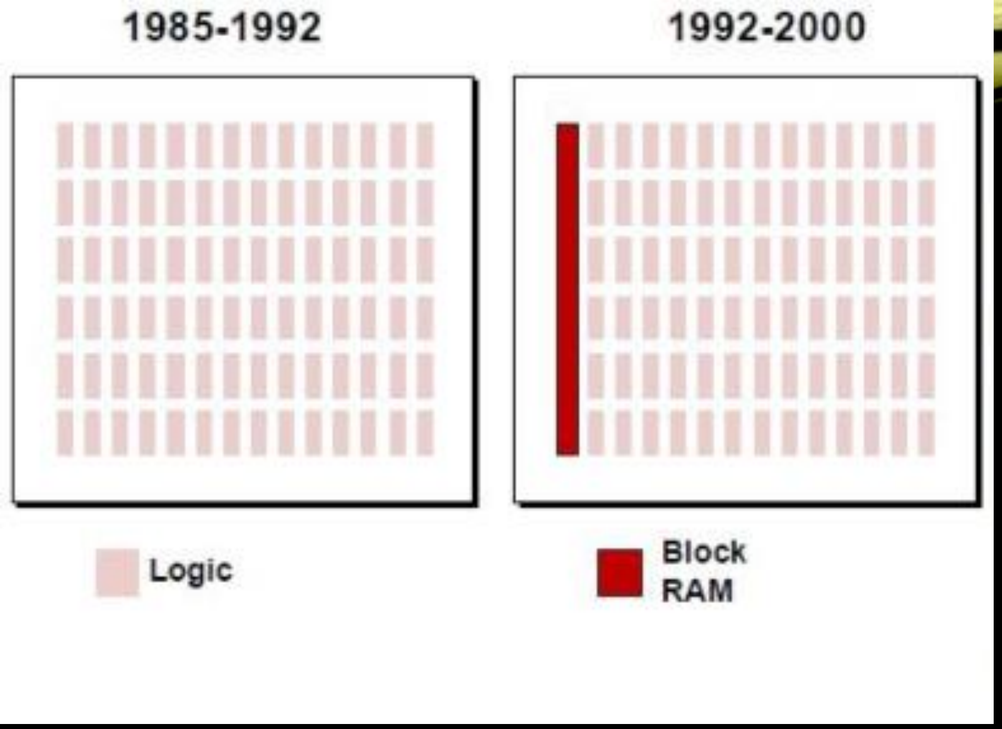
- ‘Simple’ Programmable Logic Blocks
- Massive Fabric of Programmable Interconnects



EVOLUTION OF FEATURES IN FPGAS

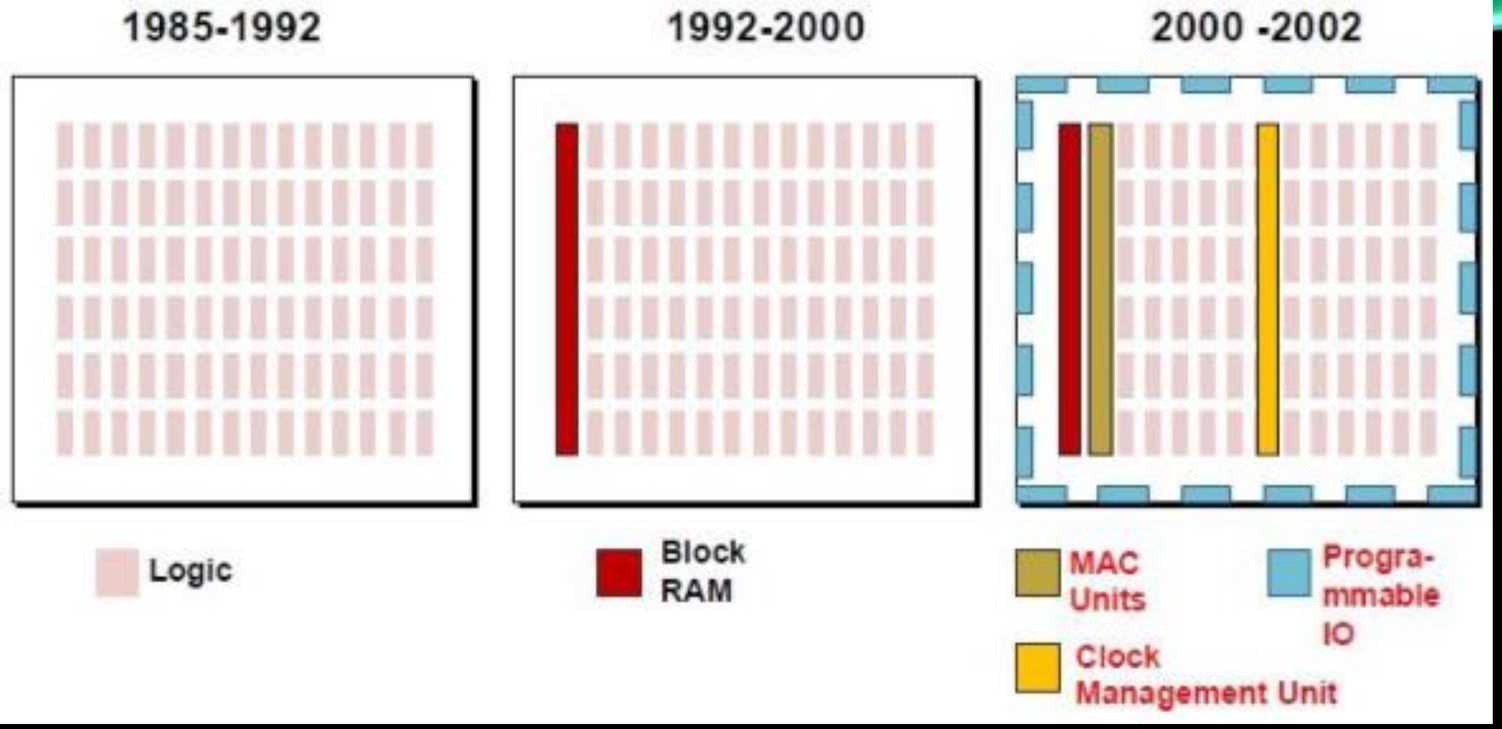


EVOLUTION OF FEATURES IN FPGAS



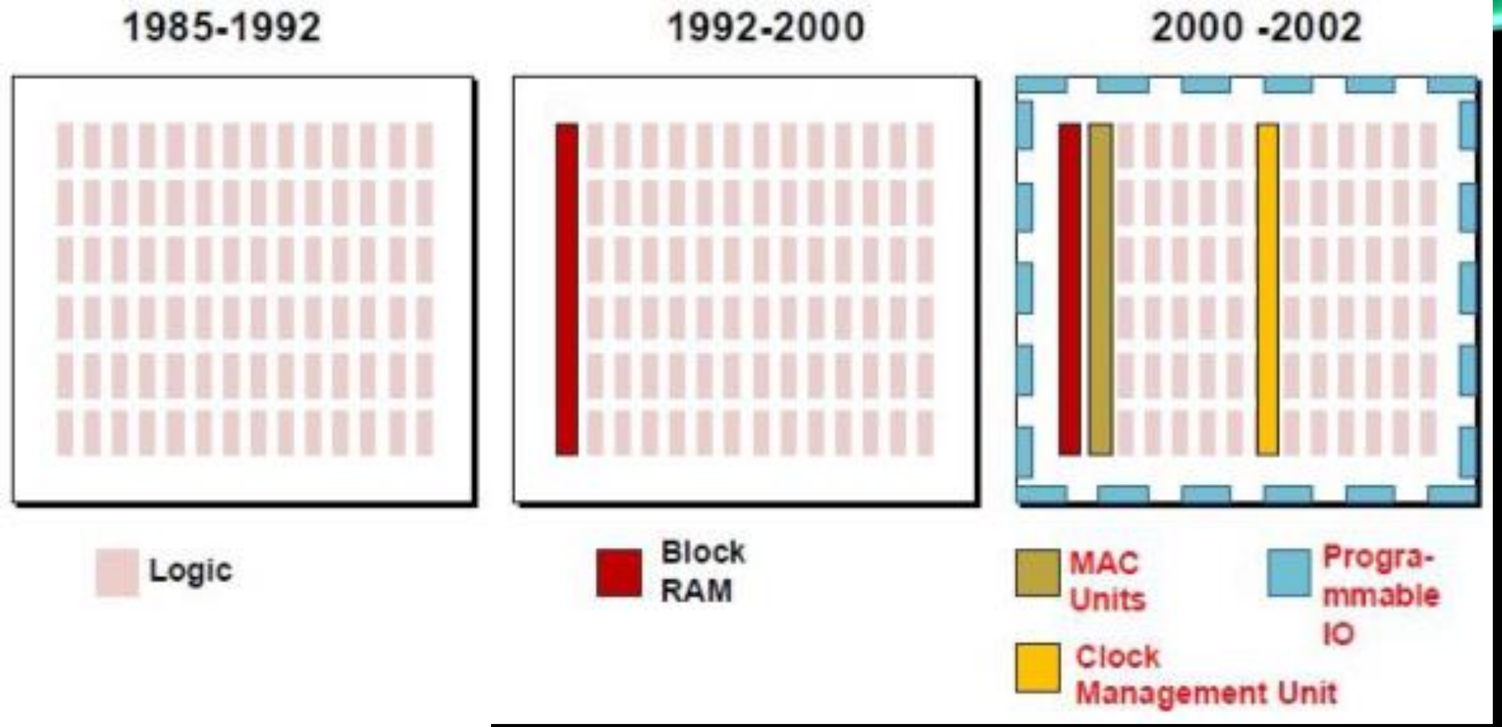
Who wants to waste all the LUTs as RAM?

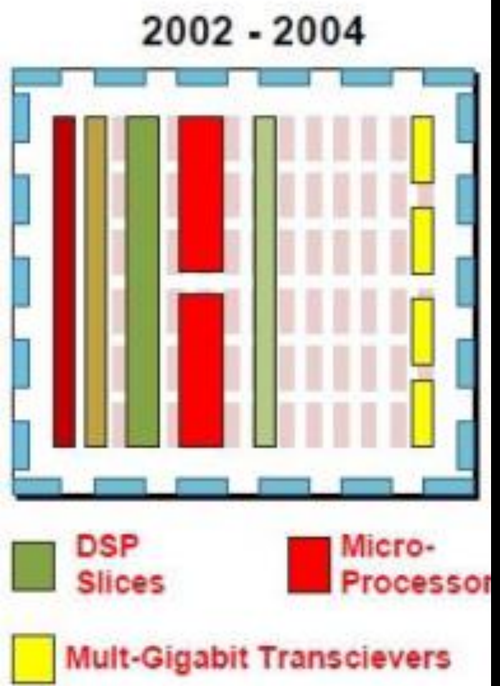
EVOLUTION OF FEATURES IN FPGAS



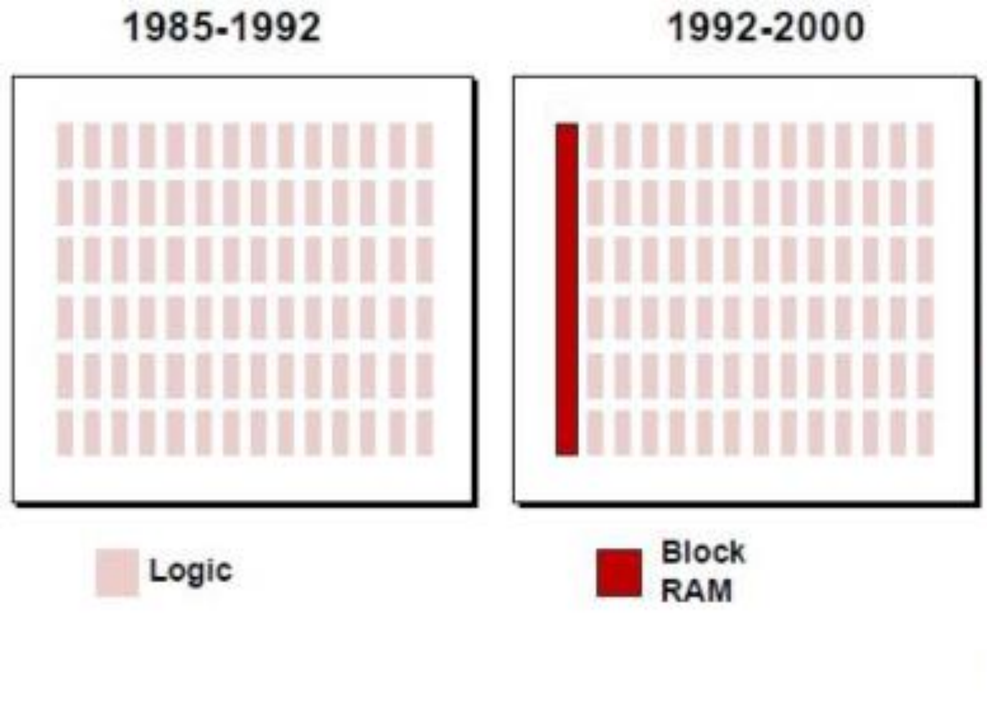
Who wants to waste all the LUTs for multiplication?
Big chips need dedicated clocking!

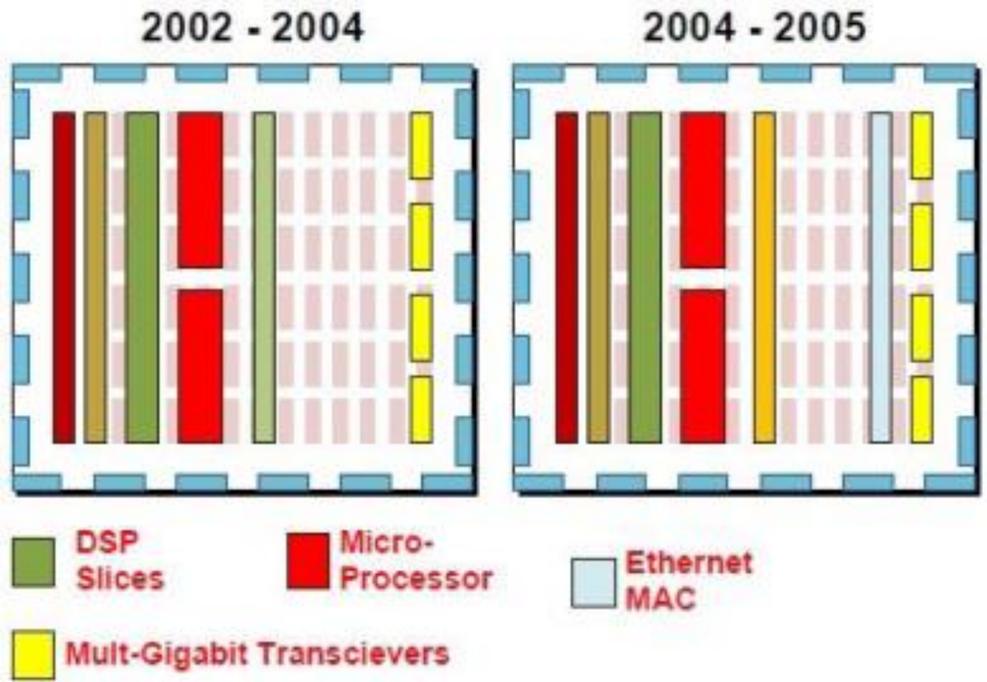
EVOLUTION OF FEATURES IN FPGAS





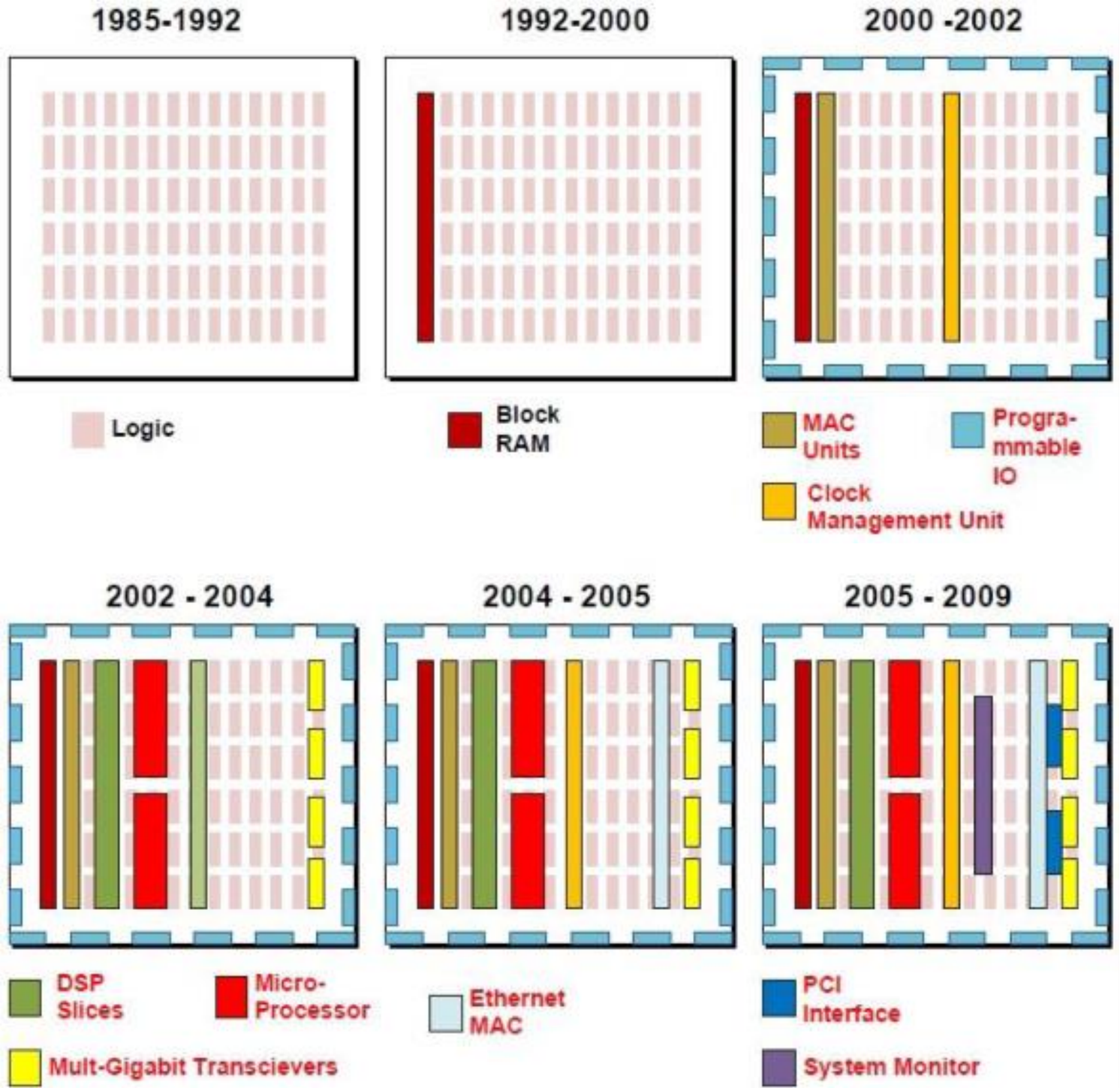
EVOLUTION OF FEATURES IN FPGAS





Who wants to waste LUTs AND re-invent industry-standard blocks?

EVOLUTION OF FEATURES IN FPGAS



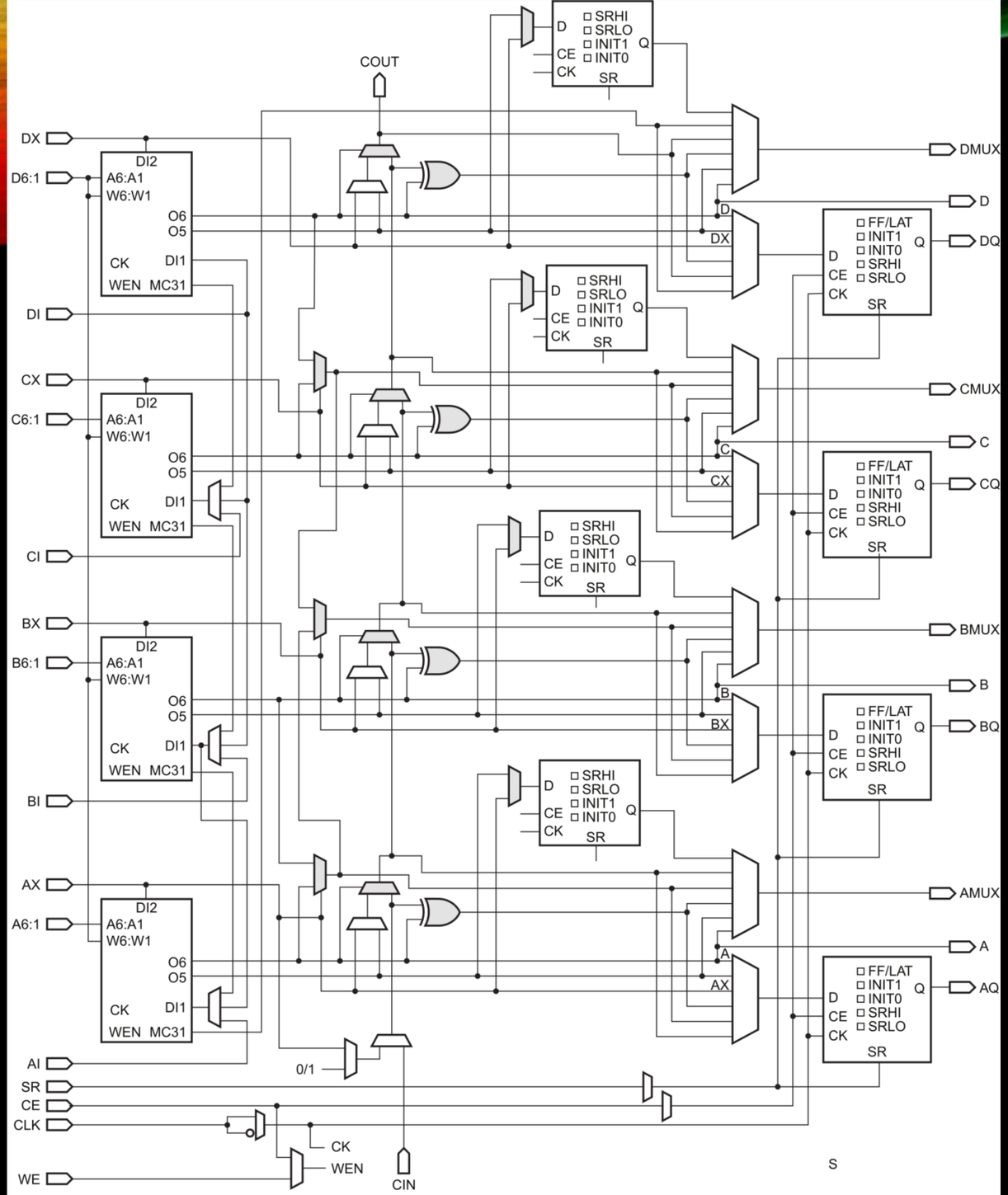
Who wants to waste LUTs AND re-invent industry-standard blocks?

A NOTE ON I/O

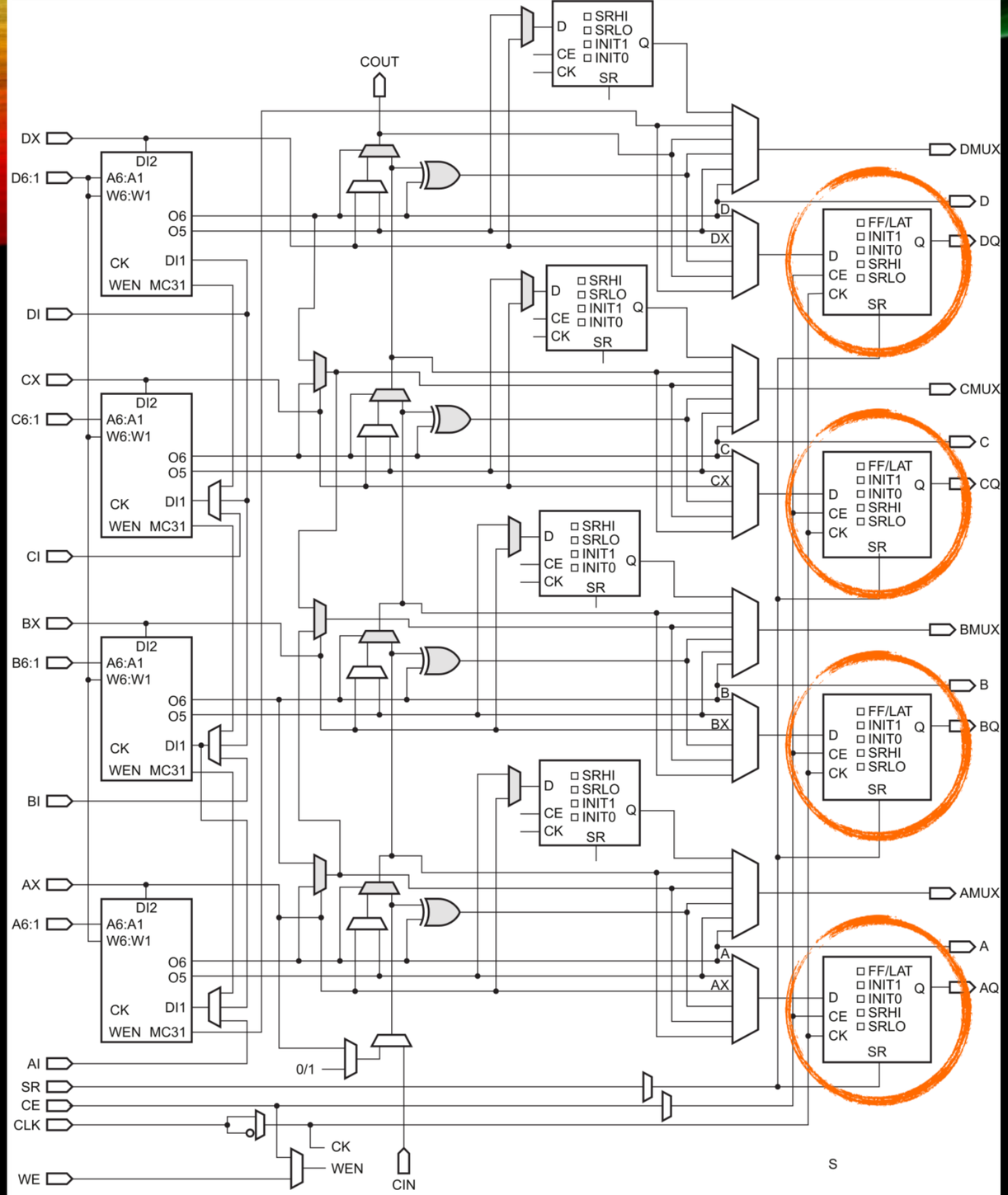
- Traditionally, many hundreds of general-purpose pins (Gen I/O) up to a few hundred MHz
- Latest generation Gen I/O up to 1.8Gbps
- Programmable logic standards

- Since 2002, FPGAs have been adding dedicated Multi-gigabit transceivers
- Arms race - Ever more and ever faster

COMBINATORIAL LOGIC BLOCK

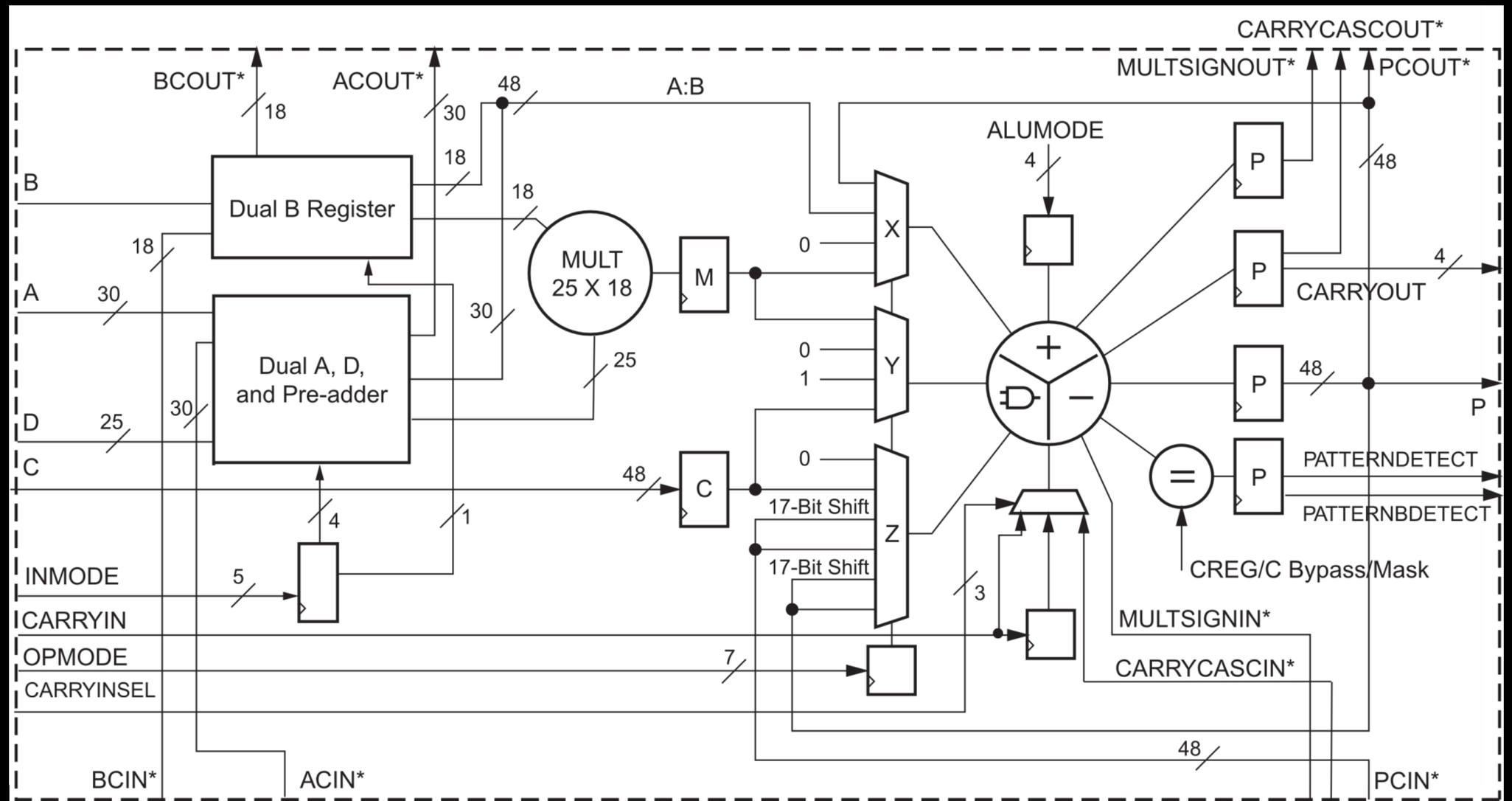


COMBINATORIAL LOGIC BLOCK

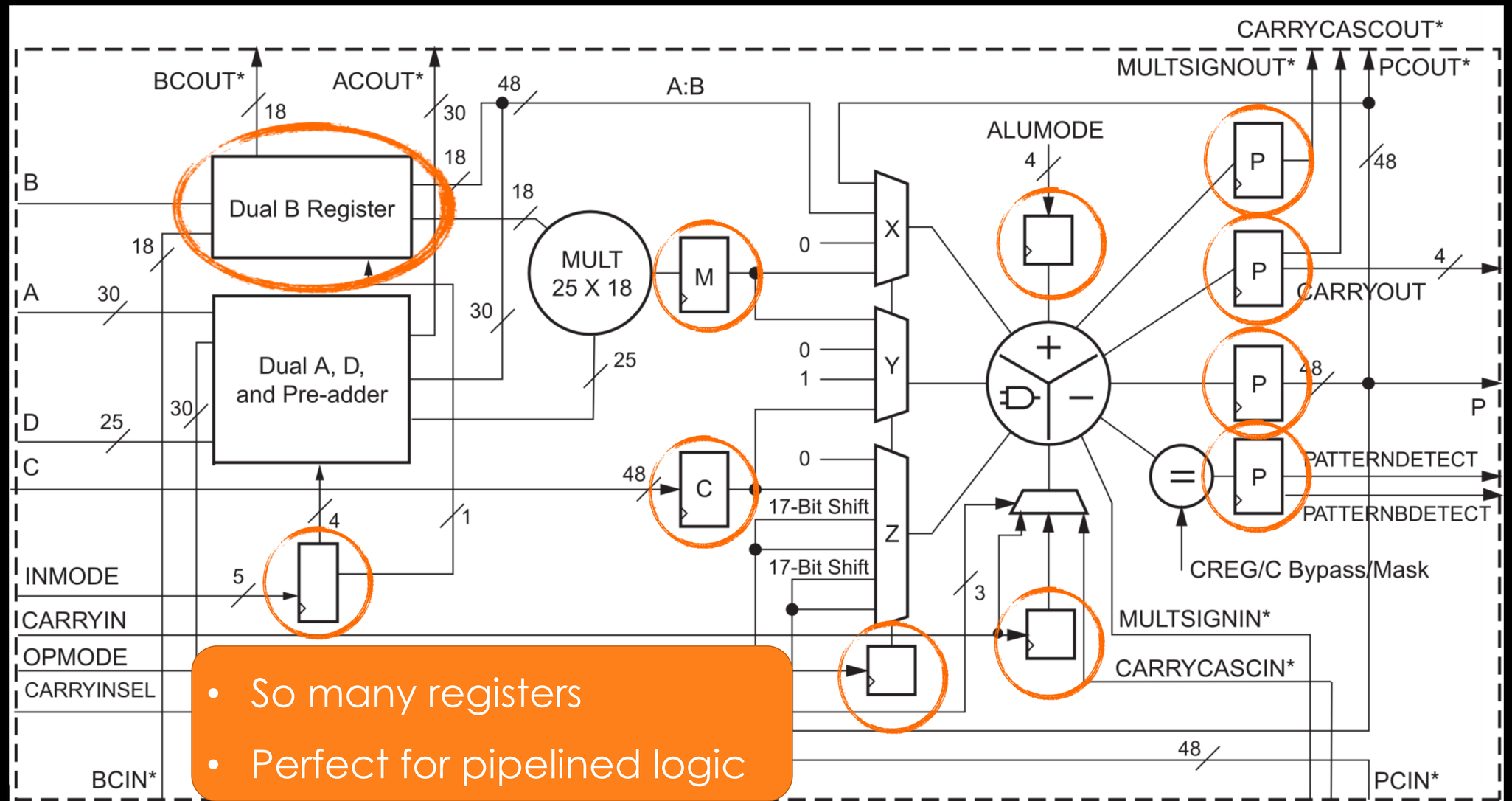


- Registers on the output of every cell
- Perfect for pipelined logic

INTEGRATED DIGITAL SIGNAL PROCESSING



INTEGRATED DIGITAL SIGNAL PROCESSING



- So many registers
- Perfect for pipelined logic

BIGGEST XILINX "ULTRASCALE+" DEVICES

- Upwards of 2million logic cells
  - All clocked at up to 500MHz
  - Up to O(10^{15}) operations/second
    - 1 PetaBOp
- Upwards of 6000 DSPs
- All pipelined
- Fully programmable

Device Name	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	2,364	2,592	3,456	8,172
CLB LUTs (K)	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	75.9	70.9	94.5	75.9
UltraRAM (Mb)	270.0	270.0	360.0	90.0
DSP Slices	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[(1)]	–	–	–	8
150G Interlaken	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	9	9	12	0
Max. Single-Ended HP I/Os	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	96
GTY 32.75Gb/s Transceivers	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–
Extended[(2)]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	–

BIGGEST XILINX "ULTRASCALE+" DEVICES

- Upwards of 2million logic cells
  - All clocked at up to 500MHz
  - Up to O(10^{15}) operations/second
    - 1 PetaBOp
- Upwards of 6000 DSPs
- All pipelined
- Fully programmable

Device Name	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	2,364	2,592	3,456	8,172
CLB LUTs (K)	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	75.9	70.9	94.5	75.9
UltraRAM (Mb)	270.0	270.0	360.0	90.0
DSP Slices	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	8
150G Interlaken	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	9	9	12	0
Max. Single-Ended HP I/Os	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	96
GTY 32.75Gb/s Transceivers	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	–

4.2 Tb/s!!!!

BIGGEST XILINX "ULTRASCALE+" DEVICES

- Upwards of 2million logic cells
  - All clocked at up to 500MHz
  - Up to O(10^{15}) operations/second
    - 1 PetaBOp
- Upwards of 6000 DSPs
- All pipelined
- Fully programmable
- So what is the catch?

Device Name	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	2,364	2,592	3,456	8,172
CLB LUTs (K)	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	75.9	70.9	94.5	75.9
UltraRAM (Mb)	270.0	270.0	360.0	90.0
DSP Slices	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	8
150G Interlaken	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	9	9	12	0
Max. Single-Ended HP I/Os	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	96
GTY 32.75Gb/s Transceivers	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	–

4.2 Tb/s!!!!

FPGAS: WHAT'S THE CATCH?

- Incredibly hard to program efficiently
  - Thinking in a parallel, pipelined-fashion is exceptionally difficult
  - A handful of real experts in CMS
- Efficient use depends on efficiently structured data

FPGAS: WHAT'S THE CATCH?

- Incredibly hard to program efficiently
  - Thinking in a parallel, pipelined-fashion is exceptionally difficult
  - A handful of real experts in CMS
- Efficient use depends on efficiently structured data

Recall:

"The parallel approach to computing does require that some original thinking be done about numerical analysis and data management in order to secure efficient use.

In an environment which has represented the absence of the need to think as the highest virtue, this is a decided disadvantage"

Daniel Slotnick, 1967

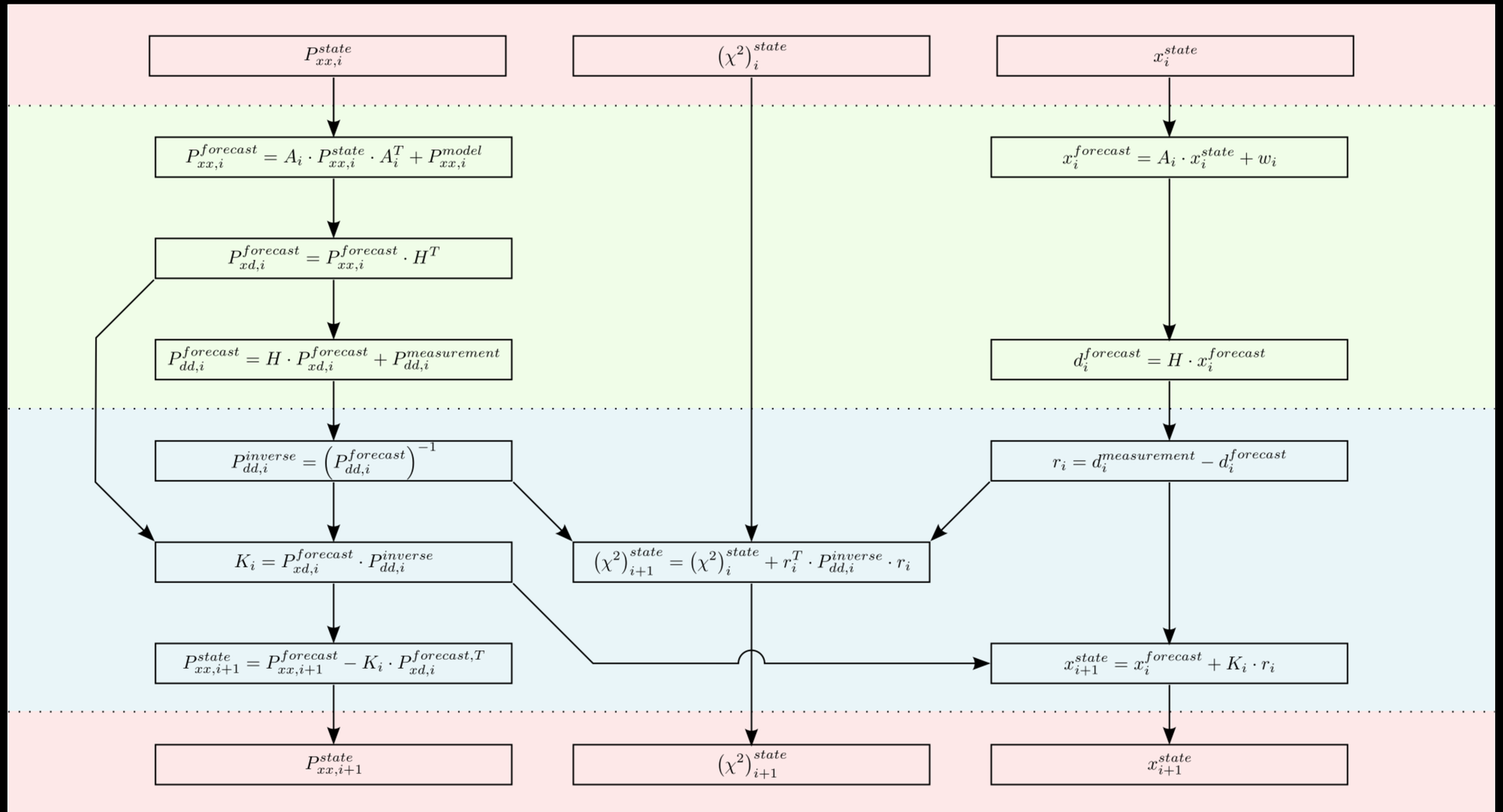
FPGAS: WHAT'S THE CATCH?

- Incredibly hard to program efficiently
  - Thinking in a parallel, pipelined-fashion is exceptionally difficult
  - A handful of real experts in CMS
- Efficient use depends on efficiently structured data
- The chip is just the start – needs to be attached to something
- You are also responsible for the infrastructure

HOW TO PRESERVE YOUR SANITY USING FPGAS

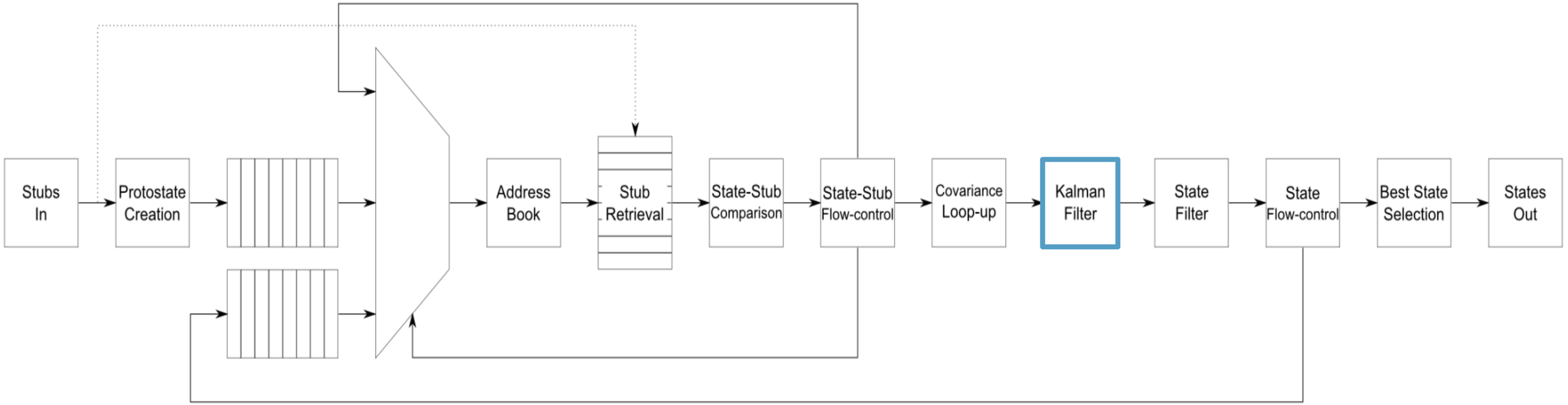
- Keep your data-flow fully flow-forwards
  - **No iterations**
  - or at least
  - **Flatten your loops**

PROTOTYPE CMS TRACKING TRIGGER: KALMAN FILTER



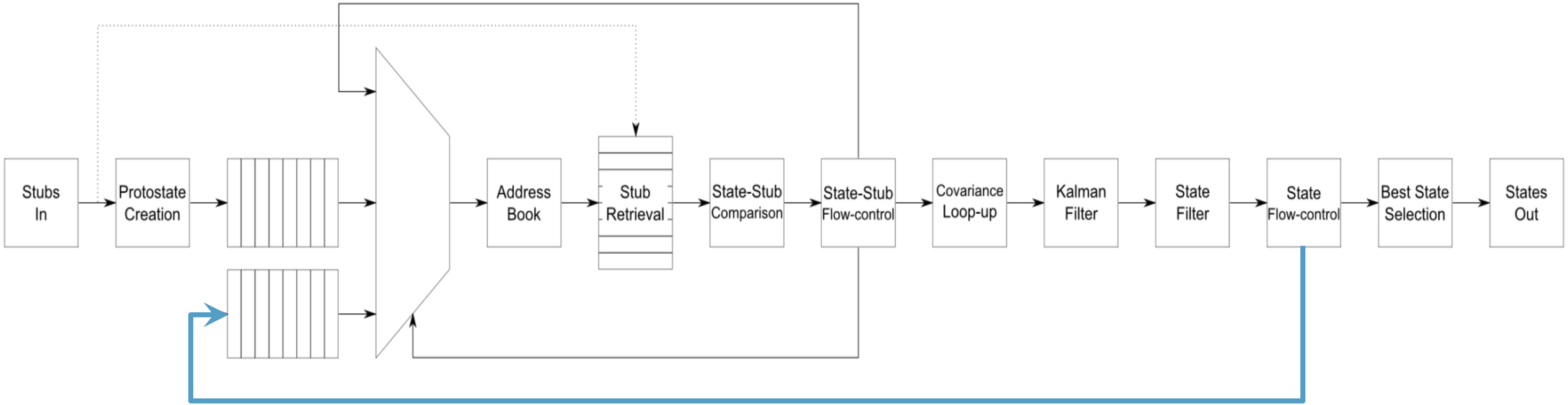
PROTOTYPE CMS TRACKING TRIGGER: KALMAN FILTER

PROTOTYPE CMS TRACKING TRIGGER: KALMAN FILTER



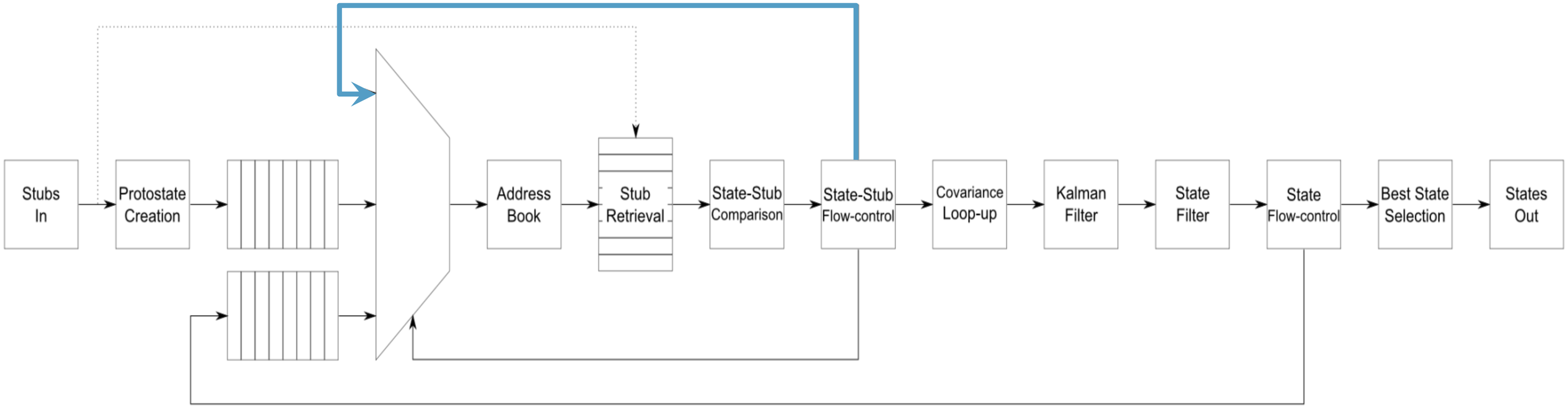
The maths is a relatively simple part of a more complex whole

PROTOTYPE CMS TRACKING TRIGGER: KALMAN FILTER



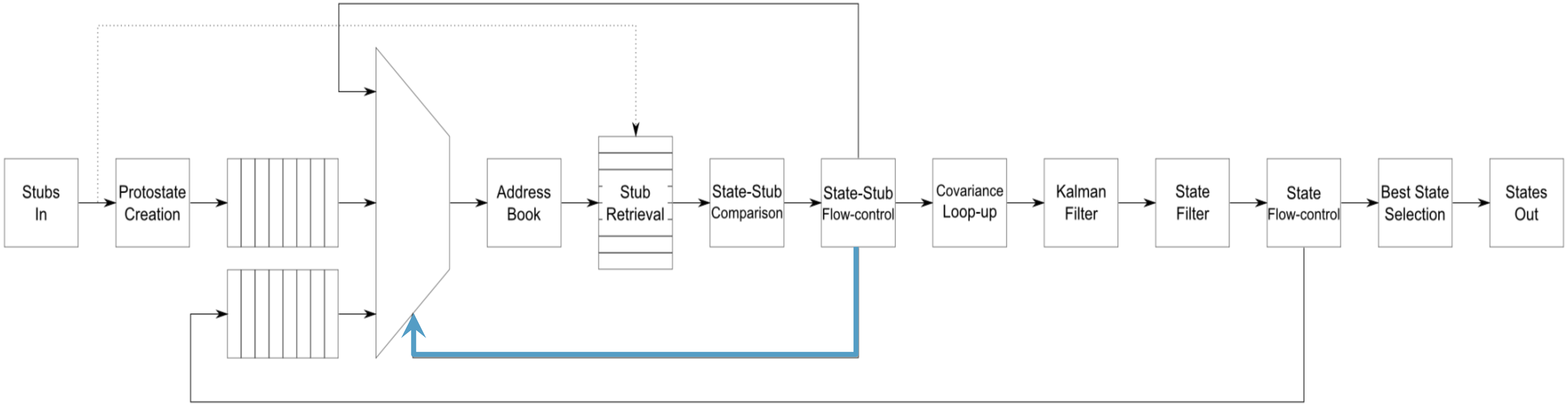
Kalman Filter is iterative

PROTOTYPE CMS TRACKING TRIGGER: KALMAN FILTER



Kalman Filter must handle combinatorics

PROTOTYPE CMS TRACKING TRIGGER: KALMAN FILTER



Kalman Filter data-flow is data-dependent

AN ASIDE ON HIGH-LEVEL SYNTHESIS

- Due to an arbitrary decision by DoE/DARPA/U.S. Govt, FPGA vendors moved C->FPGA compilers from a curiosity to a top-priority
- Reinforced by push for heterogeneous, energy-efficient computing
- Flattens loops, deals with pipelining for you
  - Very simple to get started
  - "Hurrah, we can get our software people writing firmware"
- From practical experience
  - We see very inefficient usage of resources
  - Hard to understand "what the compiler has done"
  - Requires many pre-processor directives to instruct code to do "what you want"
- So, how do you program massively parallelized devices efficiently?

HARDWARE DESCRIPTION LANGUAGES

- Need a language to describe hardware
- Novelly – called a “Hardware Description Language” (HDL)
- Also called FIRMWARE

- Two popular languages are VHDL , VERILOG
- Easy to start learning… Hard to master!

HARDWARE DESCRIPTION LANGUAGES

- Describe Logic as collection of Processes operating in Parallel
- Language Constructs for Synchronous Logic

- Compiler (Synthesis) Tools recognise certain code constructs and generates appropriate logic
- Not all constructs can be implemented in FPGA!

EXAMPLE

```
library ieee;
use ieee.std_logic_1164.all;

entity test is
port(
  x: in std_logic;
  y: in std_logic;
  F: out std_logic;
  G: out std_logic);
end test;
```

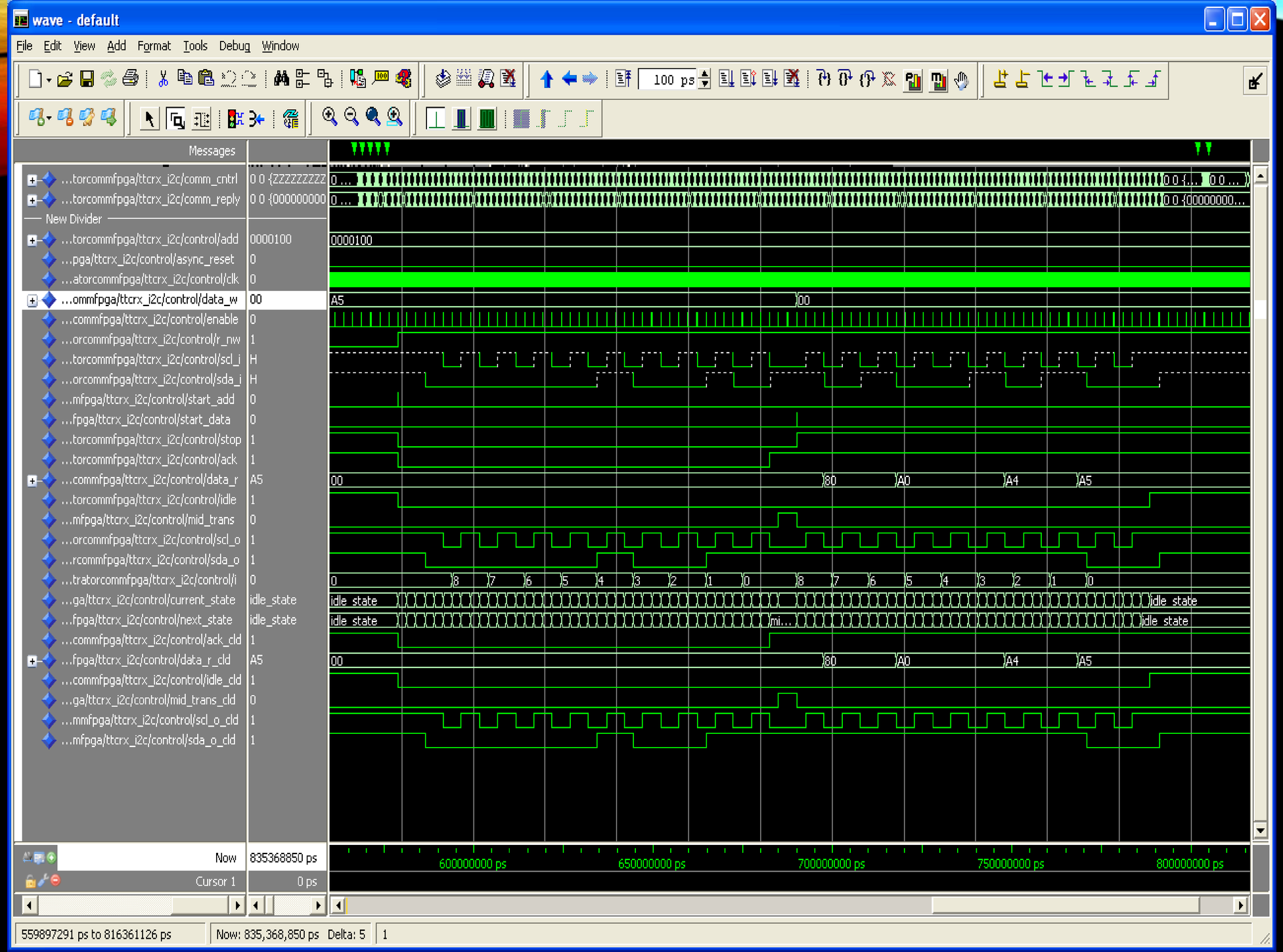
```
architecture behavioural of test is
begin

  process(x, y)
  begin
    -- compare to truth table
    if ((x='1') and (y='1')) then
      F <= '1';
    else
      F <= '0';
    end if;
  end process;

  G <= x or y;
end behavioural;
```

Must write code with understanding of how it will be implemented.

- Can also enter code via schematic entry:
  - Easier to navigate, but not vendor independent
  - Will there ever be a standard graphical programming language?

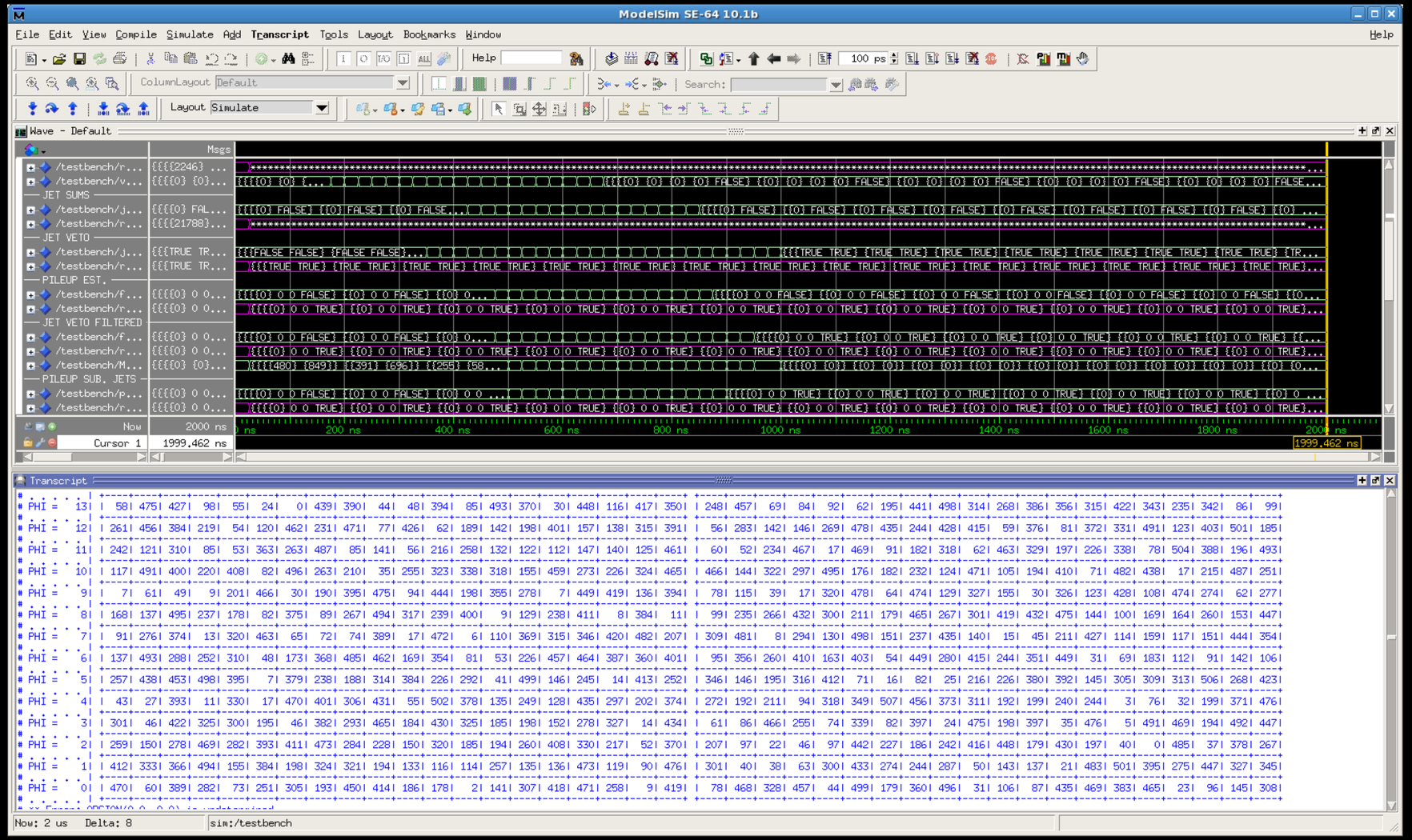
HOW TO YOU KNOW IT WORKS?

- Simulate design extensively!
  - Much quicker than debugging inside the FPGA


wave - default
File Edit View Add Format Tools Debug Window
100 ps
Messages
...torcommfpga/ttcrx_i2c/comm_cntrl
...torcommfpga/ttcrx_i2c/comm_reply
New Divider
...torcommfpga/ttcrx_i2c/control/add
...pga/ttcrx_i2c/control/async_reset
...atorcommfpga/ttcrx_i2c/control/clk
...ommfpga/ttcrx_i2c/control/data_w
...commfpga/ttcrx_i2c/control/enable
...orcommfpga/ttcrx_i2c/control/r_nw
...torcommfpga/ttcrx_i2c/control/scl_i
...orcommfpga/ttcrx_i2c/control/sda_i
...mfpga/ttcrx_i2c/control/start_add
...fpga/ttcrx_i2c/control/start_data
...torcommfpga/ttcrx_i2c/control/stop
...torcommfpga/ttcrx_i2c/control/ack
...commfpga/ttcrx_i2c/control/data_r
...torcommfpga/ttcrx_i2c/control/idle
...mfpga/ttcrx_i2c/control/mid_trans
...orcommfpga/ttcrx_i2c/control/scl_o
...rcommfpga/ttcrx_i2c/control/sda_o
...tratorcommfpga/ttcrx_i2c/control/i
...ga/ttcrx_i2c/control/current_state
...fpga/ttcrx_i2c/control/next_state
...commfpga/ttcrx_i2c/control/ack_cld
...fpga/ttcrx_i2c/control/data_r_cld
...commfpga/ttcrx_i2c/control/idle_cld
...ga/ttcrx_i2c/control/mid_trans_cld
...mmfpga/ttcrx_i2c/control/scl_o_cld
...mfpga/ttcrx_i2c/control/sda_o_cld
0000100
A5
00
idle_state
Now
835368850 ps
Cursor 1
0 ps
600000000 ps
650000000 ps
700000000 ps
750000000 ps
800000000 ps
559897291 ps to 816361126 ps
Now: 835,368,850 ps Delta: 5

TESTBENCH SUITE

"Event display"



TESTBENCH SUITE

Clock-by-clock summary

Transcript

```
# Jets 9x1 Sum[2] : latency 4 clks : Matches expected latency
#
# <<<<<<<<<< Clock 8 >>>>>>>>>>
# Towers[5] : latency 2 clks : Matches expected latency
# Jets 9x1 Sum[3] : latency 4 clks : Matches expected latency
#
# <<<<<<<<<< Clock 9 >>>>>>>>>>
# Towers[6] : latency 2 clks : Matches expected latency
# Jets 9x1 Sum[4] : latency 4 clks : Matches expected latency
# Jets 1x9 Sum[0] : latency 8 clks : Matches expected latency
#
# <<<<<<<<<< Clock 10 >>>>>>>>>>
# Towers[7] : latency 2 clks : Matches expected latency
# Jets 9x1 Sum[5] : latency 4 clks : Matches expected latency
# Jets 1x9 Sum[1] : latency 8 clks : Matches expected latency
# ET AND MET Rings[0] : latency 9 clks : Matches expected latency
#
# <<<<<<<<<< Clock 11 >>>>>>>>>>
# Towers[8] : latency 2 clks : Matches expected latency
# Jets 9x1 Sum[6] : latency 4 clks : Matches expected latency
# Jets 1x9 Sum[2] : latency 8 clks : Matches expected latency
# ET AND MET Rings[1] : latency 9 clks : Matches expected latency
# Accumulated ET AND MET Rings[0] : latency 10 clks : Matches expected latency
#
# <<<<<<<<<< Clock 12 >>>>>>>>>>
# Towers[9] : latency 2 clks : Matches expected latency
# Jets 9x1 Sum[7] : latency 4 clks : Matches expected latency
# Jets 1x9 Sum[3] : latency 8 clks : Matches expected latency
# ET AND MET Rings[2] : latency 9 clks : Matches expected latency
```

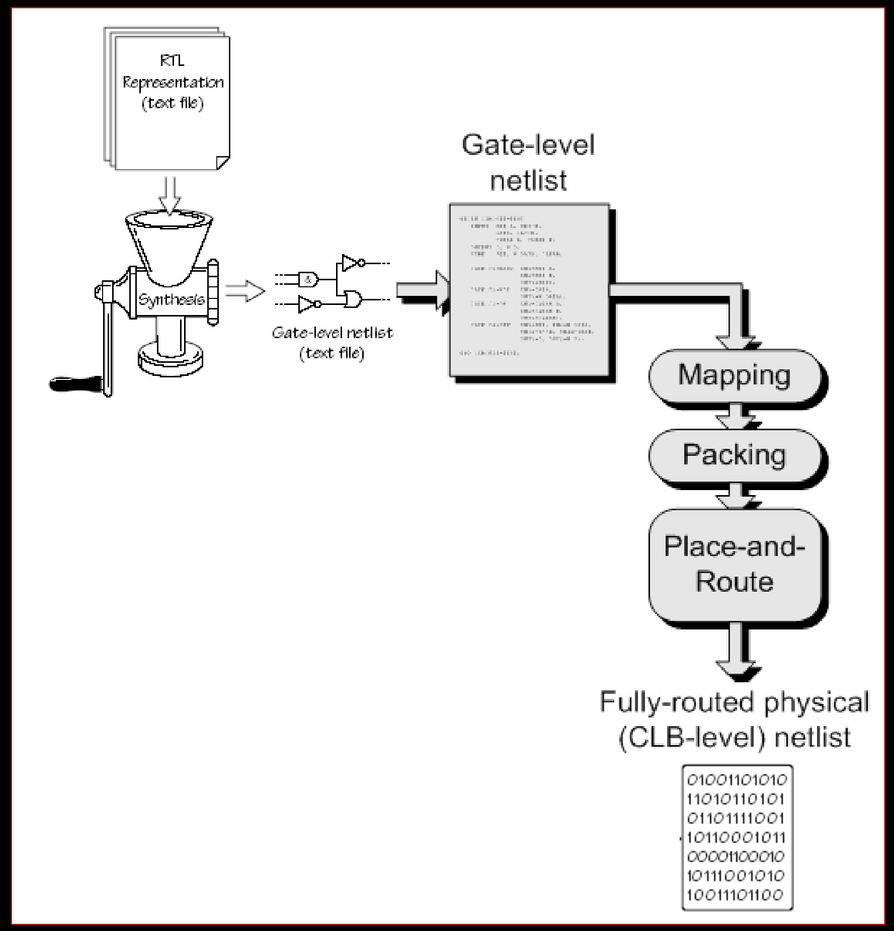
End-of-event summary

Transcript

```
# |                                          Jets 9x9 Filtered[ 9] |                                                                  Test Successful |
# |                                          Jets 9x9 Filtered[10] |                                                                  Test Successful |
# |                                          Jets 9x9 Filtered[11] |                                                                  Test Successful |
# |                                          Jets 9x9 Filtered[12] |          Test Successful [Multiple matches including correct latency] |
# |                                          Jets 9x9 Filtered[13] |                                                                  Test Successful |
# |                                          Jets 9x9 Filtered[14] |                                                                  Test Successful |
# +-------------------------------------------------------------------------------------------------------------------------------------------------------+
# | Jets 9x9 Filtered                                              | SUCCESS                                                                              |
# +-------------------------------------------------------------------------------------------------------------------------------------------------------+
# |                                 Jets 9x9 PU subtracted Sum[ 0] |          Test Successful [Multiple matches including correct latency] |
# |                                 Jets 9x9 PU subtracted Sum[ 1] |                                                                  Test Successful |
# |                                 Jets 9x9 PU subtracted Sum[ 2] |          Test Successful [Multiple matches including correct latency] |
# |                                 Jets 9x9 PU subtracted Sum[ 3] |                                                                  Test Successful |
# |                                 Jets 9x9 PU subtracted Sum[ 4] |                                                                  Test Successful |
# |                                 Jets 9x9 PU subtracted Sum[ 5] |                                                                  Test Successful |
# |                                 Jets 9x9 PU subtracted Sum[ 6] |                                                                  Test Successful |
# |                                 Jets 9x9 PU subtracted Sum[ 7] |                                                                  Test Successful |
# |                                 Jets 9x9 PU subtracted Sum[ 8] |                                                                  Test Successful |
# |                                 Jets 9x9 PU subtracted Sum[ 9] |                                                                  Test Successful |
# |                                 Jets 9x9 PU subtracted Sum[10] |                                                                  Test Successful |
# |                                 Jets 9x9 PU subtracted Sum[11] |                                                                  Test Successful |
# |                                 Jets 9x9 PU subtracted Sum[12] |          Test Successful [Multiple matches including correct latency] |
# |                                 Jets 9x9 PU subtracted Sum[13] |                                                                  Test Successful |
# |                                 Jets 9x9 PU subtracted Sum[14] |                                                                  Test Successful |
# +-------------------------------------------------------------------------------------------------------------------------------------------------------+
# | Jets 9x9 PU subtracted Sum                                     | SUCCESS                                                                              |
# +-------------------------------------------------------------------------------------------------------------------------------------------------------+
# |                                       Sorted Jets 9x9[ 0] |          Test Successful [Multiple matches including correct latency] |
# |                                       Sorted Jets 9x9[ 1] |                                                                  Test Successful |
```

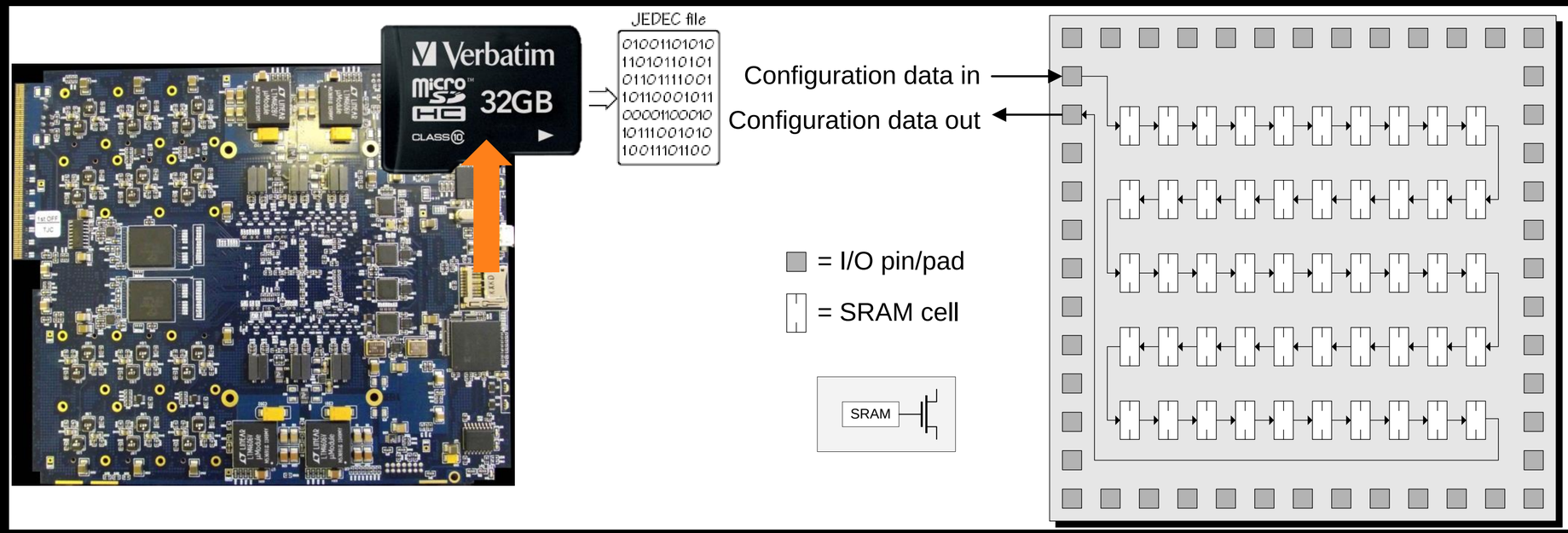
DESIGNING LOGIC WITH FPGAS

- High level Description of Logic Design (HDL)
- Synthesise into a Netlist
  - Boolean Logic Representation
- Target FPGA Device
  - Translate
  - Mapping
  - Routing
- Bit File for FPGA



CONFIGURING AN FPGA

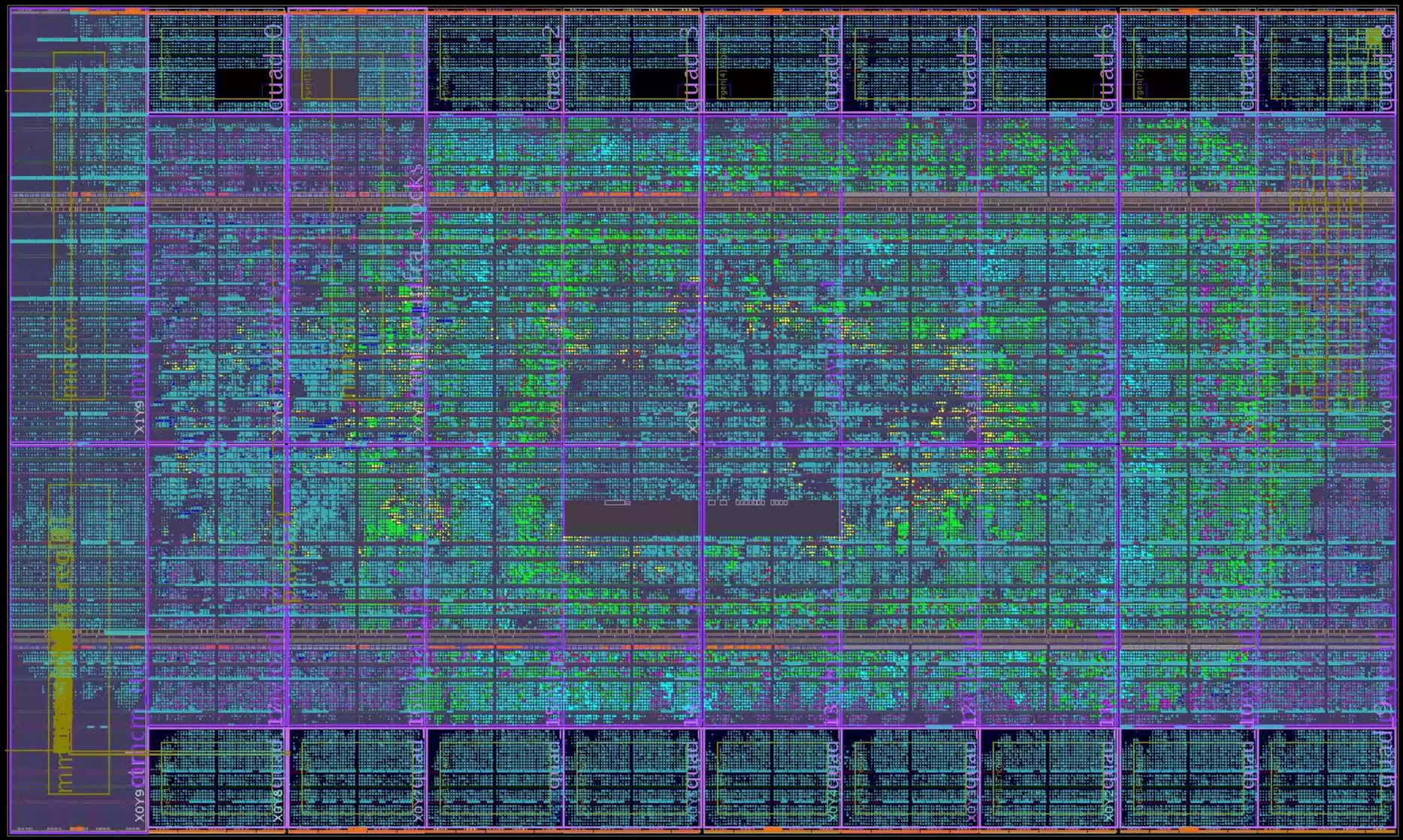
- Millions of SRAM cells holding LUTs and Interconnect Routing
- Volatile Memory: Lose configuration when board power is turned off.
- Keep bit patterns describing the SRAM cells in non-Volatile Memory e.g. PROM or memory card
- Configuration takes ~ secs



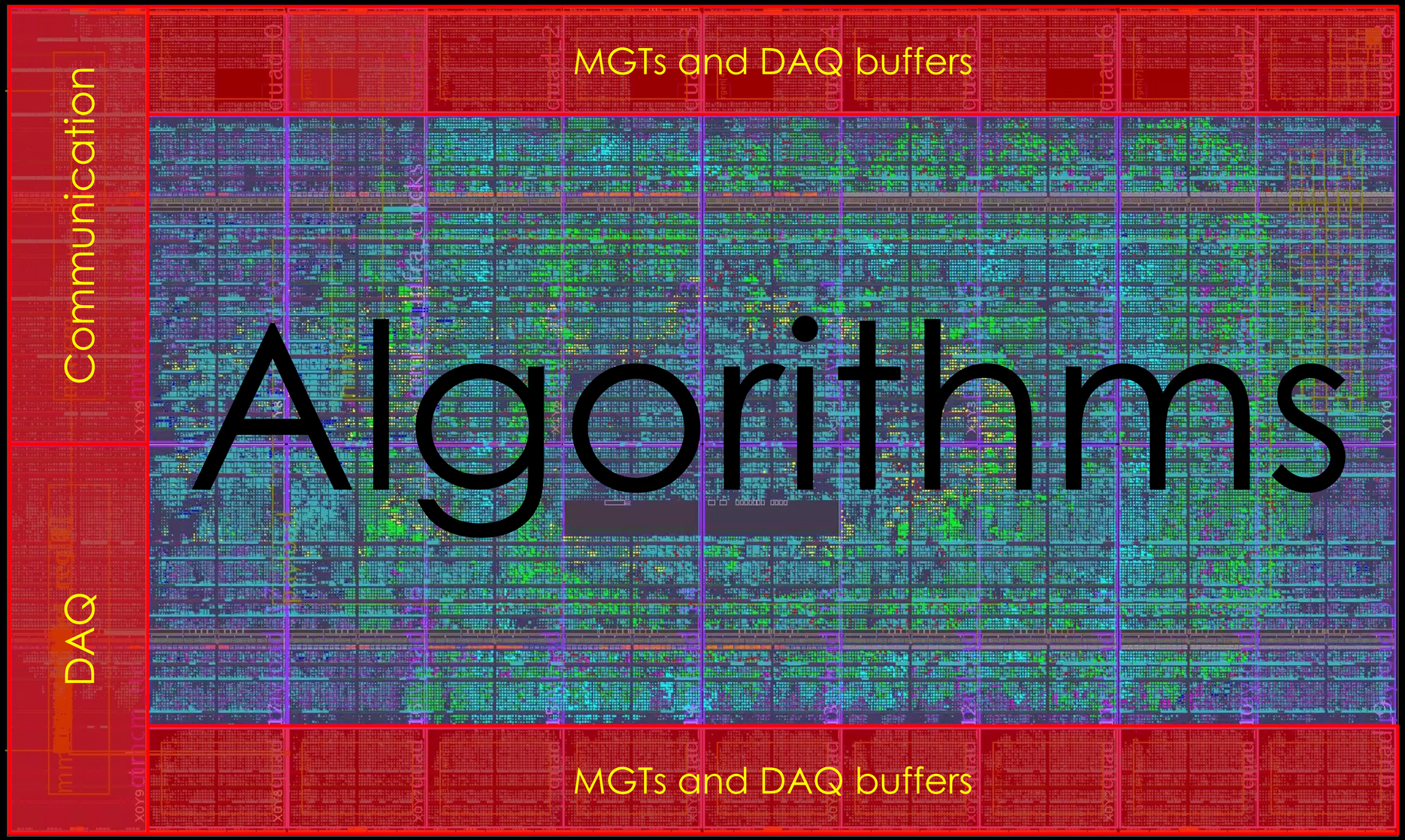
IT DOESN'T WORK: HOW TO DEBUG

- Simulate, simulate & simulate again!
  - Much quicker than debugging inside the FPGA
- Route out signal to periphery
  - Few debug pins always handy
  - Can connect UART for uC debug (StdIn/StdOut)
- Use chipscope
  - Rebuild design with embedded logic analyser
    - Can be a bit like quantum mechanics
    - If you look (i.e. make a measurement) your code can behave differently
    - Chipscope presence can affect the original design

FLOORPLAN OF FIRMWARE IN MP7



FLOORPLAN OF FIRMWARE IN MP7



WHEN & WHY SHOULD I (NOT) USE AN FPGA?

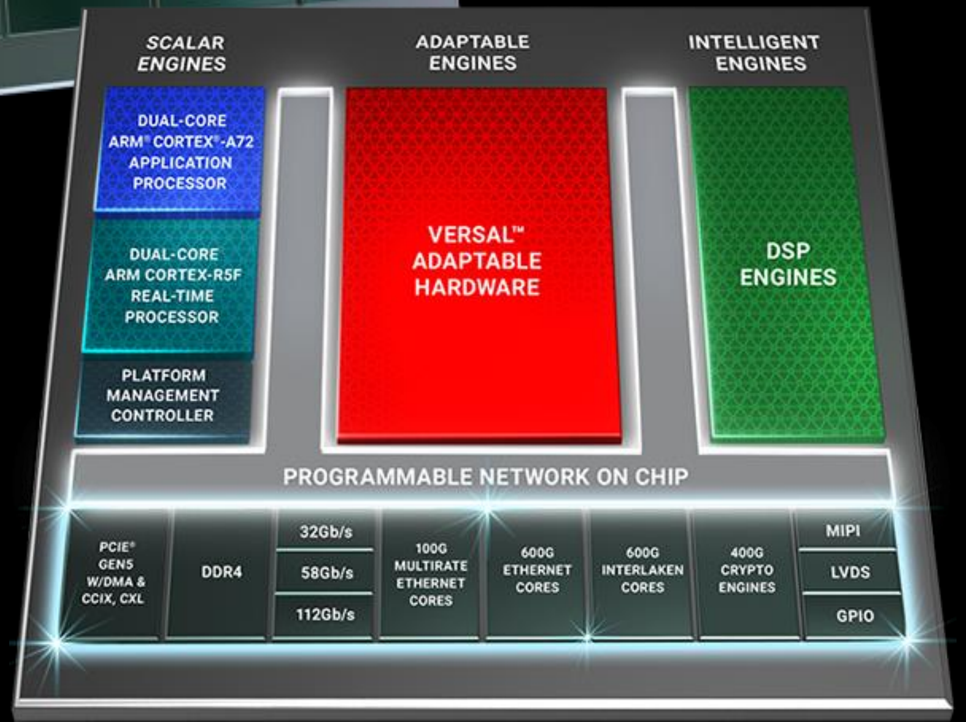
- FPGAs are expensive (high-end £10k-100k cf. £100)
- FPGAs are power-hungry
- Programming FPGAs is like designing logic circuits not like programming sequential microcontrollers
- Large firmware build-times are tens of hours or days
- Floating-point ops and iterative algorithms awkward in FPGAs (That said, you "control" the silicon, so, of course, it can be done)
- **FPGAs best for high through-put, low- and/or fixed-latency operations**

CONCLUSION

- FPGAs are intrinsically parallel
- Modern FPGAs are exceptionally powerful
- FPGAs are a monumental PAIN IN THE BACKSIDE to program
  - Partly due to the clunky, verbose HDLs
  - Mainly due to the difficulty of conceptualizing massively parallel logic and pipelined logic

- **Get them right and you can do magic**
- **Get them wrong and you unleashed a world of pain on yourself**

THE FUTURE OF THE FPGA?

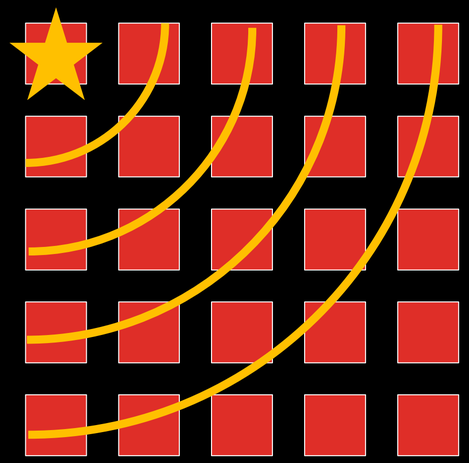


- Heterogenous computing on chip
- But is it suitable for our typical applications in particle physics?
  - Is it suitable for future applications?
  - Hardware Triggers? Probably not – designed as co-processor
  - Accelerated HLTs? Maybe – but GPUs more likely…

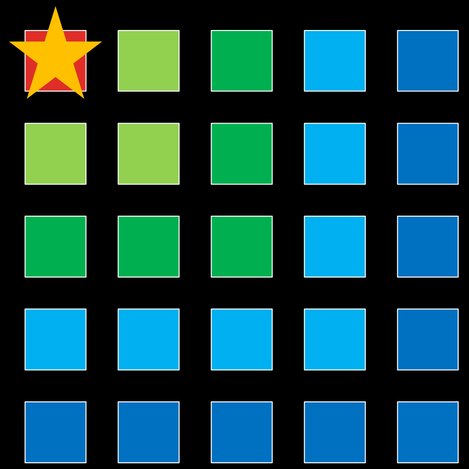
THANK YOU

Any questions?

UP AGAINST THE SPEED OF LIGHT…



- Wait for the signal to propagate
- “Sea-of-logic” approach
- Limits clock speed



- Do less each clock-cycle
- Compensated for by much higher clock speeds