

AN INTRODUCTION TO
FIELD PROGRAMMABLE
GATE ARRAYS

UK Advanced Instrumentation Course 2026

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 weeks or months of hands-on work would be more realistic

RECALL FROM TRIGGER & DAQ LECTURES

	6pm	7pm	8pm	9pm	10pm	11pm	12pm	01am	02am	03am
										
										
										
										

That would just be stupid

CPU

	6pm	7pm	8pm	9pm	10pm	11pm	12pm	01am	02am	03am
										
										
										
										

FPGA

RECALL FROM TRIGGER & DAQ LECTURES

	6pm	7pm	8pm	9pm	10pm	11pm	12pm	01am	02am	03am
										
										
										
										

That would just be stupid

CPU

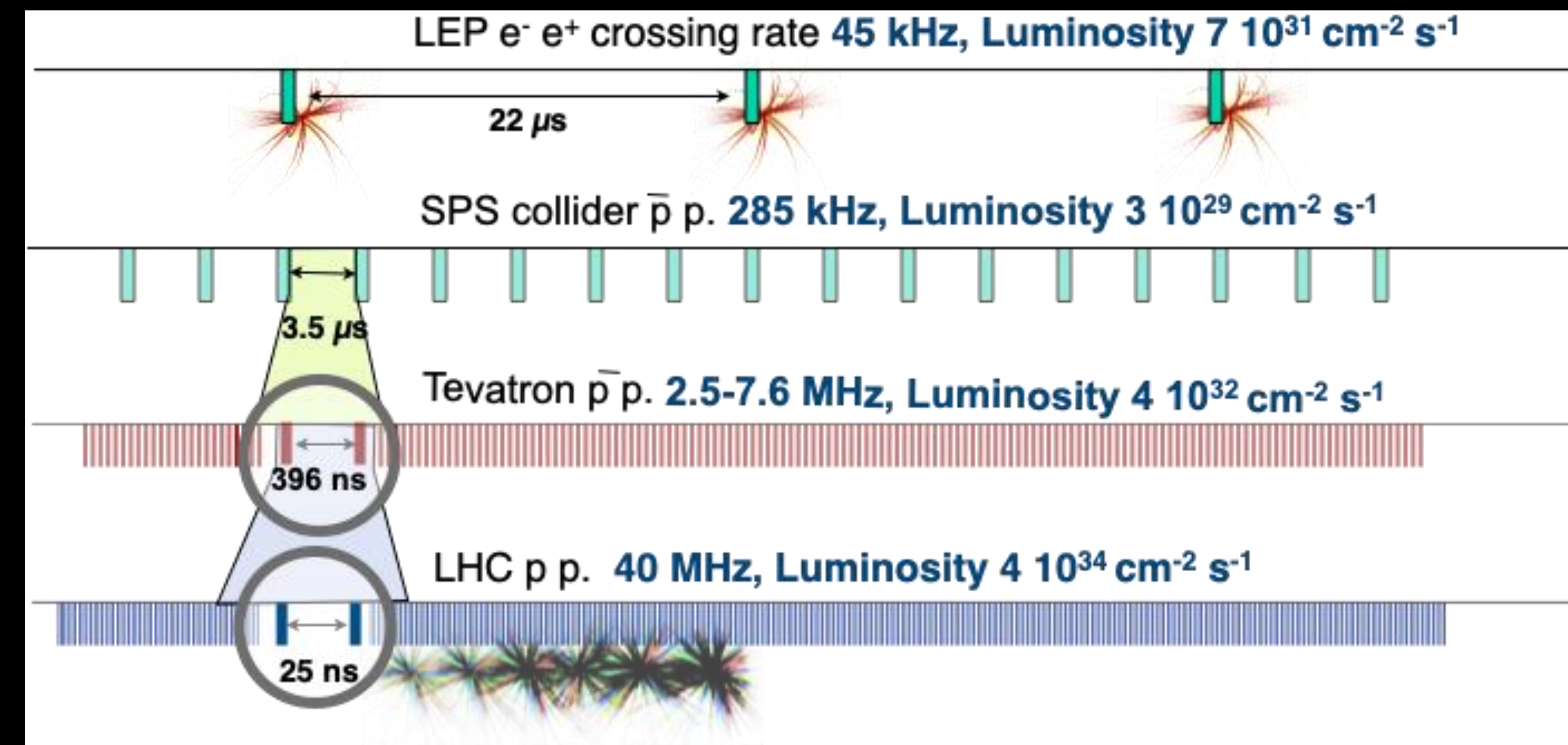
	6pm	7pm	8pm	9pm	10pm	11pm	12pm	01am	02am	03am
										
										
										
										

FPGA

So... I should probably justify those statements...

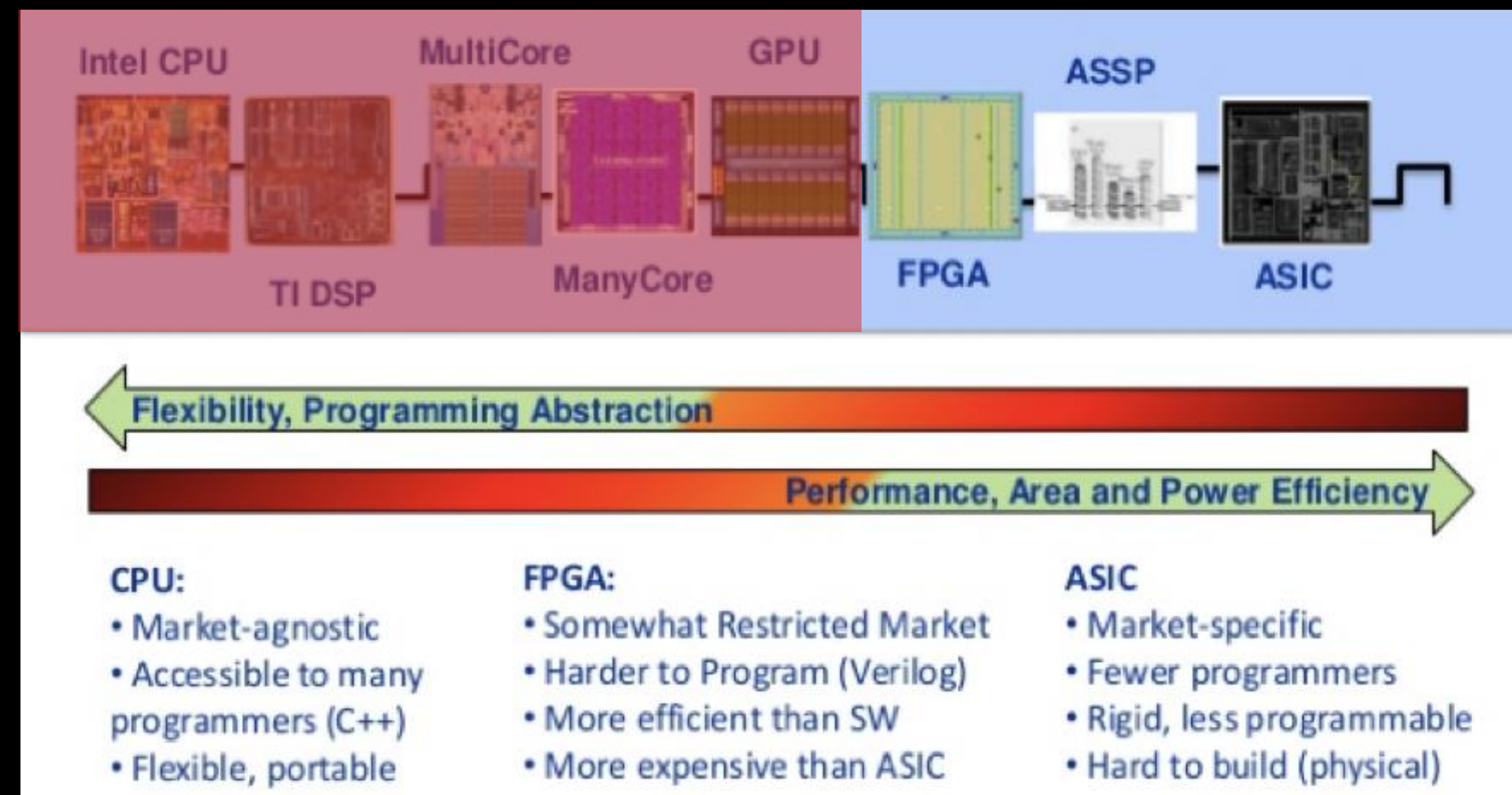
A NOTE ON TIMESCALES

- 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)

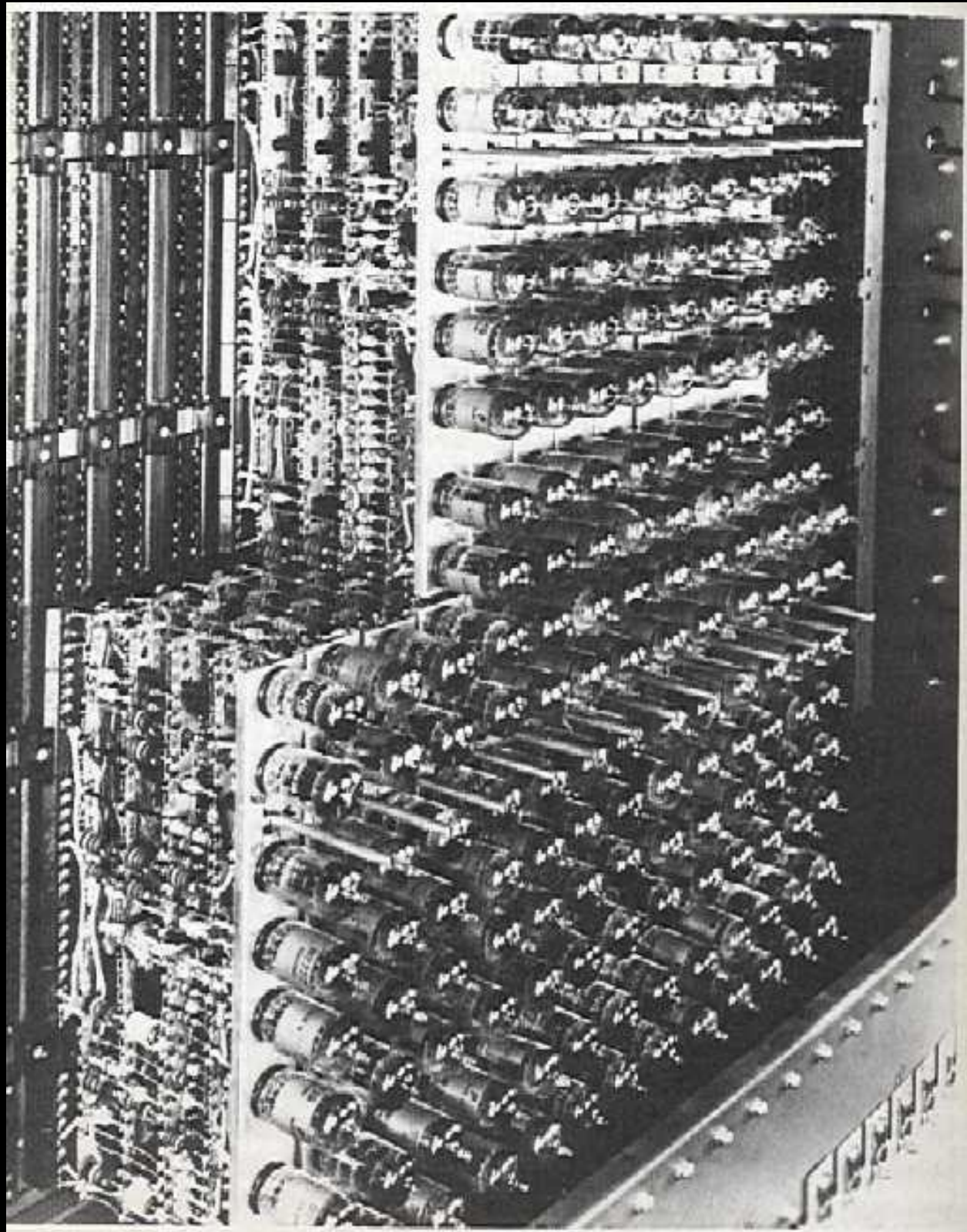


AN ASIDE: THE HISTORY OF ELECTRONICS

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

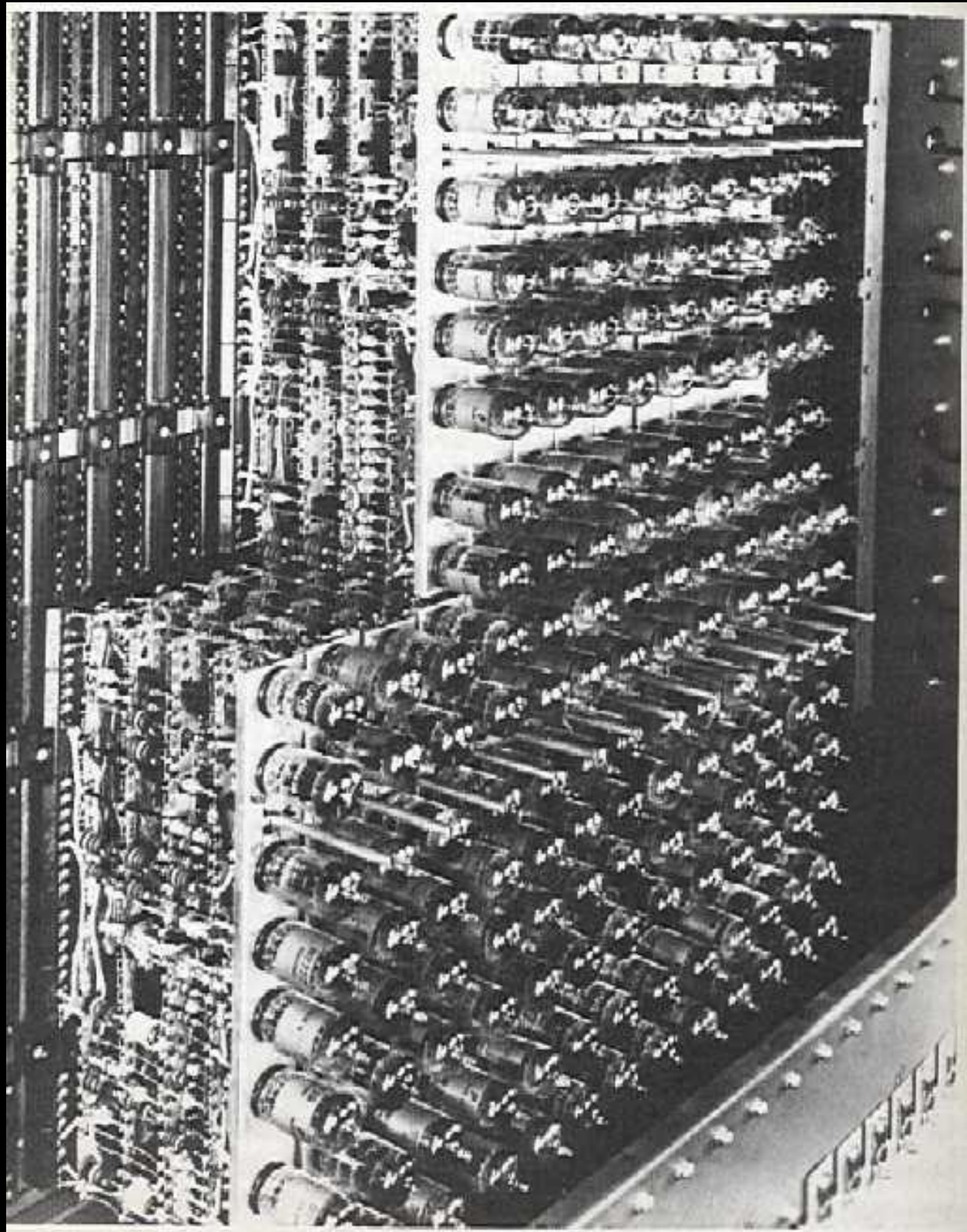


THE HISTORY OF ELECTRONICS



Valve
transistors

THE HISTORY OF ELECTRONICS

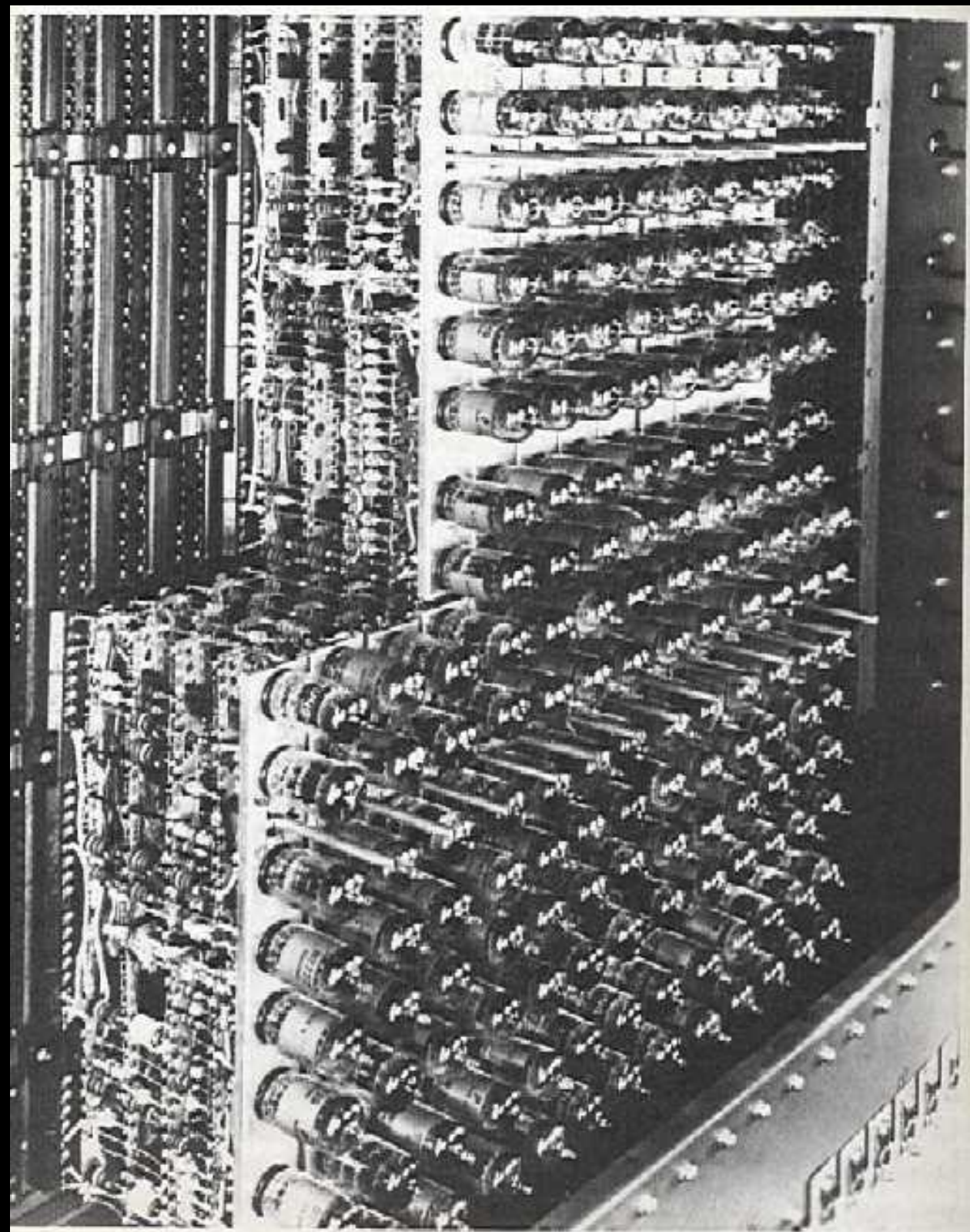


Valve
transistors



First
solid-state
transistors

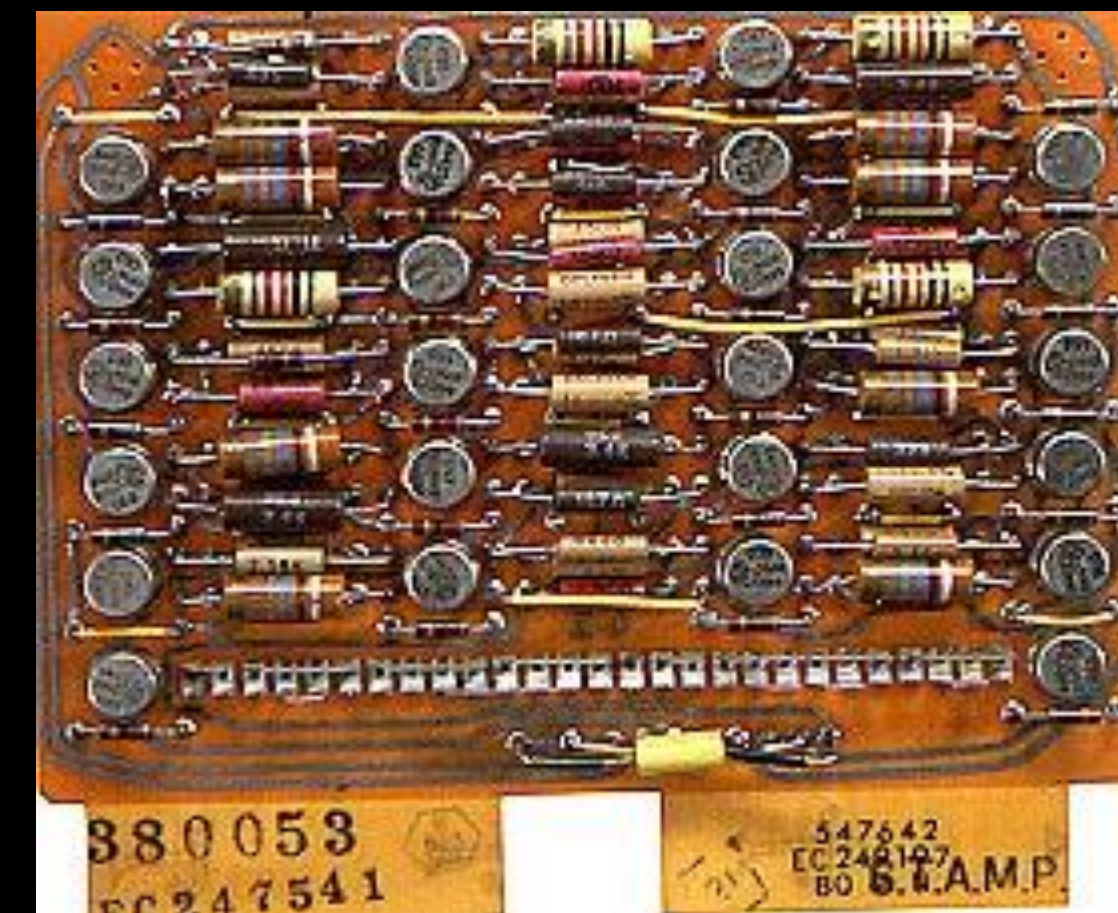
THE HISTORY OF ELECTRONICS



Valve transistors

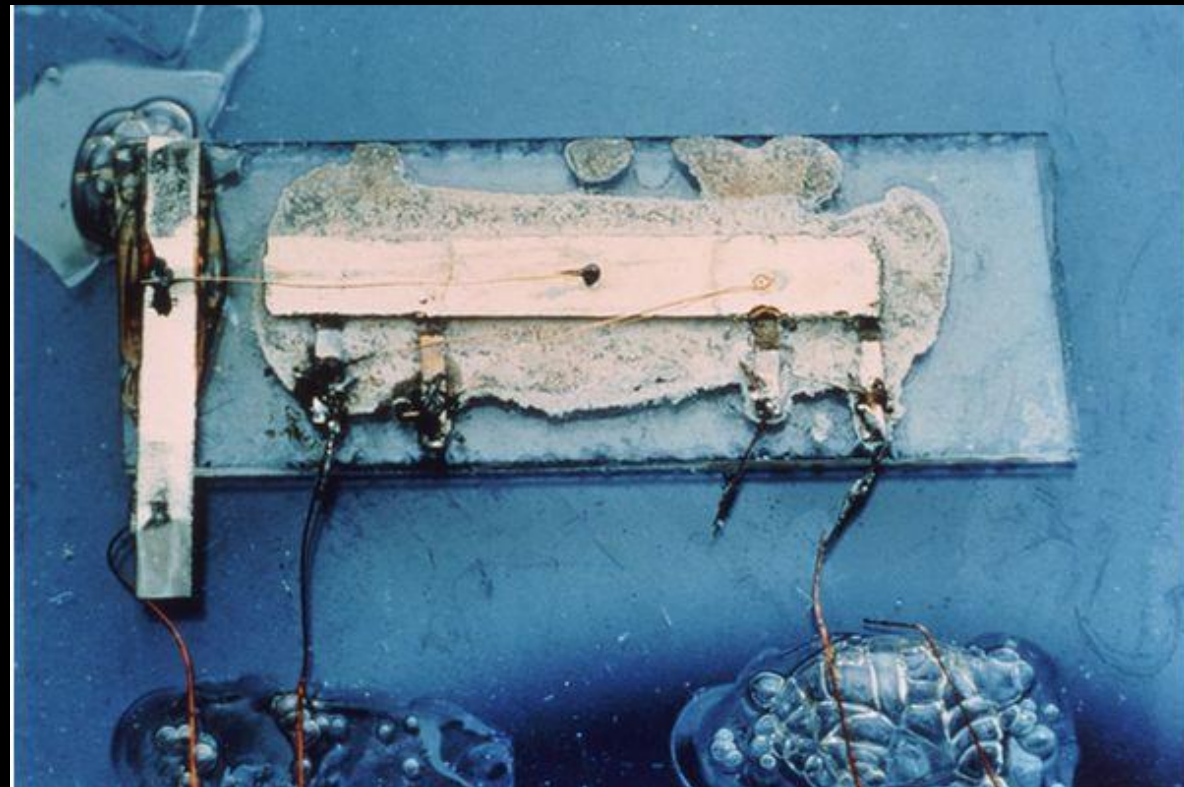


First solid-state transistors



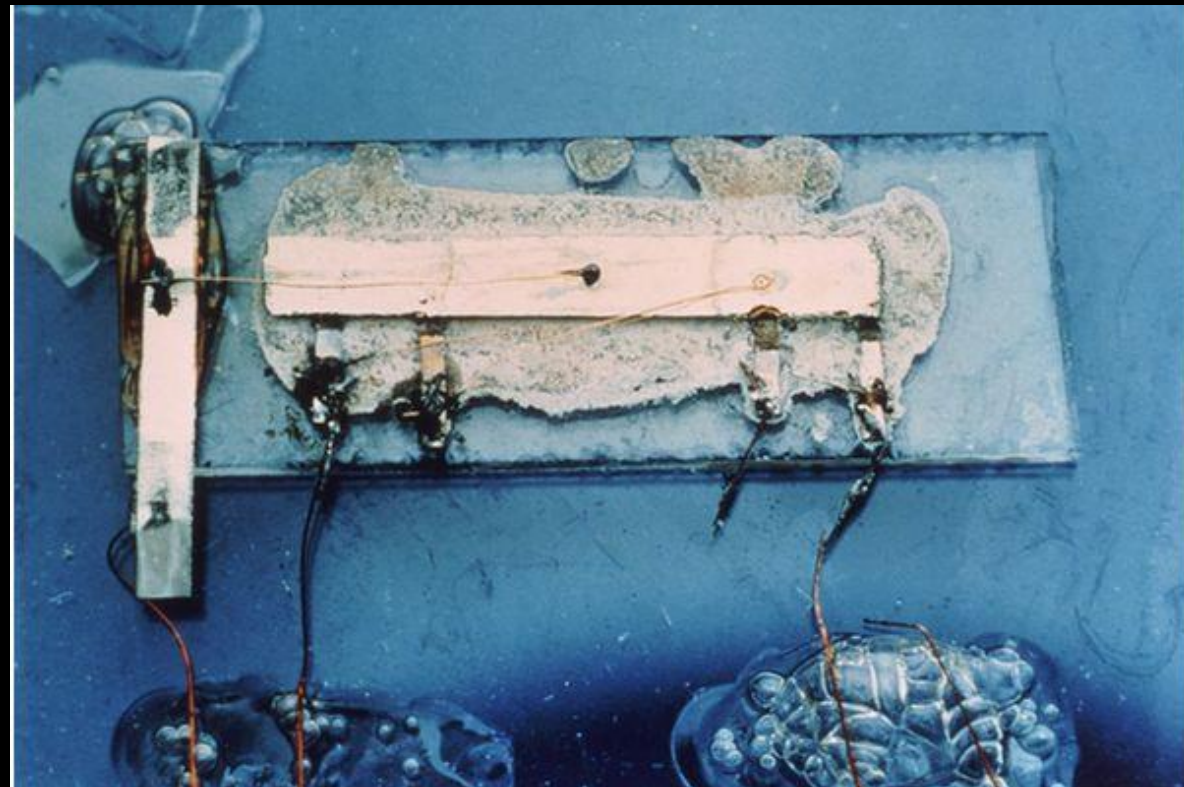
Solid-state transistors

THE HISTORY OF ELECTRONICS

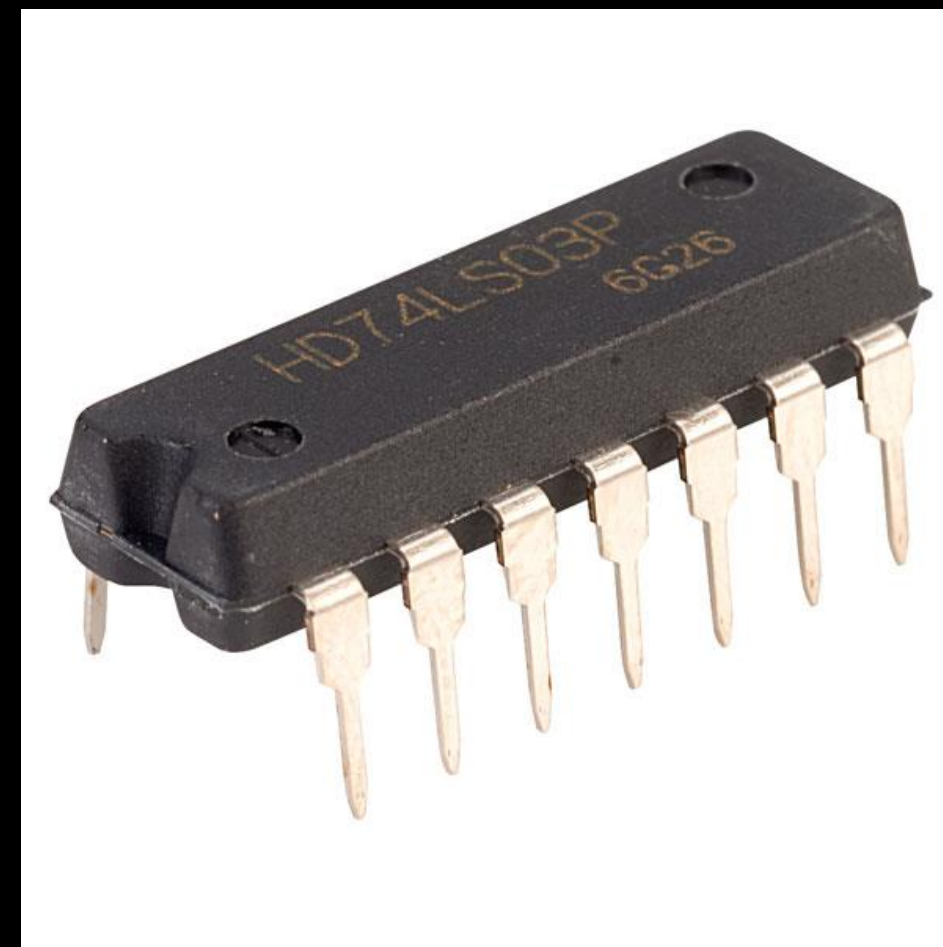


First
multi-transistor
silicon

THE HISTORY OF ELECTRONICS

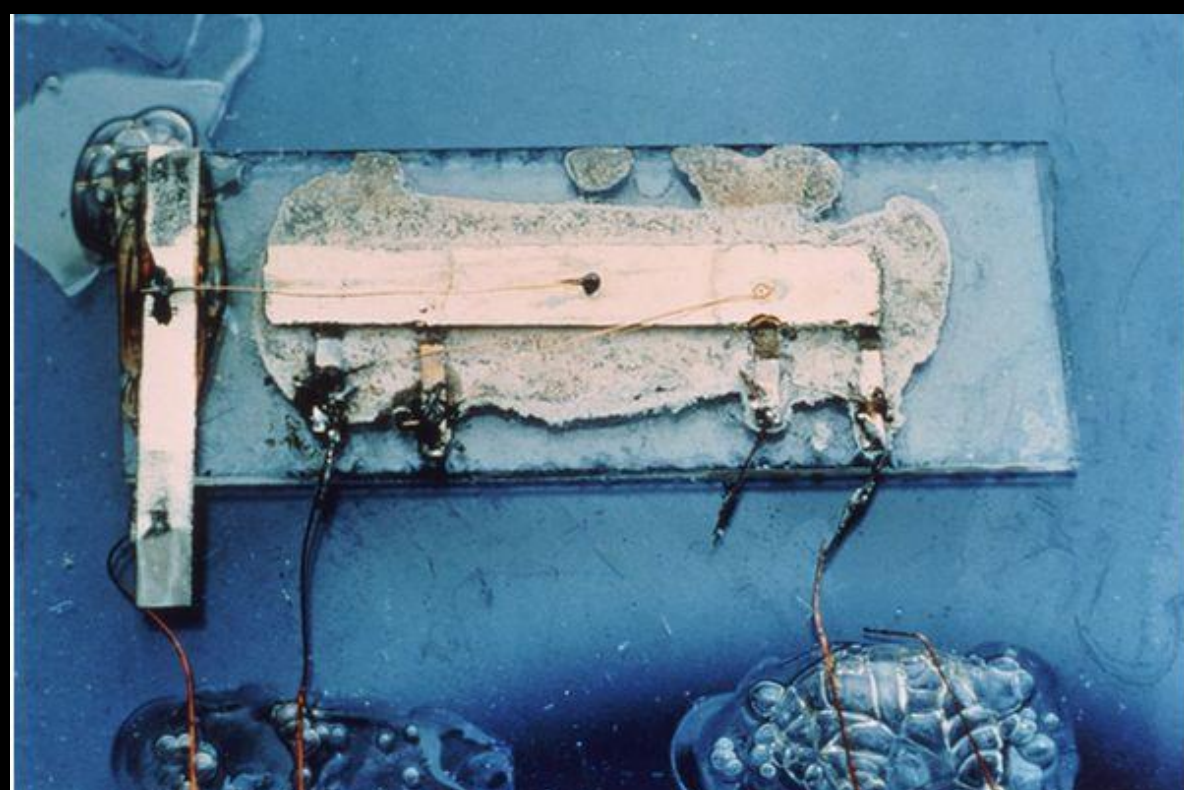


First
multi-transistor
silicon

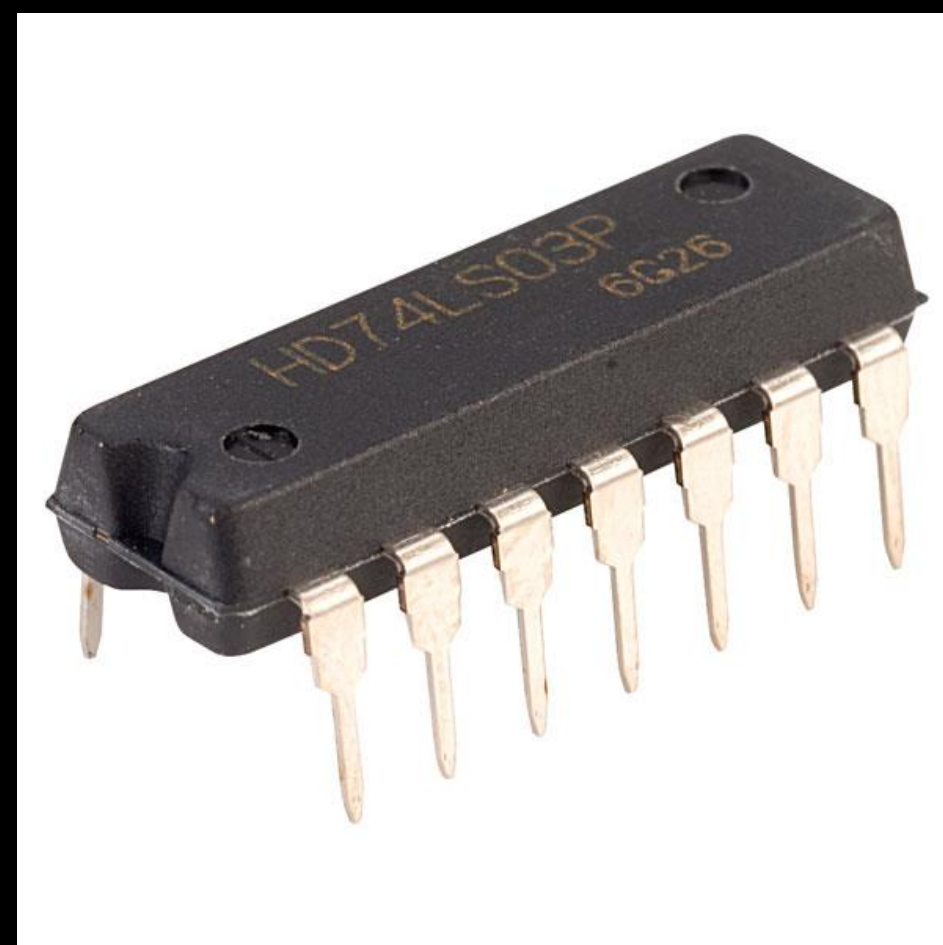


Packaged
Logic

THE HISTORY OF ELECTRONICS



First
multi-transistor
silicon

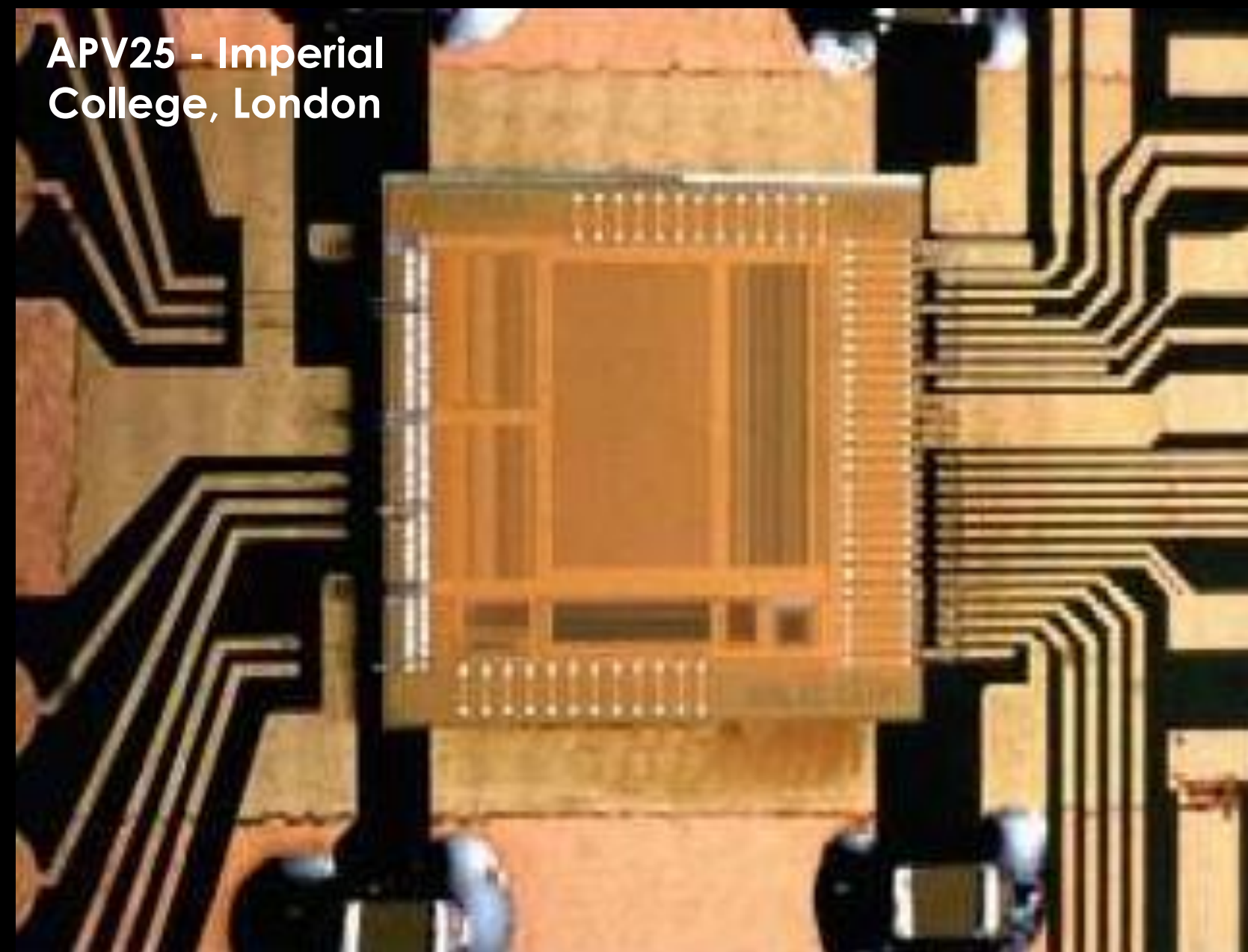


Packaged
Logic



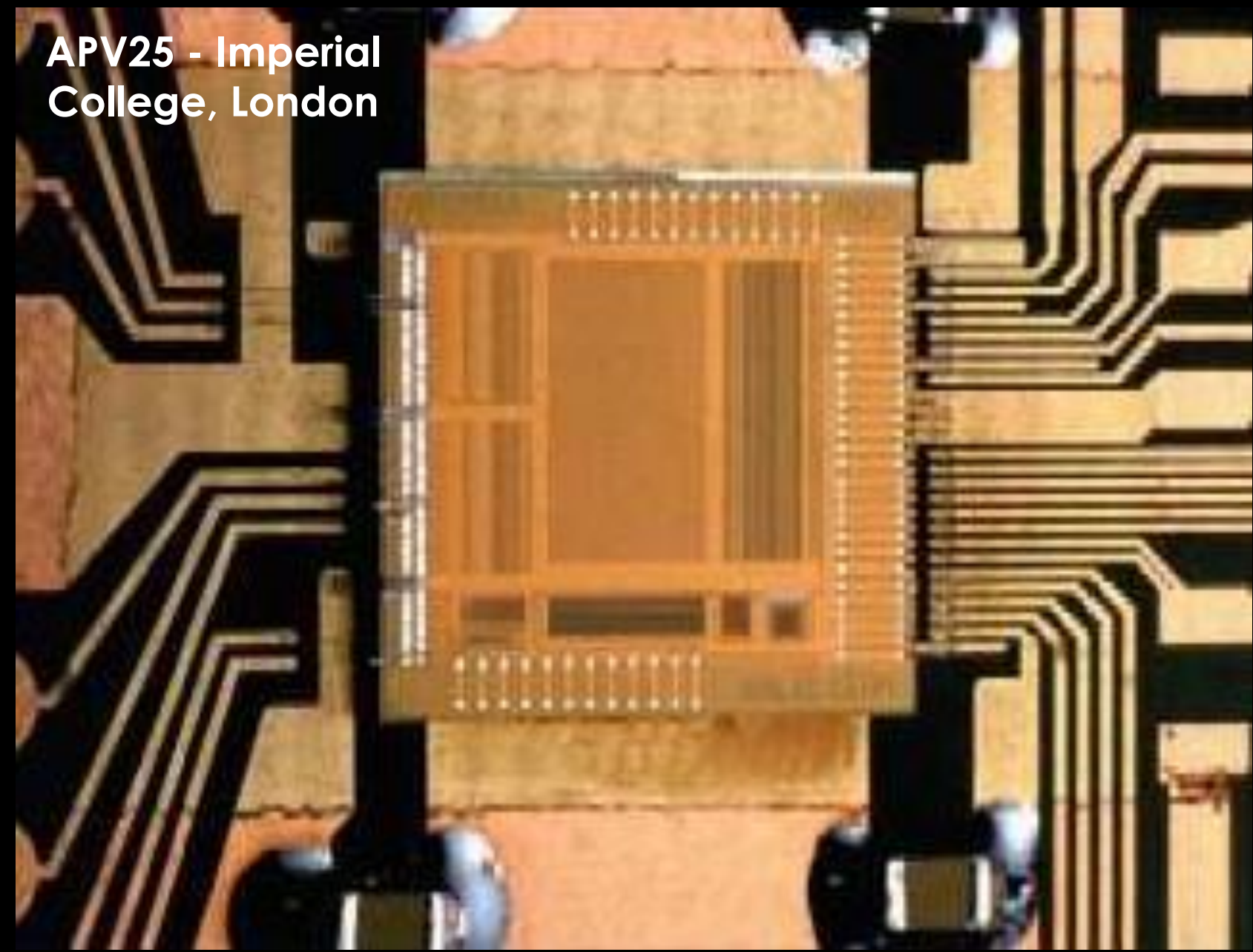
"Mini" processor
board

ASICs

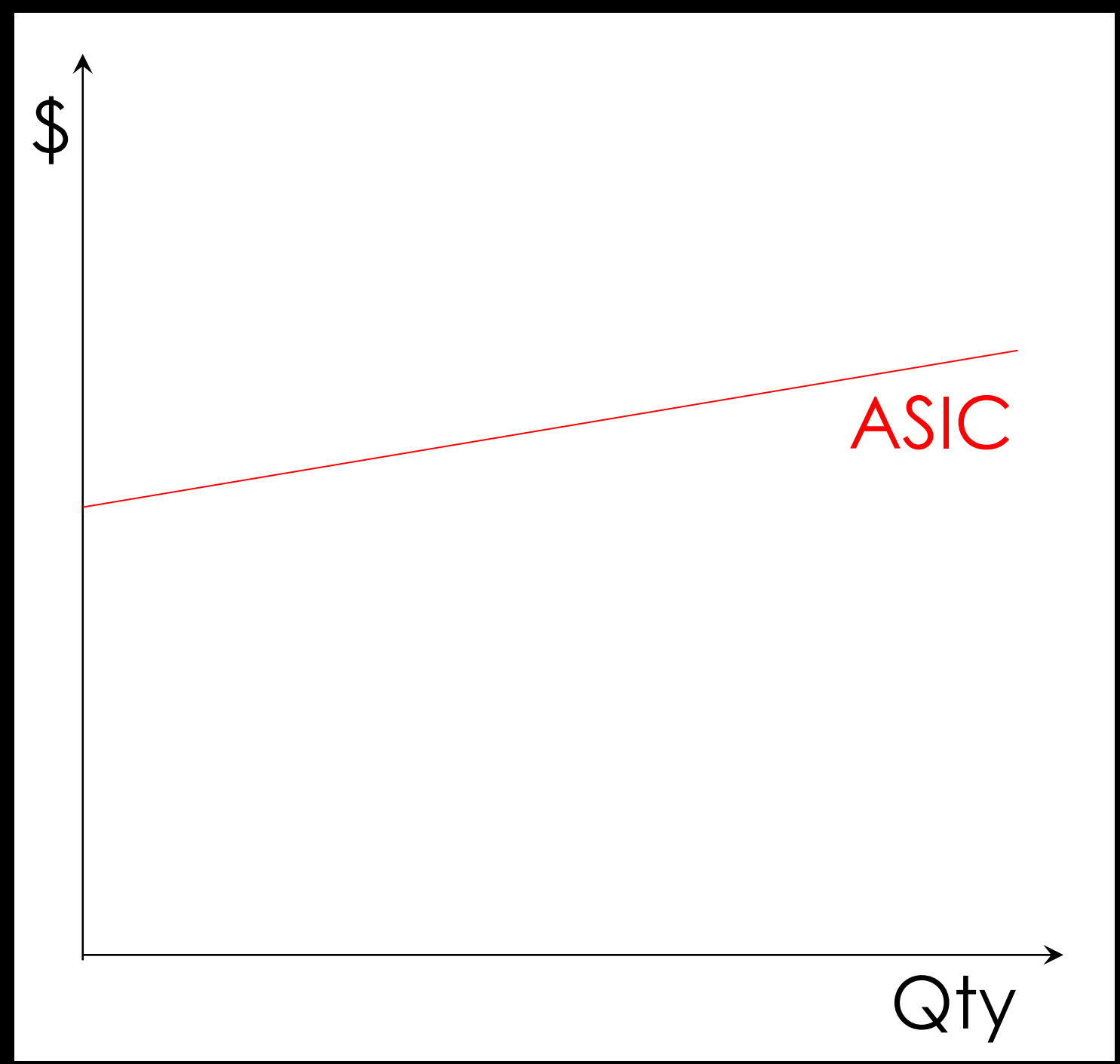


Application Specific Integrated
Circuit (ASIC)

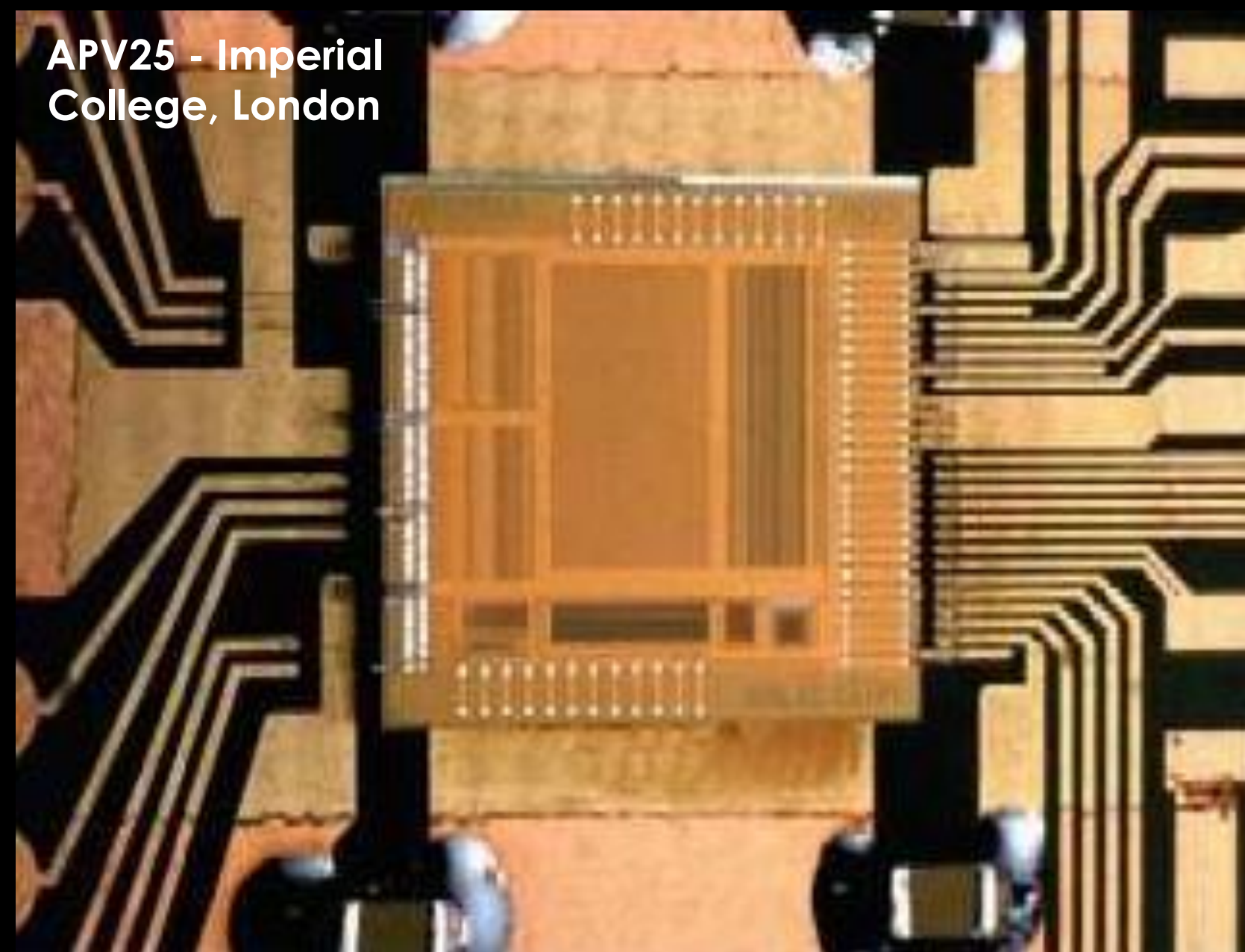
ASICs



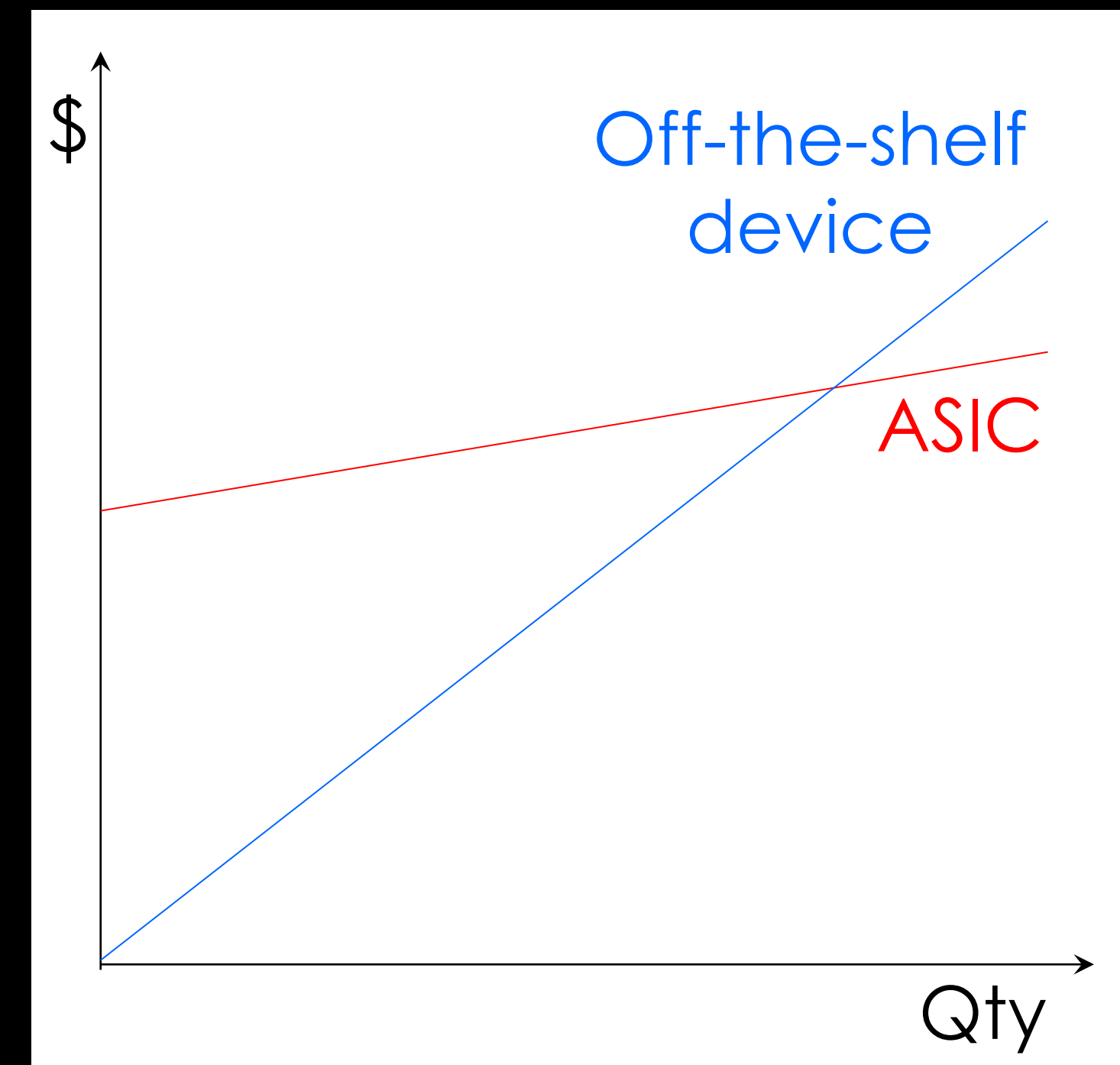
Application Specific Integrated Circuit (ASIC)



ASICs



Application Specific Integrated Circuit (ASIC)



TWO PHILOSOPHIES: SPACE VS. TIME

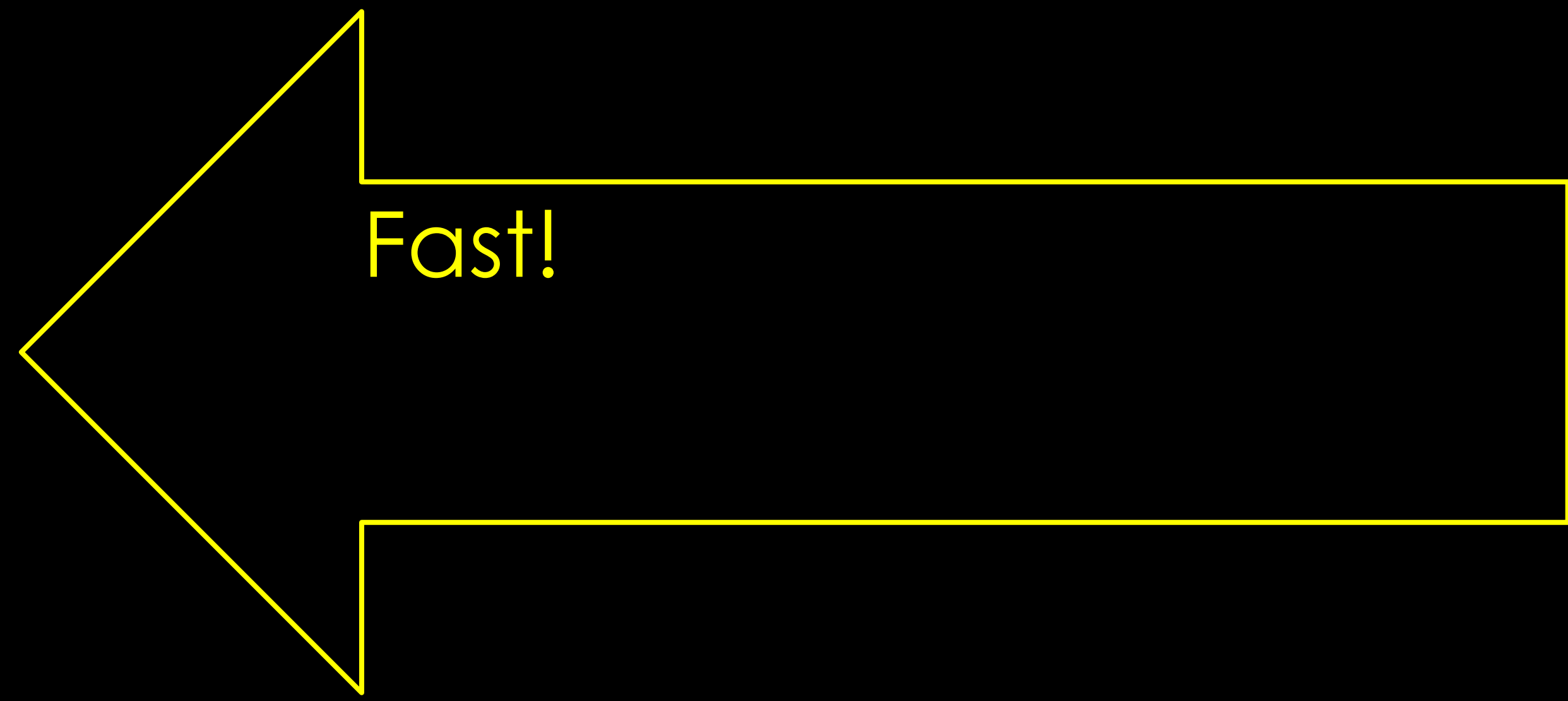
Have each operation performed by
dedicated logic
and
let signal propagate as a wave
through the logic

Combinatorial

TWO PHILOSOPHIES: SPACE VS. TIME

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

Combinatorial



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,
as a result, low throughput

TWO PHILOSOPHIES: SPACE VS. TIME

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

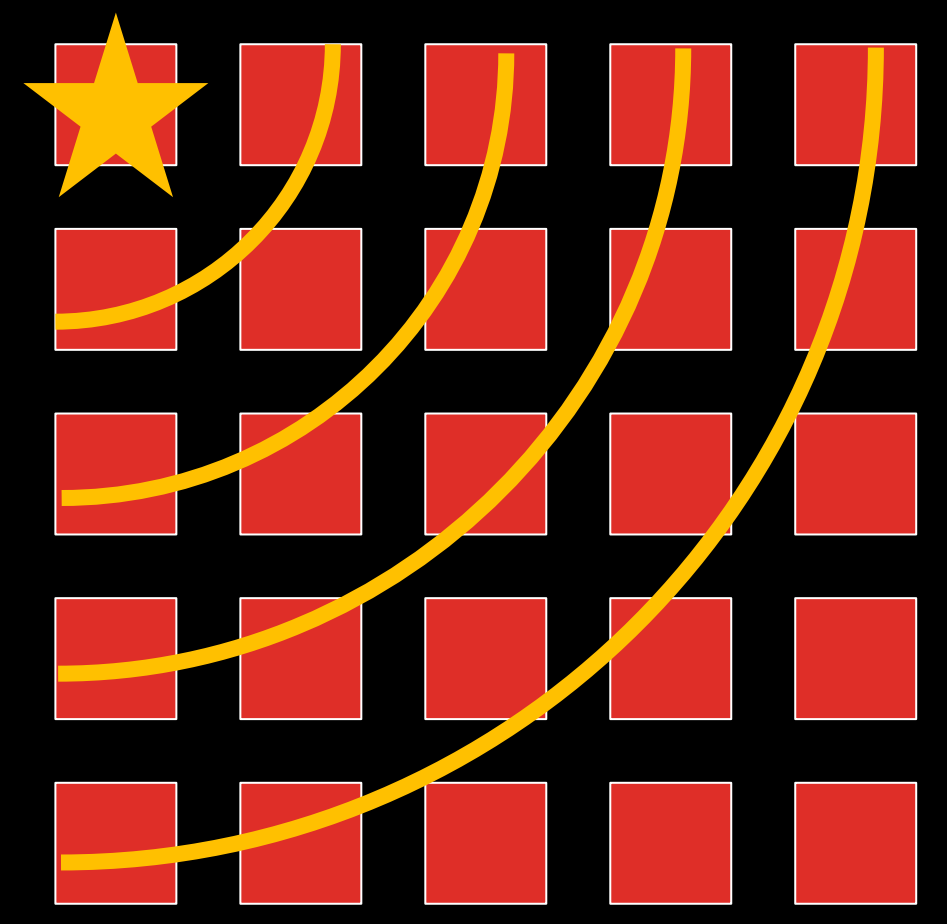
Slightly slower, but clean, easy to understand, scalable and high throughput

Pipelined/Parallel

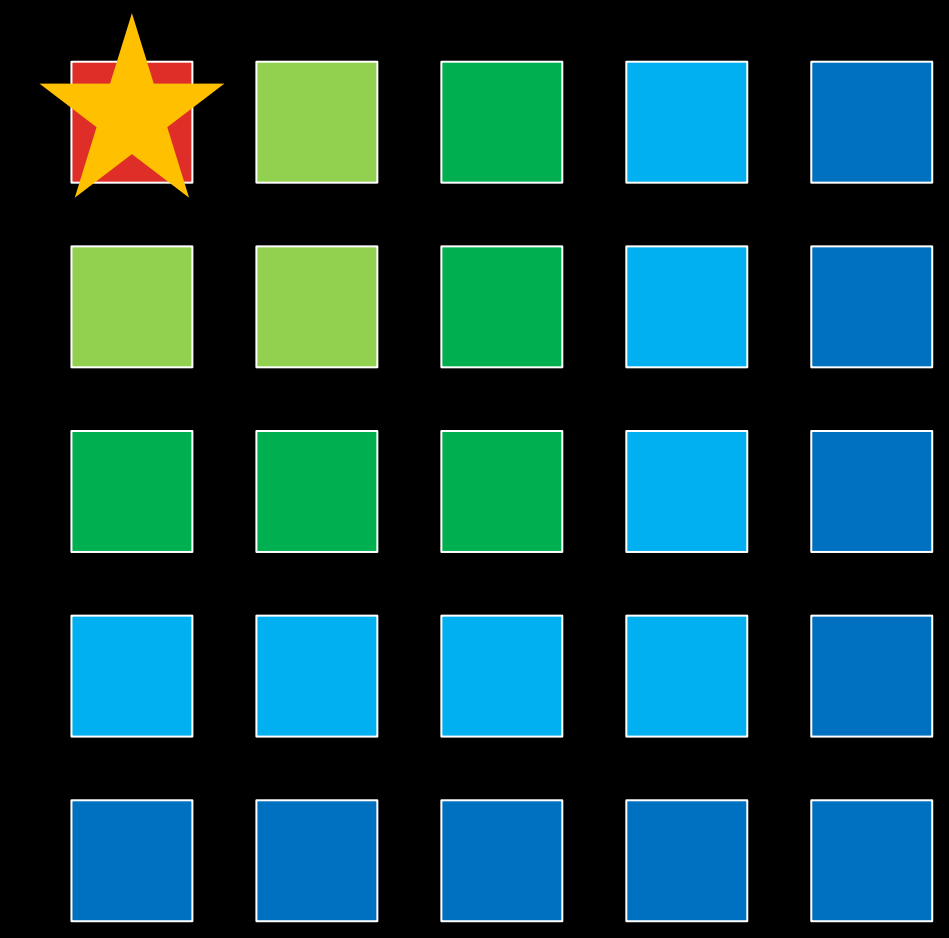
	6pm	7pm	8pm	9pm	10pm	11pm	12pm	01am	02am	03am
										
										
										
										

UP AGAINST THE SPEED OF LIGHT...

Combinatorial



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



- Do less each clock-cycle
- Reuse logic for different data on each clock-cycle
- Many steps needed for final result compensated for by much higher clock speeds

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

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

Pipelined/Parallel

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

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

Pipelined/Parallel

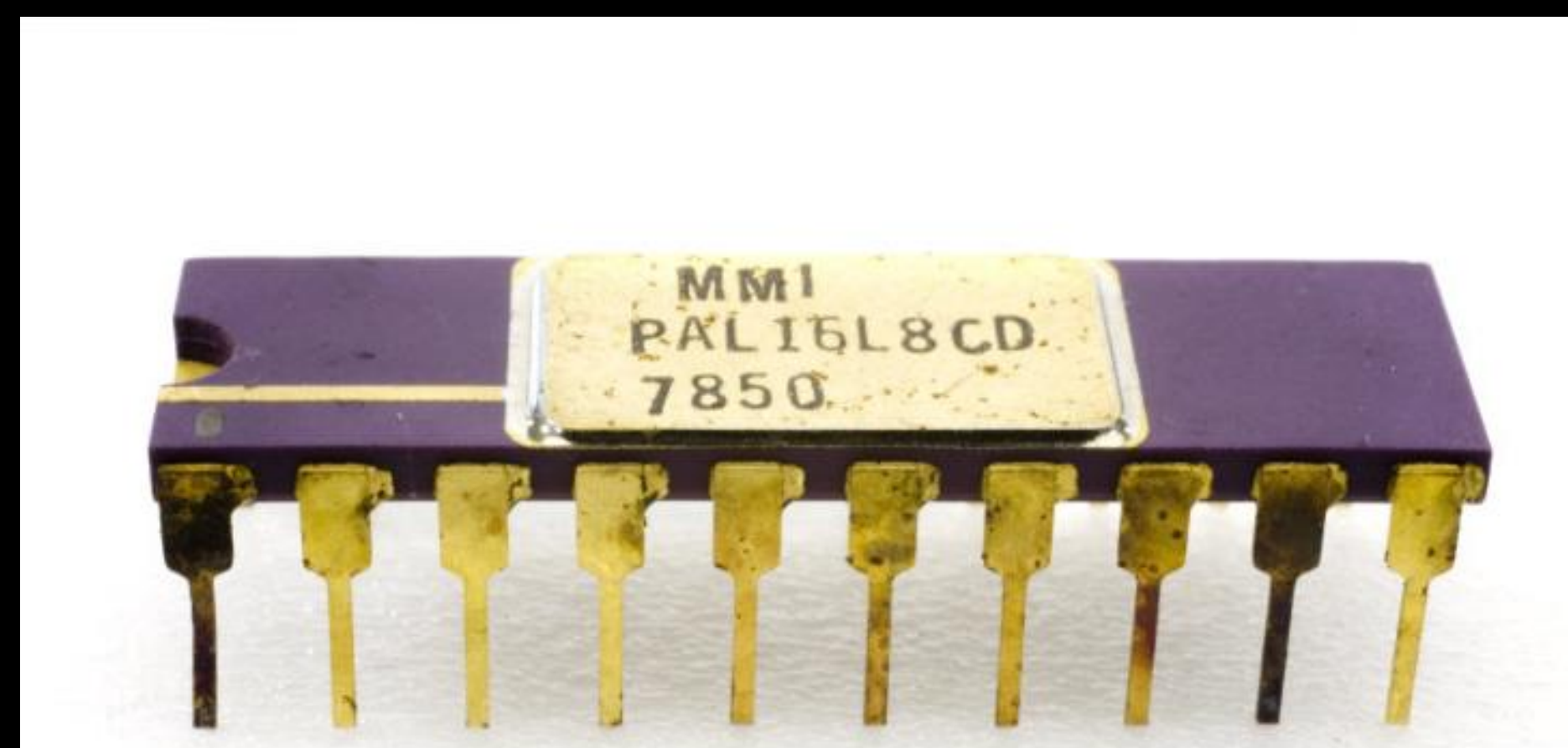
Sequential

“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

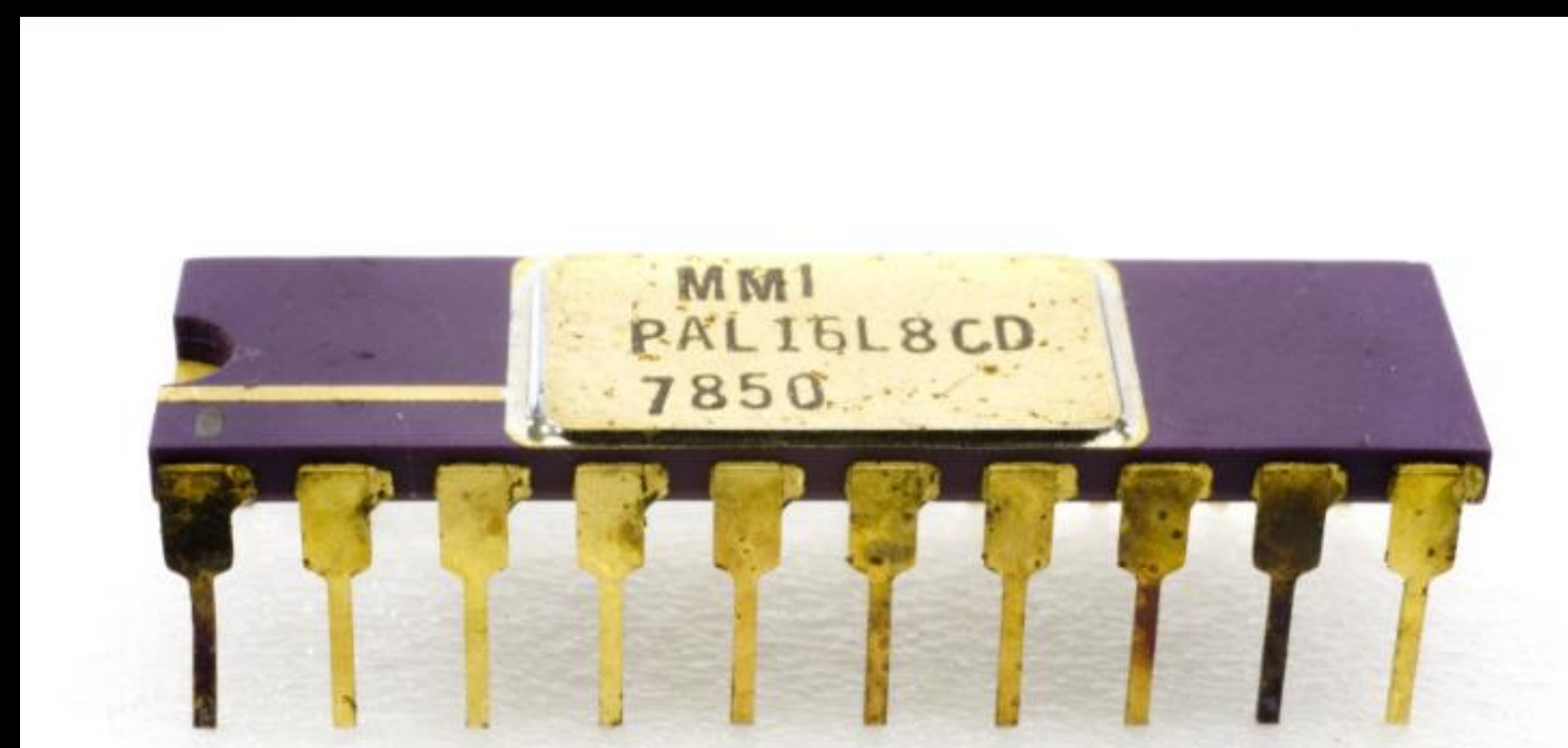


Sequential

Microprocessor

Perform all logical operations in one location, but sequentially

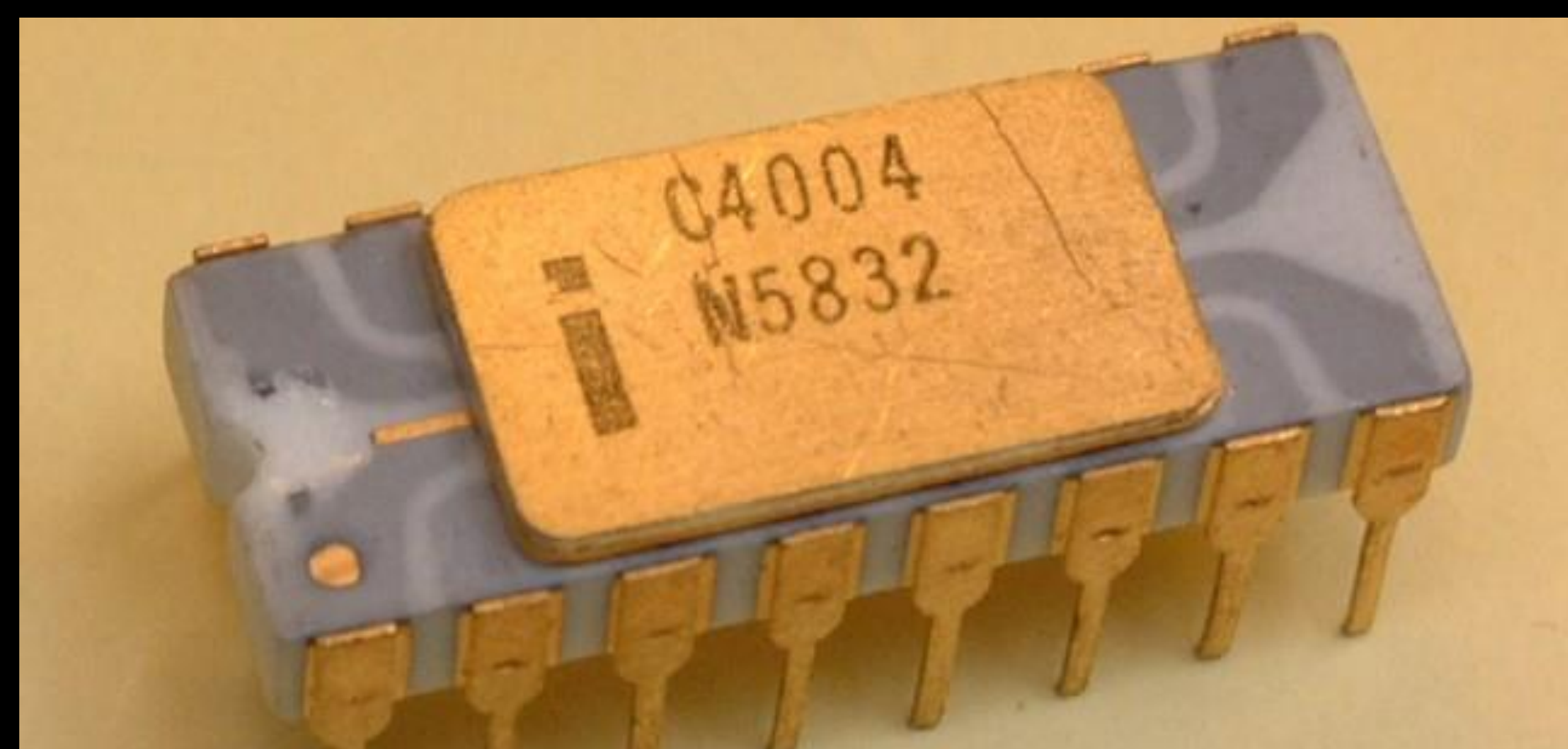
AND THE STORY DIVERGES...



Pipelined/Parallel

Programmable Array Logic

Pack entire logic circuits in a chip

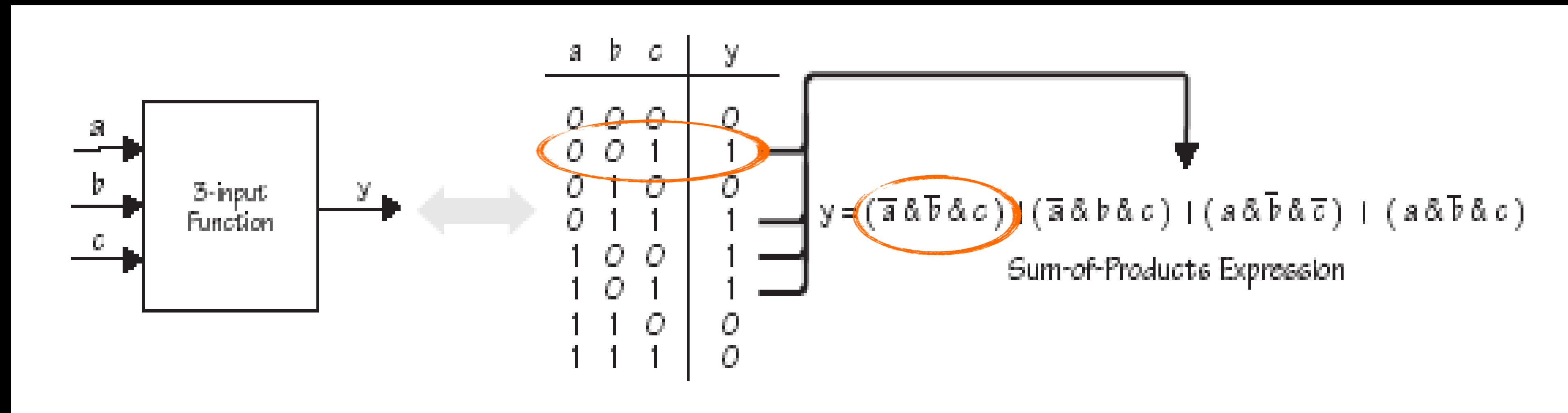


Sequential

No further discussion of microprocessors

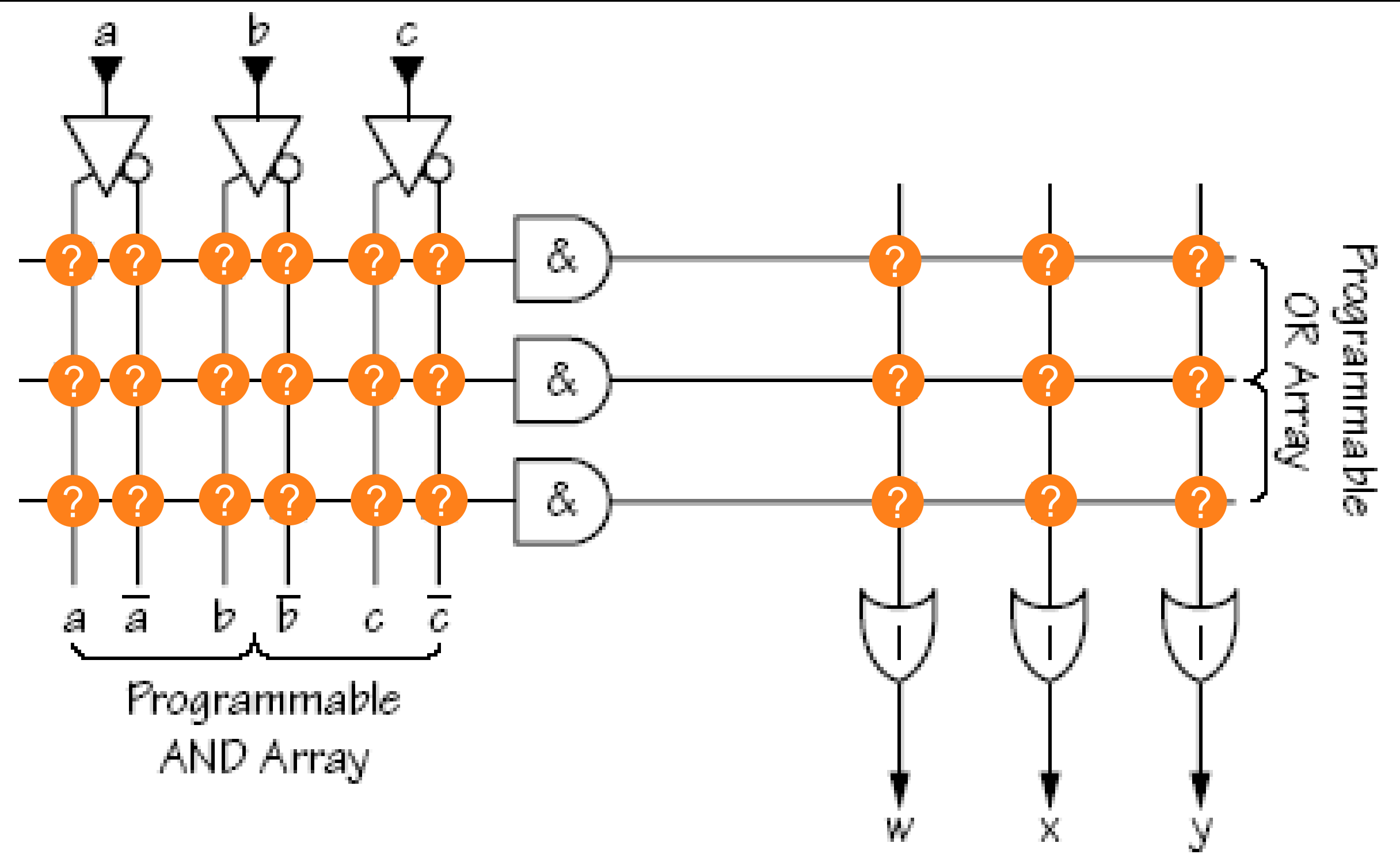
Perform all logical operations in one location, but sequentially

SUM-OF-PRODUCTS THEOREM



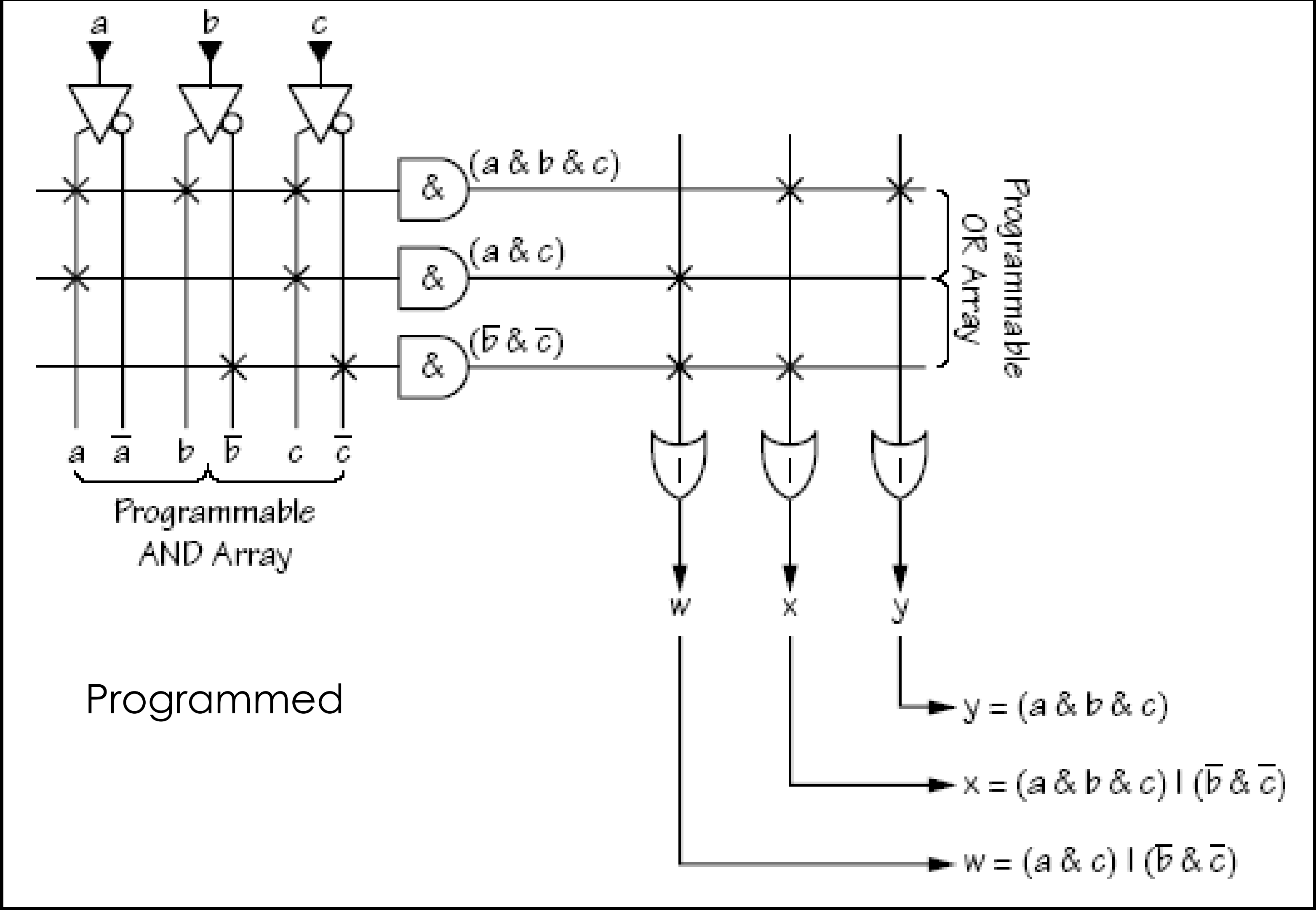
- Any Boolean operation may be expressed as
 the OR of AND operations (Sum of products form)
- Or
 the AND of OR operations (Product of sums form)

PROGRAMMABLE LOGIC DEVICES (PLDS)



Unprogrammed

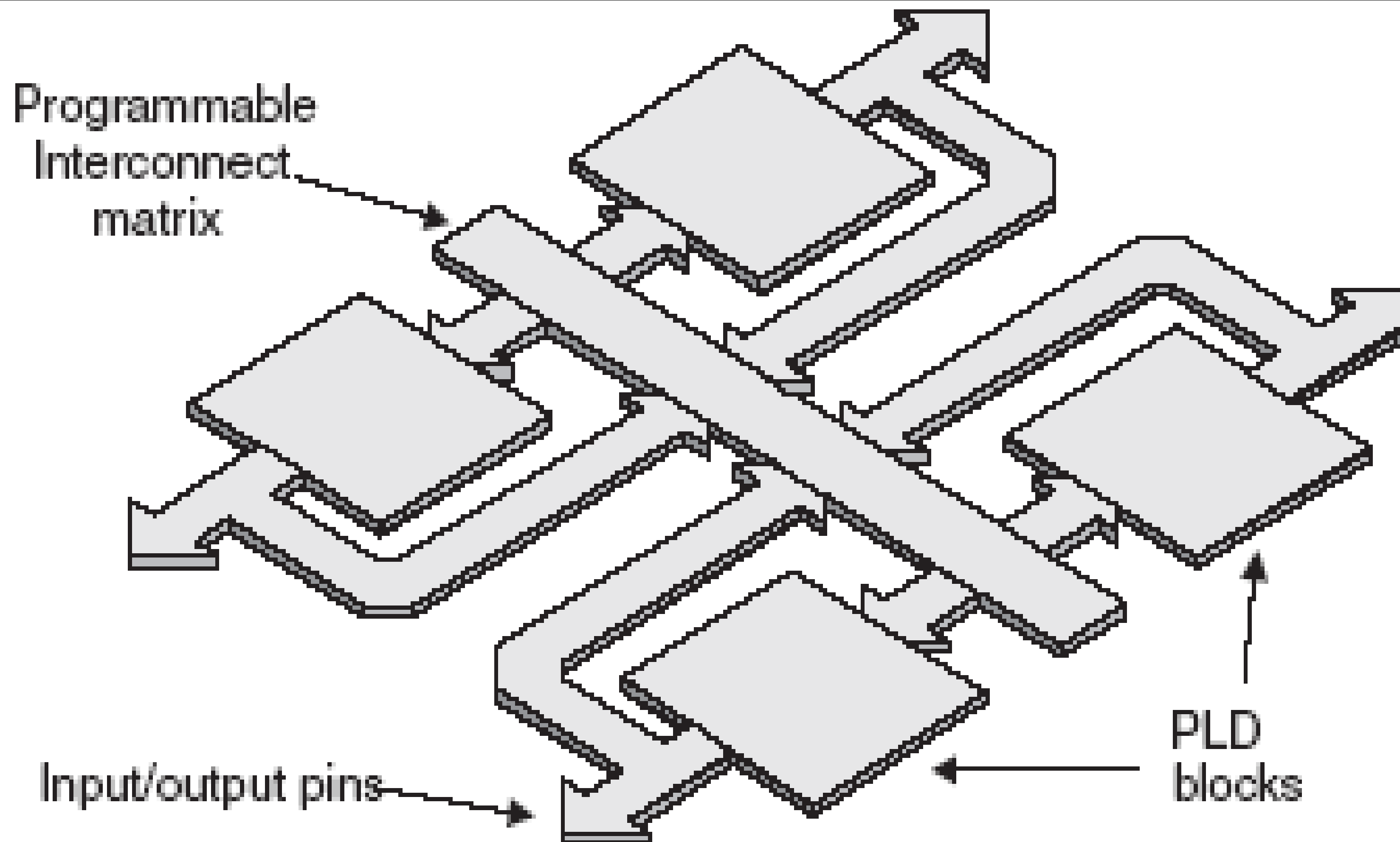
PROGRAMMABLE LOGIC DEVICES (PLDs)



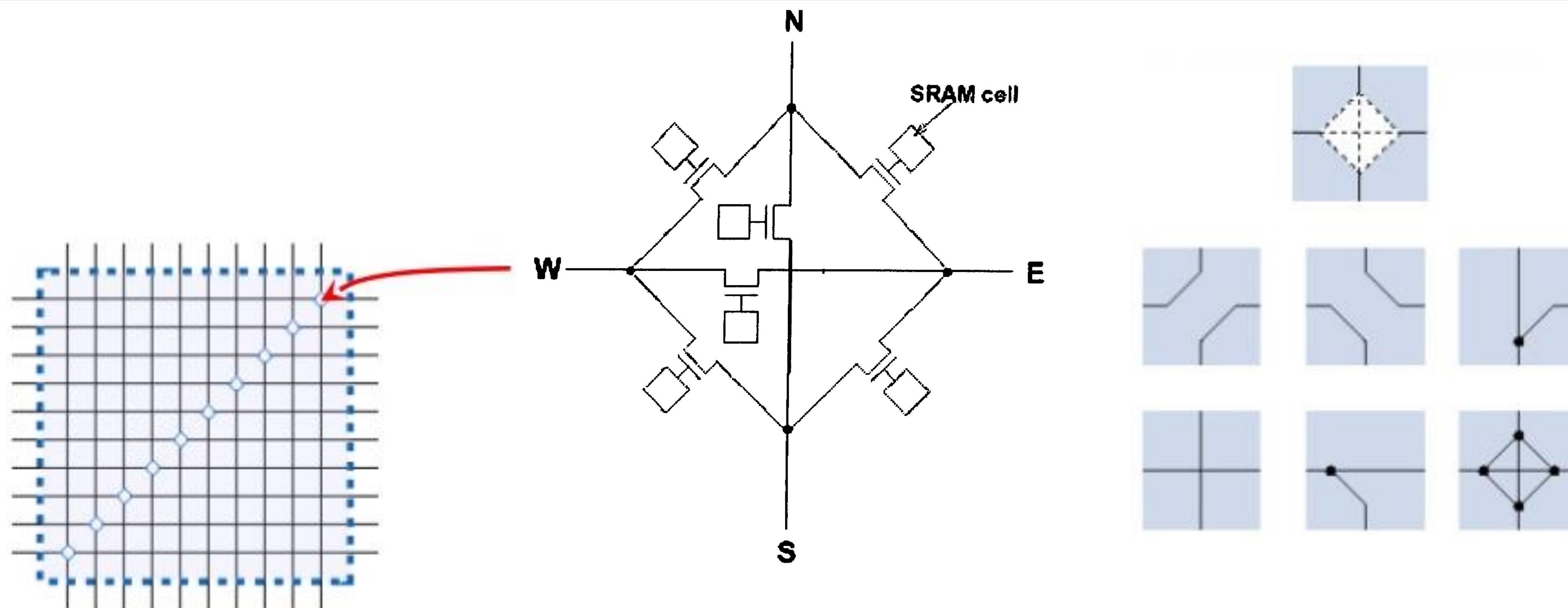
PROGRAMMABLE LOGIC DEVICES (PLDS)

- 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

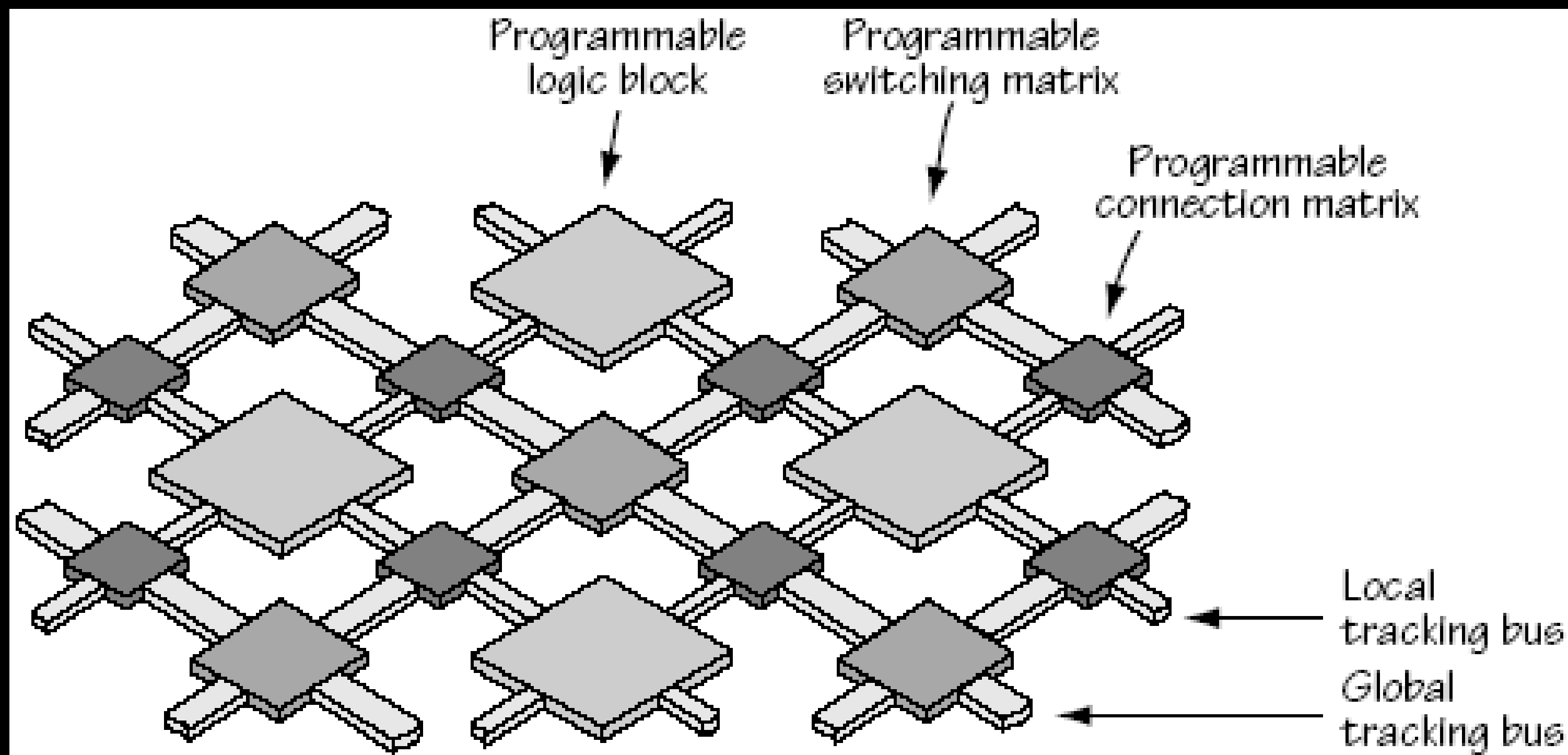
AN ALTERNATIVE APPROACH

- 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

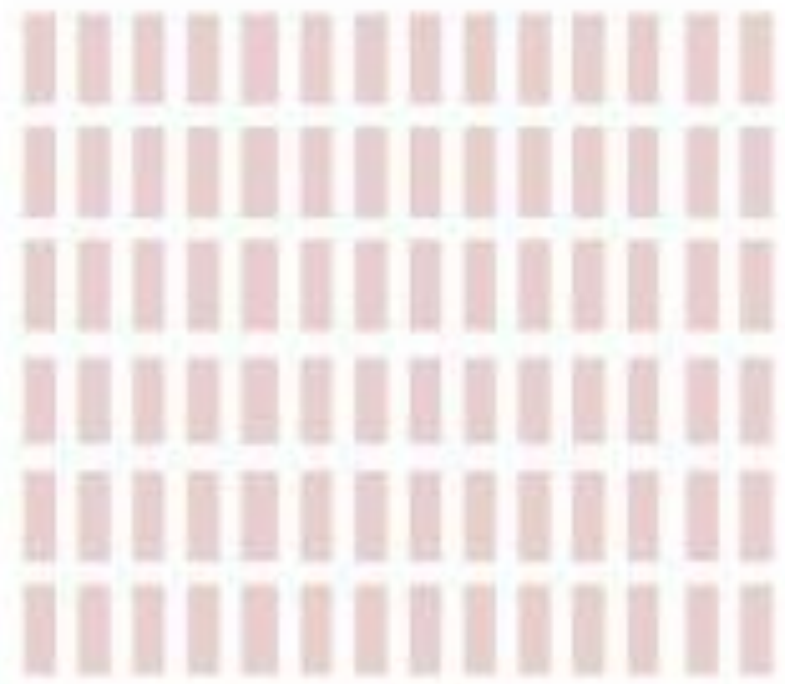
The Field Programmable Gate Array
(FPGA)

FIELD PROGRAMMABLE GATE ARRAYS (FPGAs)

- 'Simple' Programmable Logic Blocks
- Massive Fabric of Programmable Interconnects



1985-1992



Logic

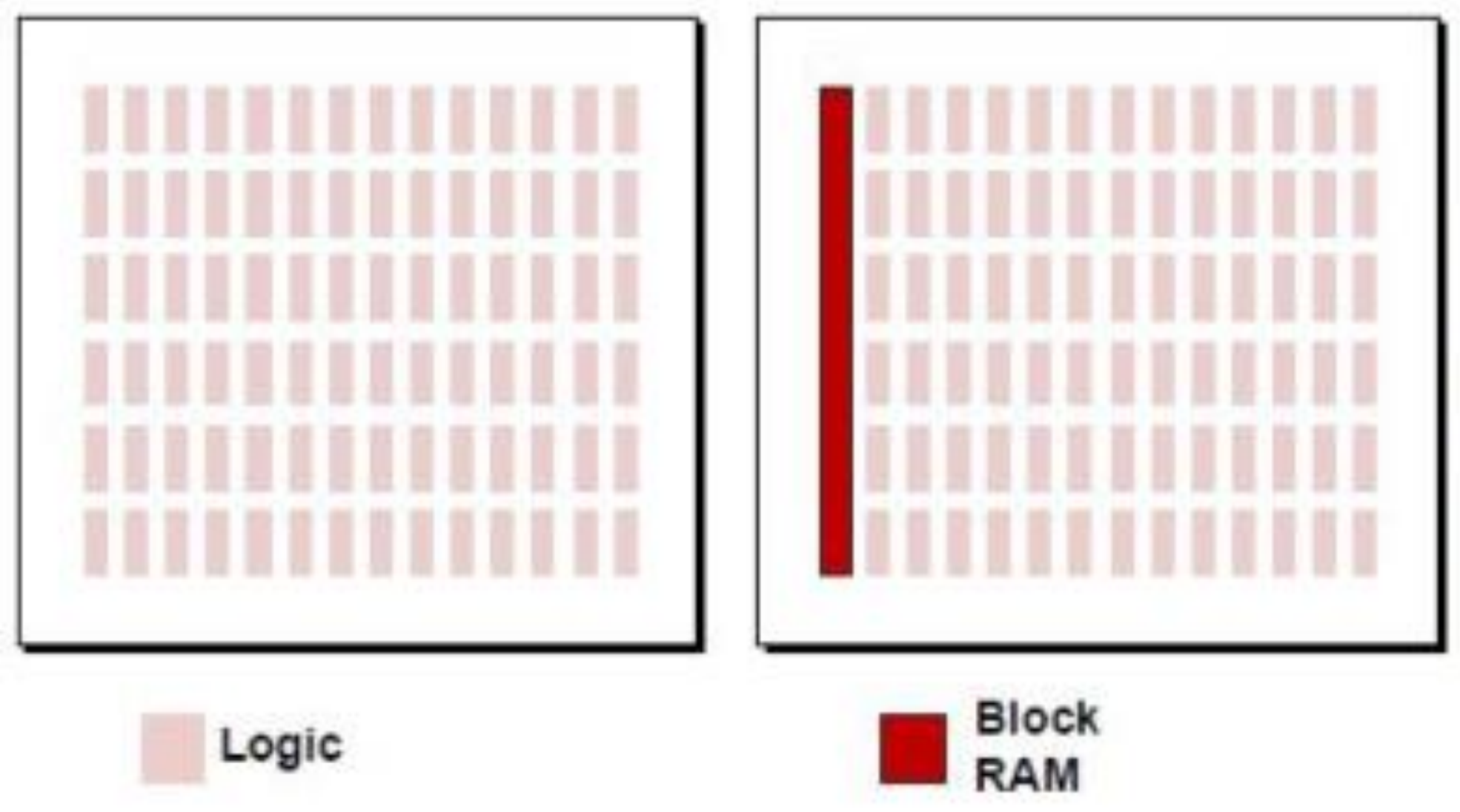
EVOLUTION OF FEATURES IN FPGAS

1985-1992

1992-2000

37

EVOLUTION OF FEATURES IN FPGAS

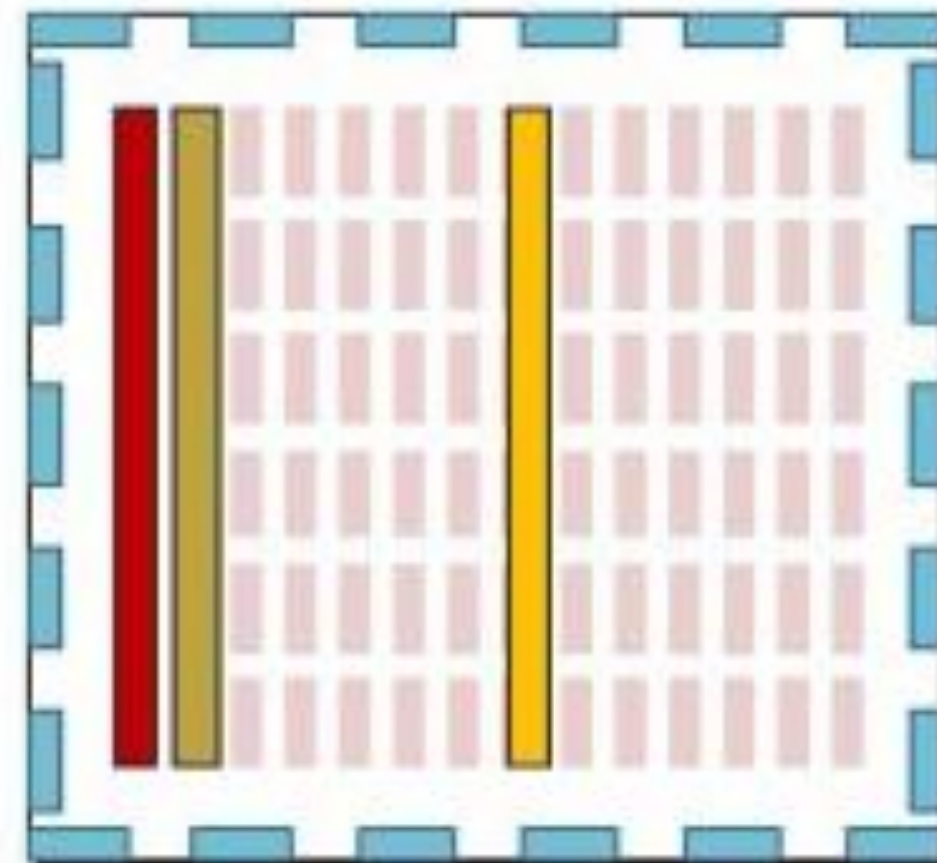
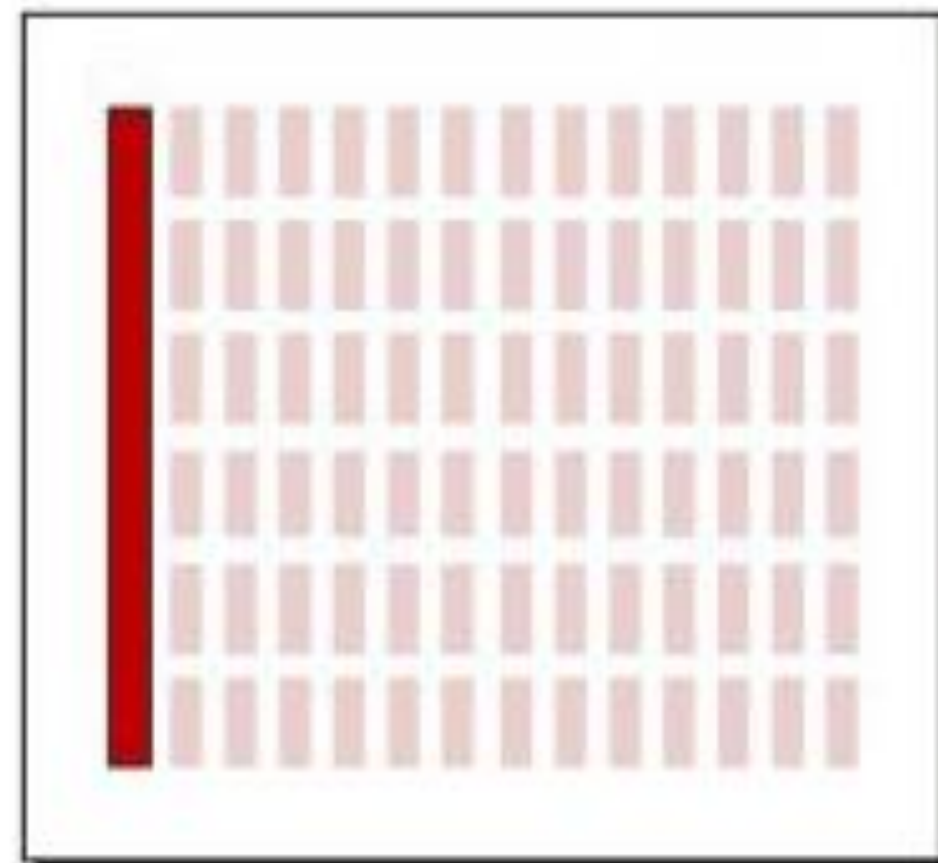
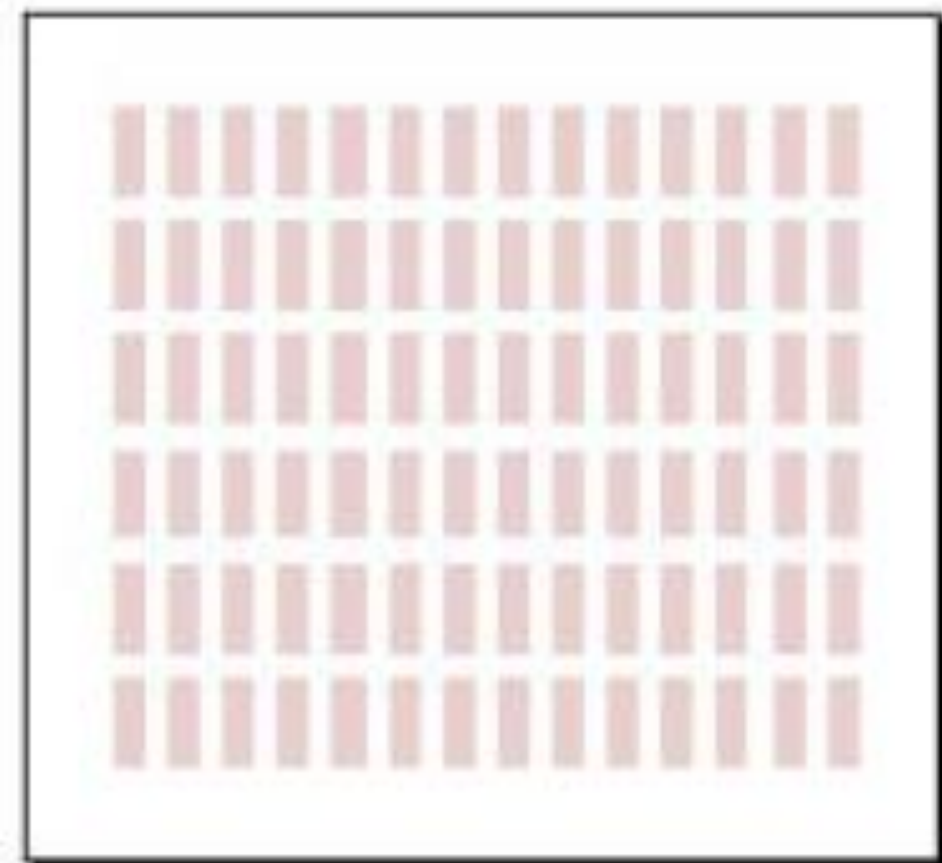


Who wants to waste all the LUTs as RAM?

1985-1992

1992-2000

2000 -2002



Logic

Block RAM

MAC Units

Programmable IO

Clock Management Unit

EVOLUTION OF FEATURES IN FPGAS

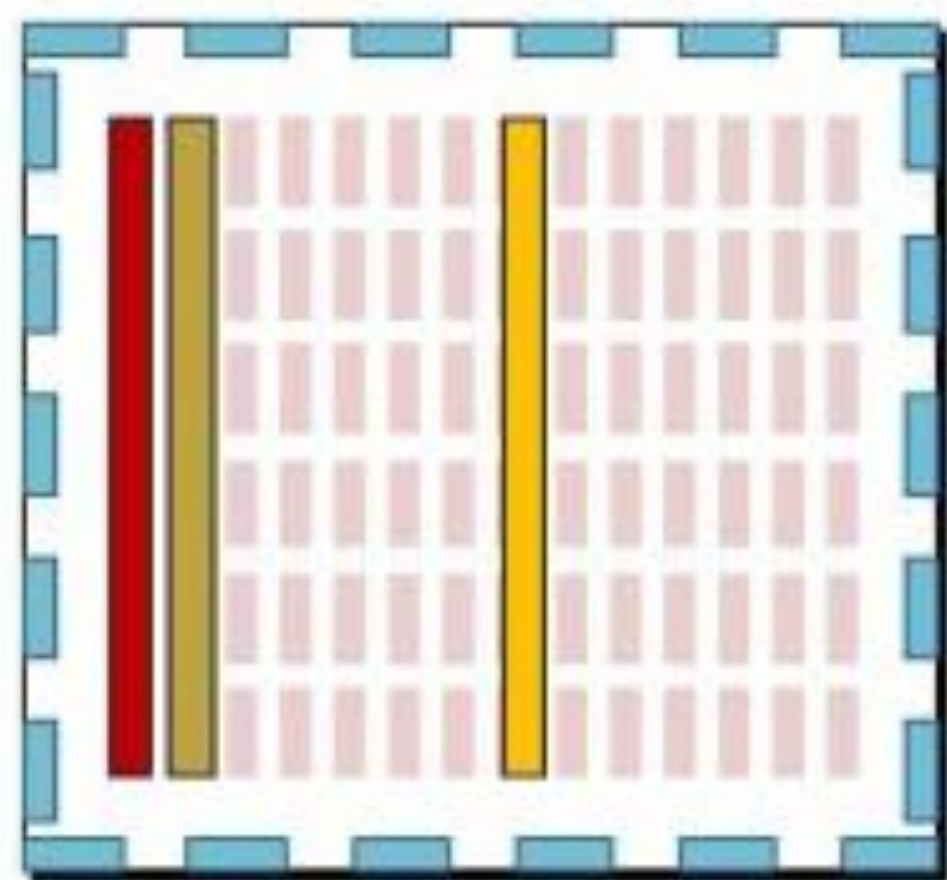
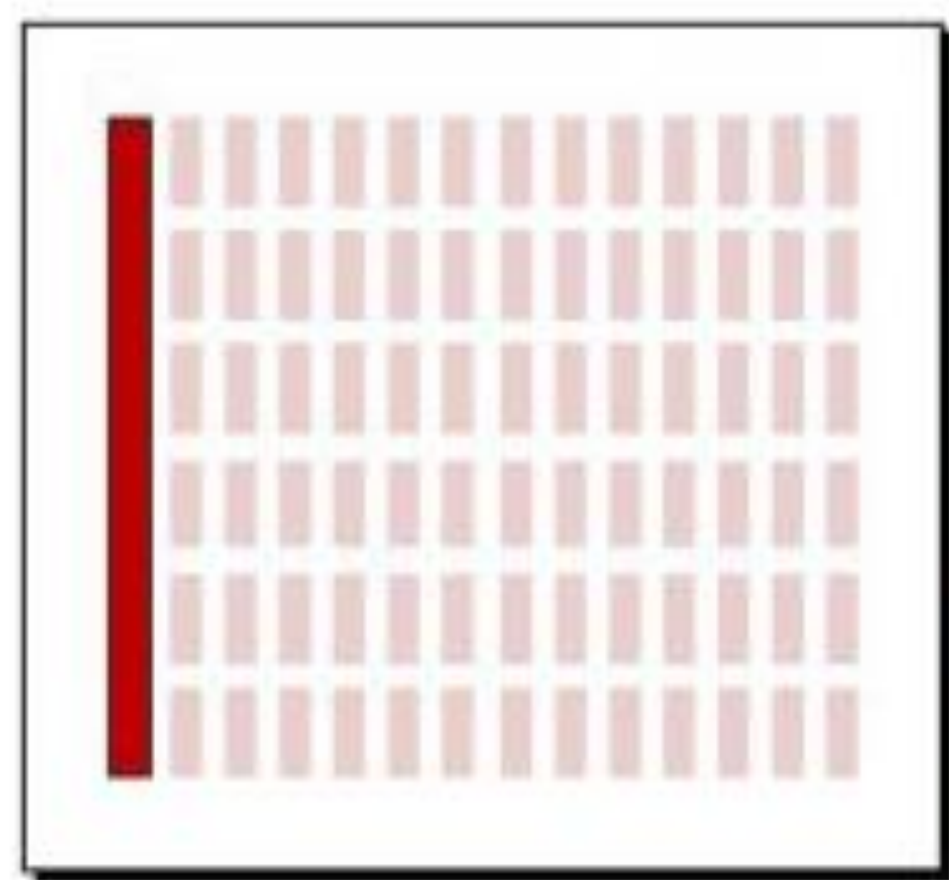
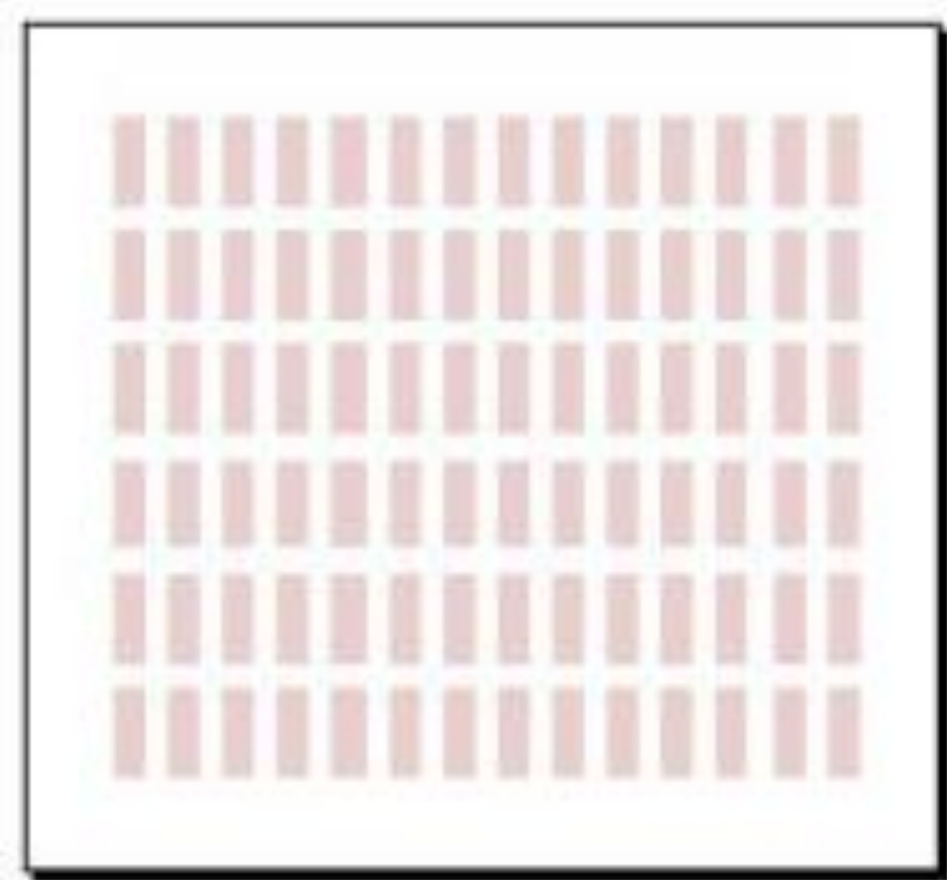
Who wants to waste all the LUTs for multiplication?
Big chips need dedicated clocking!

1985-1992

1992-2000

2000 -2002

EVOLUTION OF FEATURES IN FPGAS



Logic

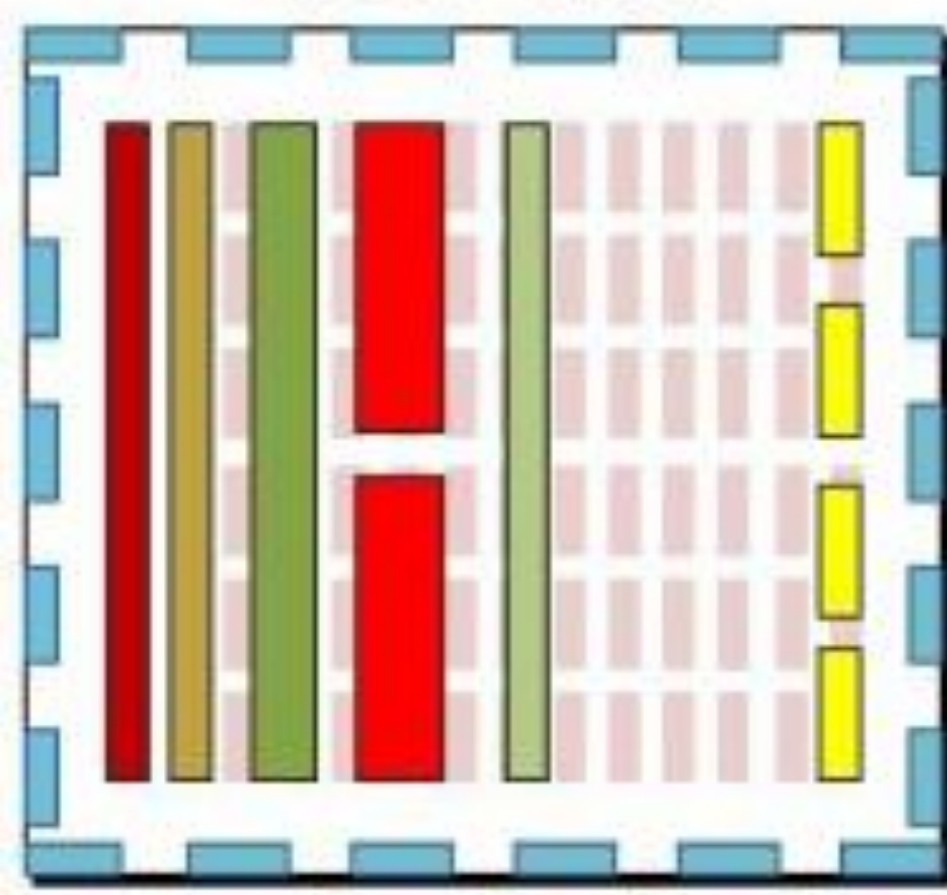
Block RAM

MAC Units

Programmable IO

Clock Management Unit

2002 - 2004



DSP Slices Micro-Processor

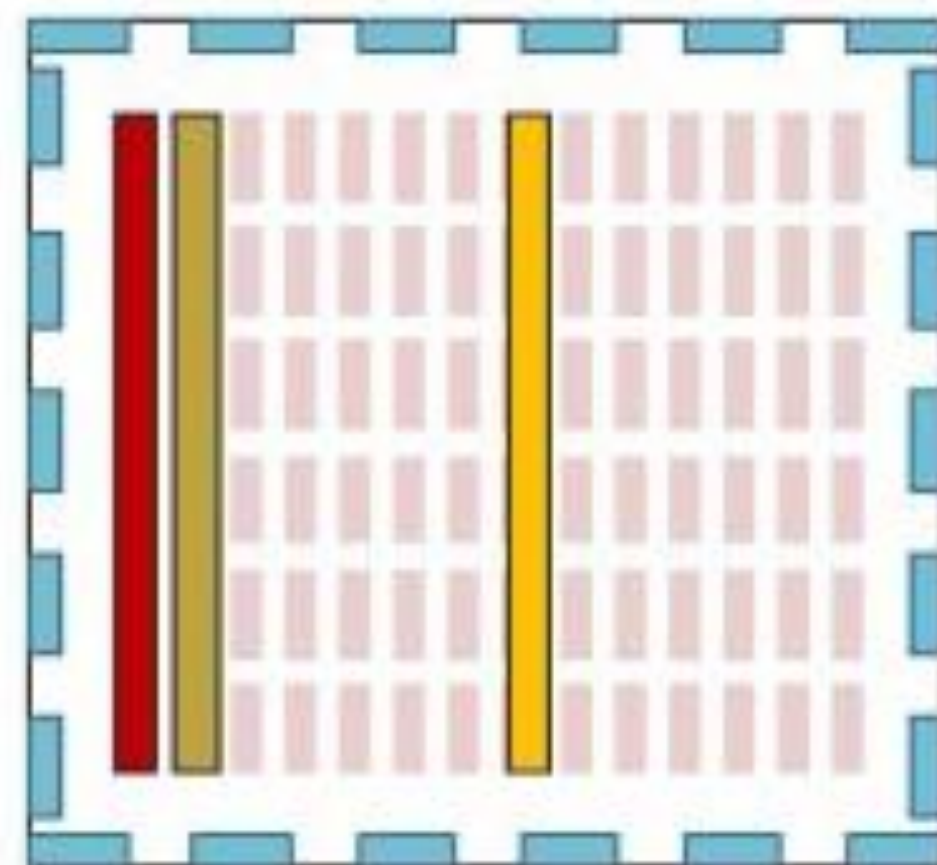
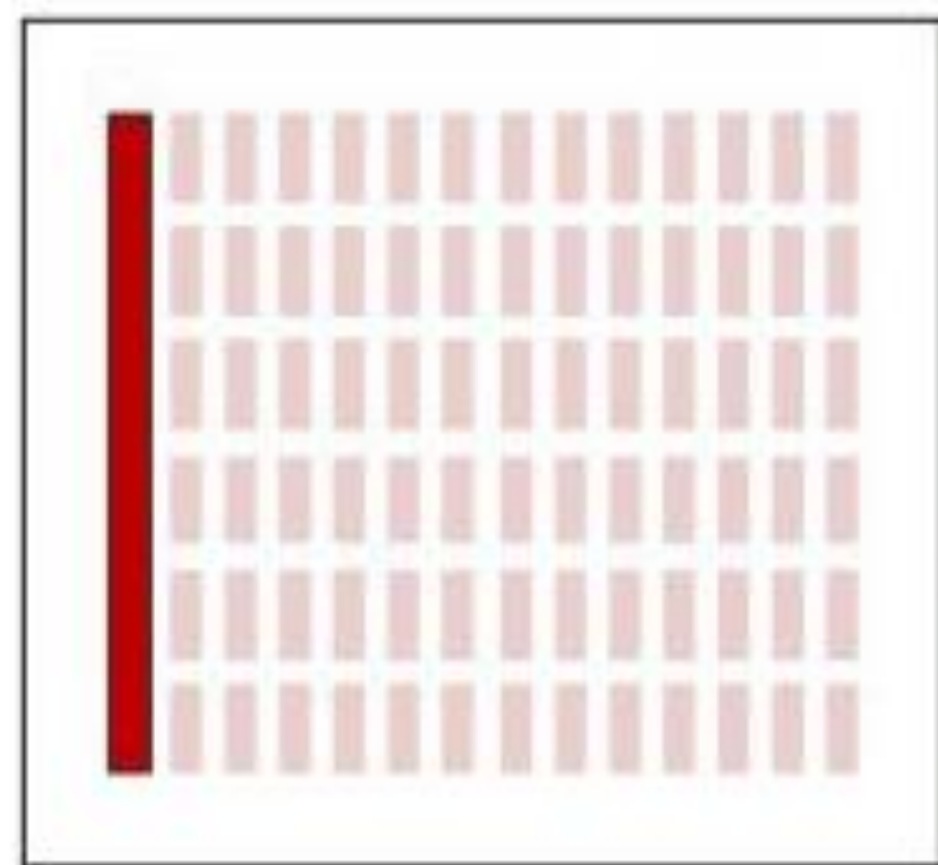
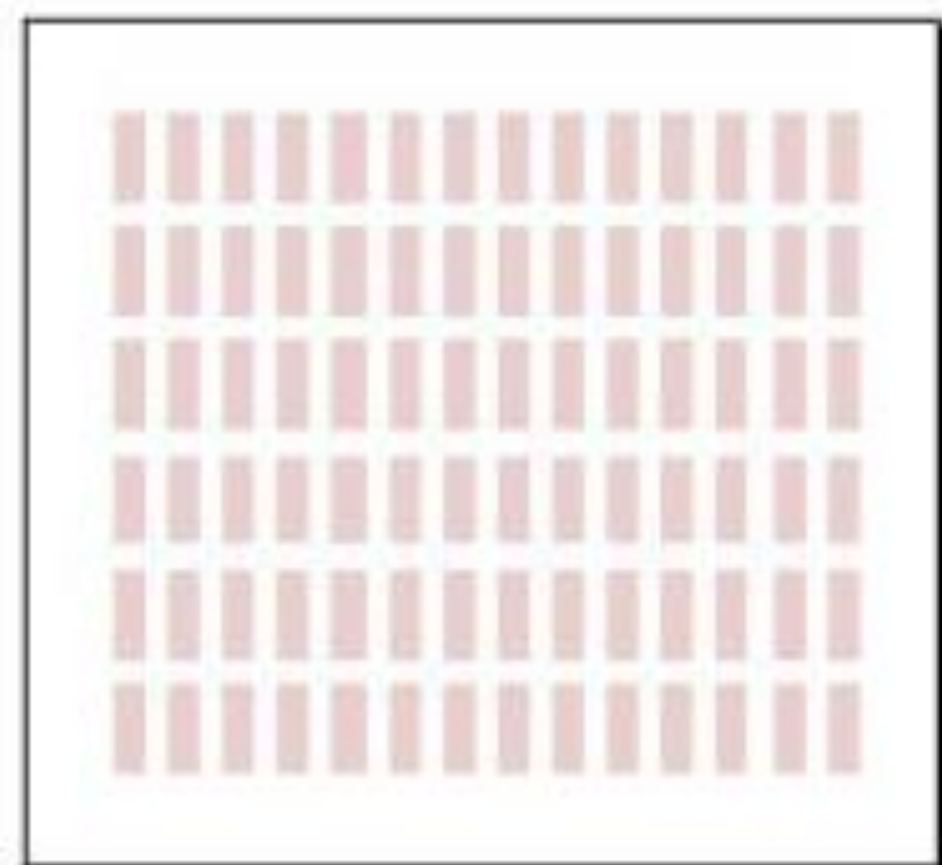
Mult-Gigabit Transceivers

1985-1992

1992-2000

2000 -2002

EVOLUTION OF FEATURES IN FPGAS



Logic

Block RAM

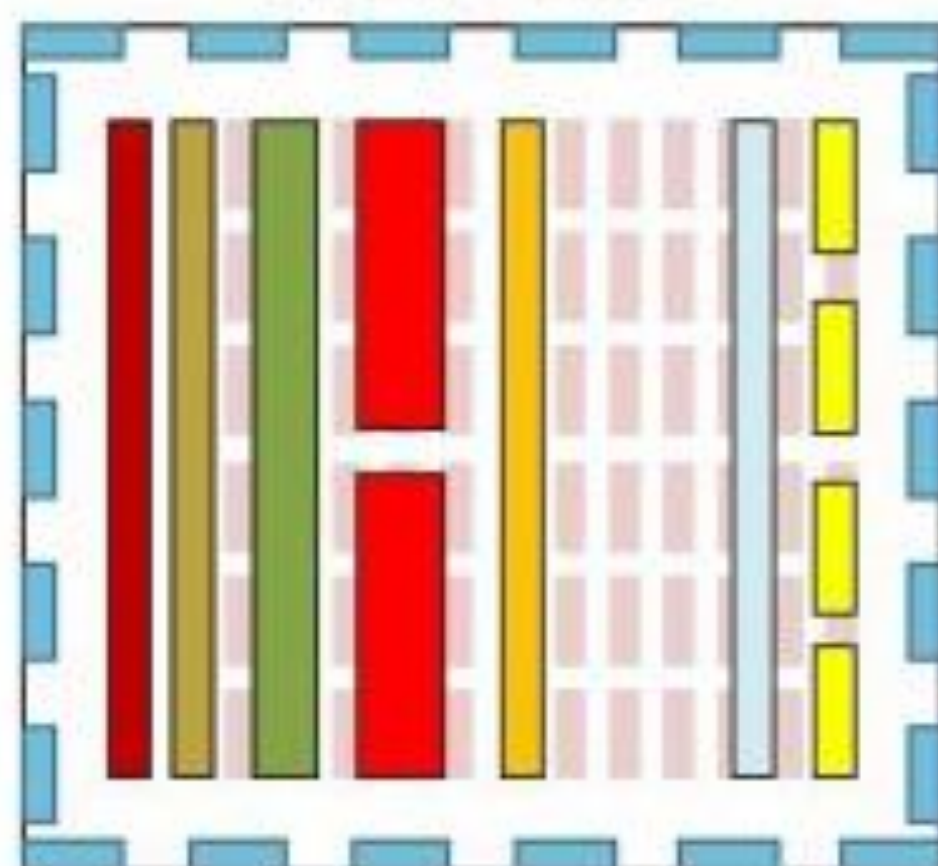
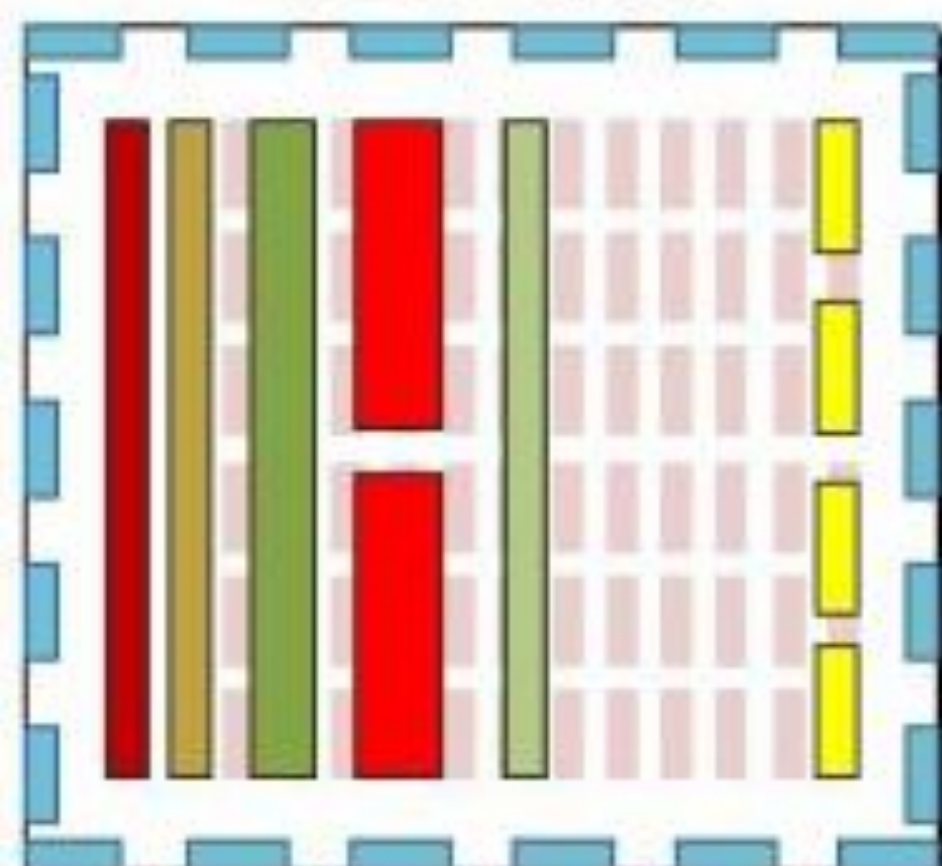
MAC Units

Programmable IO

Clock Management Unit

2002 - 2004

2004 - 2005



DSP Slices

Micro-Processor

Ethernet MAC

Mult-Gigabit Transceivers

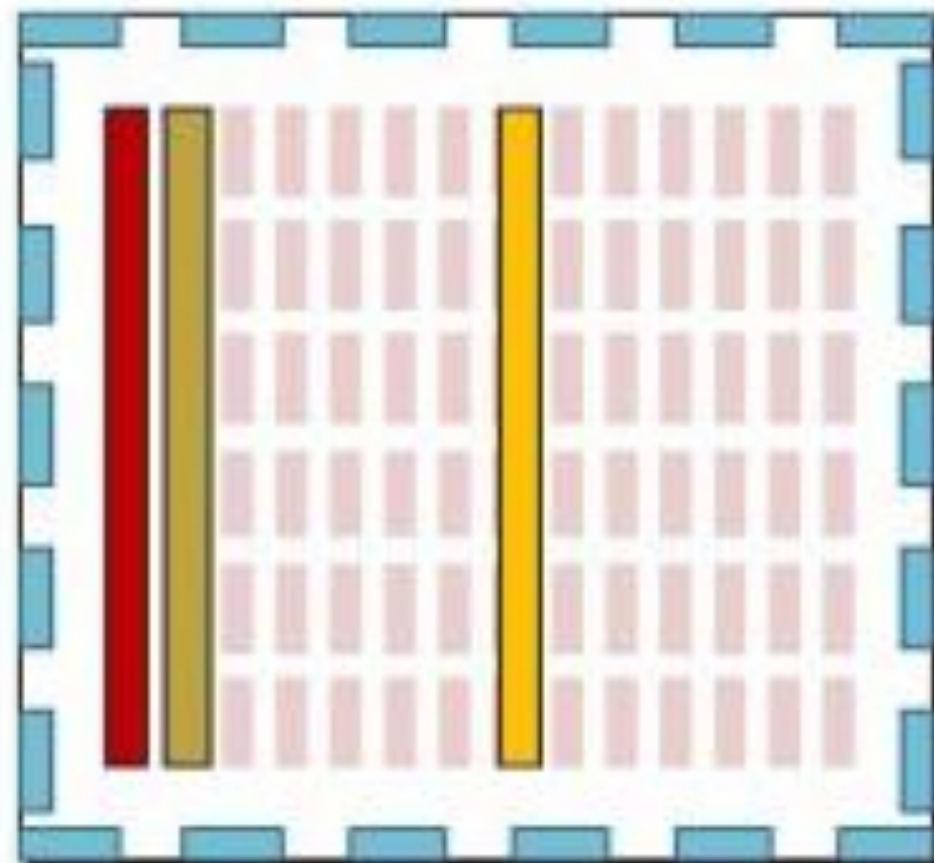
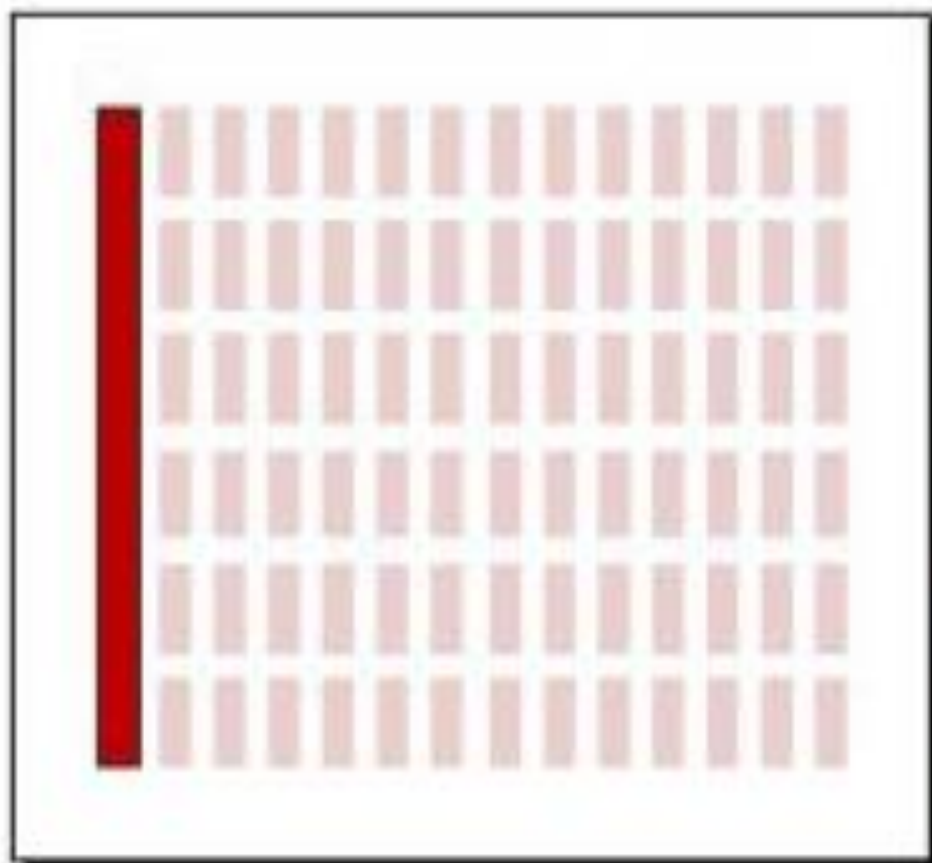
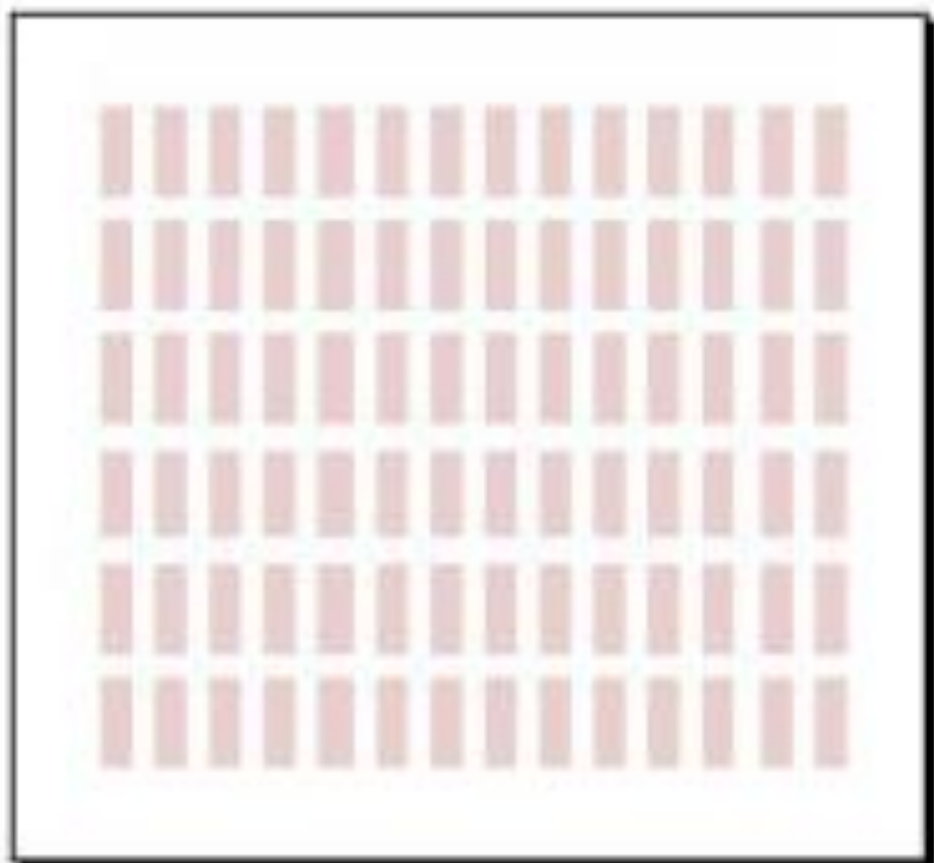
Who wants to waste LUTs AND re-invent industry-standard blocks?

1985-1992

1992-2000

2000 -2002

EVOLUTION OF FEATURES IN FPGAS



Logic

Block RAM

MAC Units

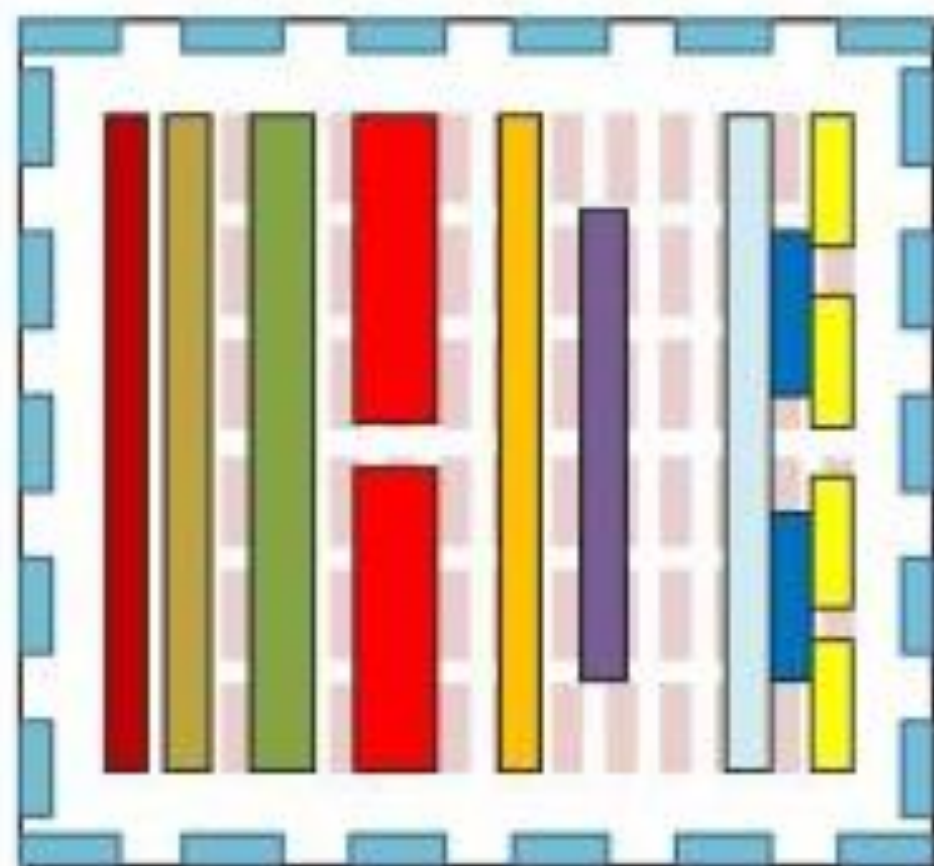
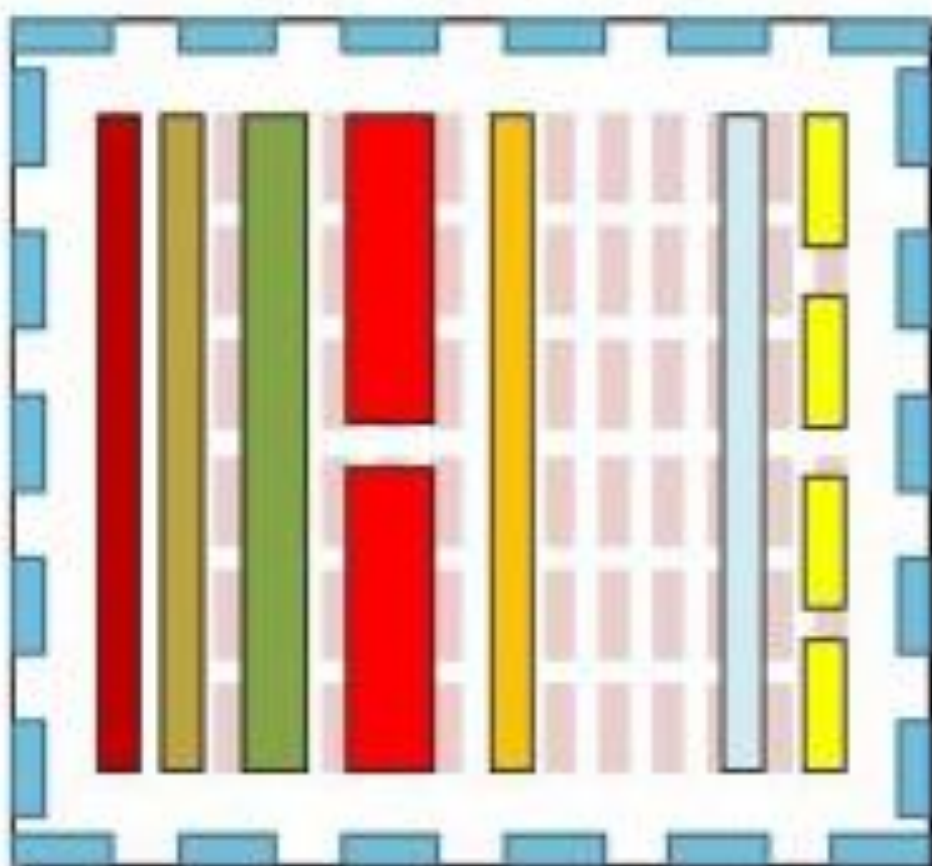
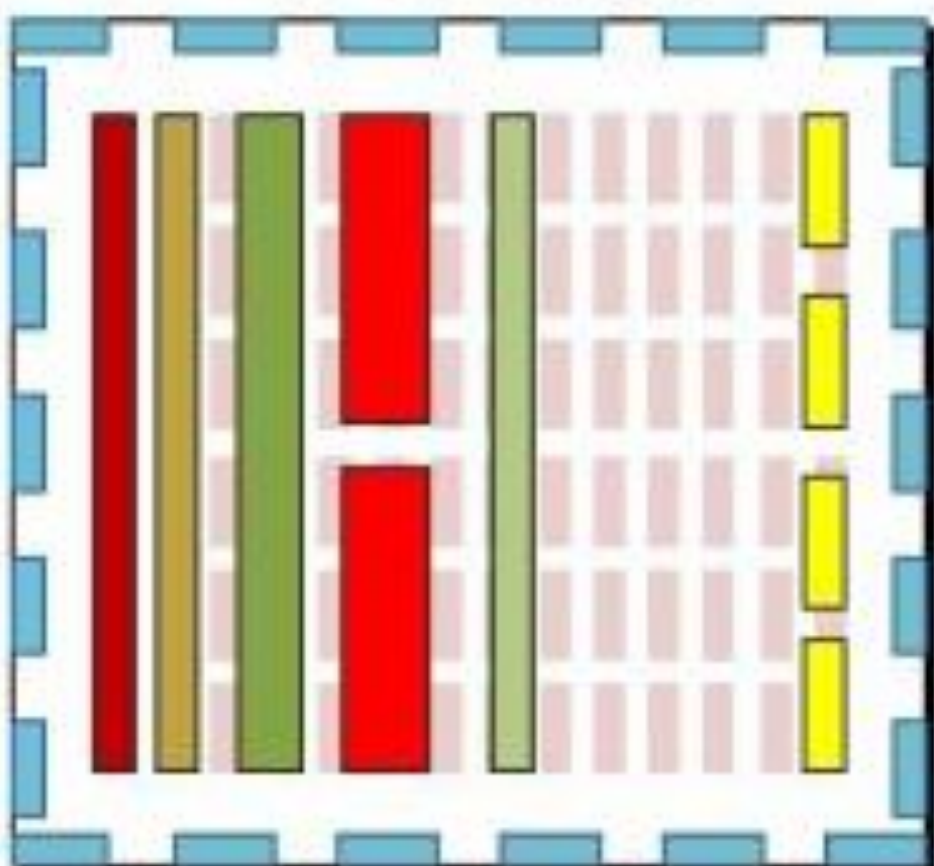
Programmable IO

Clock Management Unit

2002 - 2004

2004 - 2005

2005 - 2009



DSP Slices

Micro-Processor

Ethernet MAC

PCI Interface

System Monitor

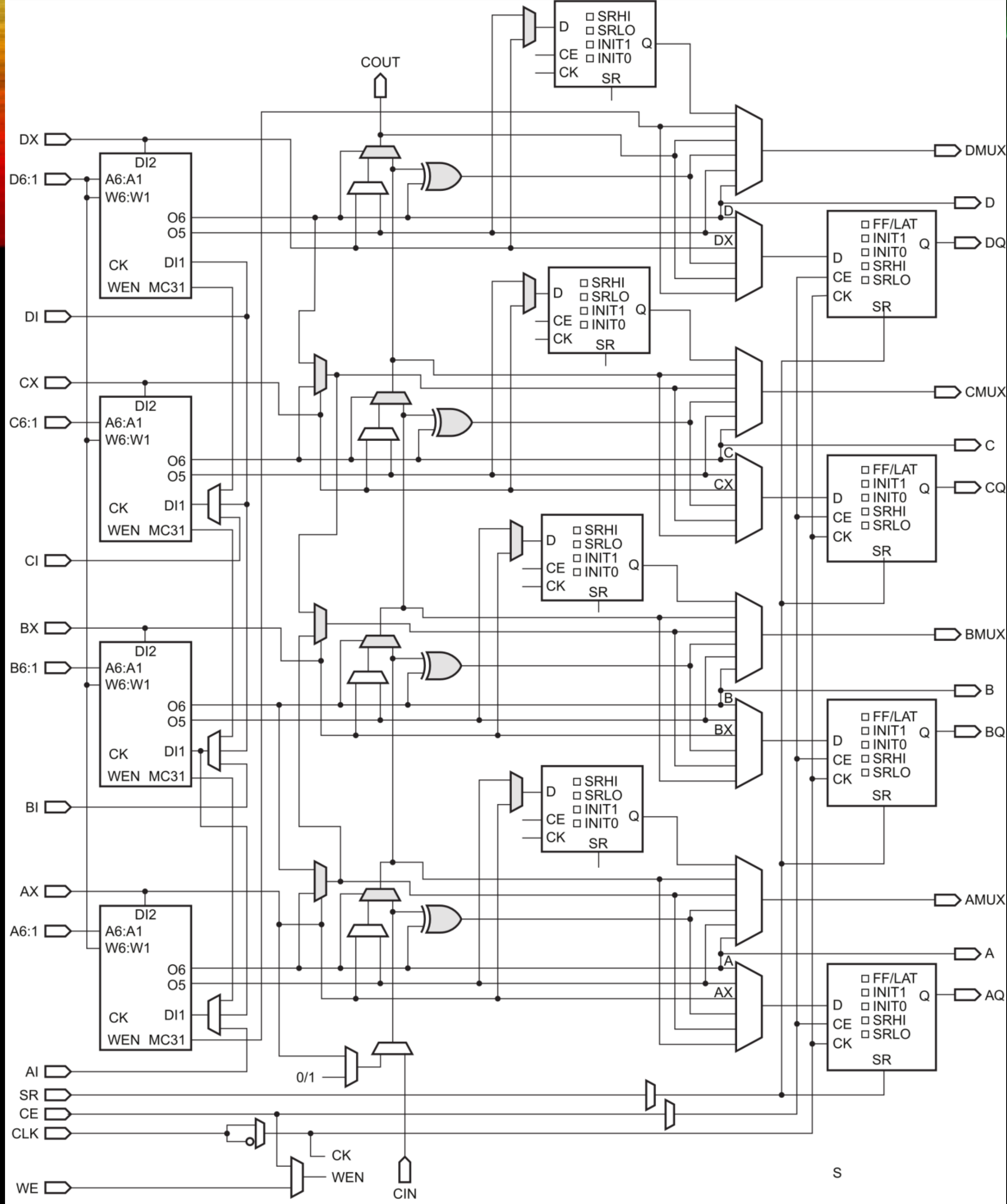
Who wants to waste LUTs AND re-invent industry-standard blocks?

Mult-Gigabit Transceivers

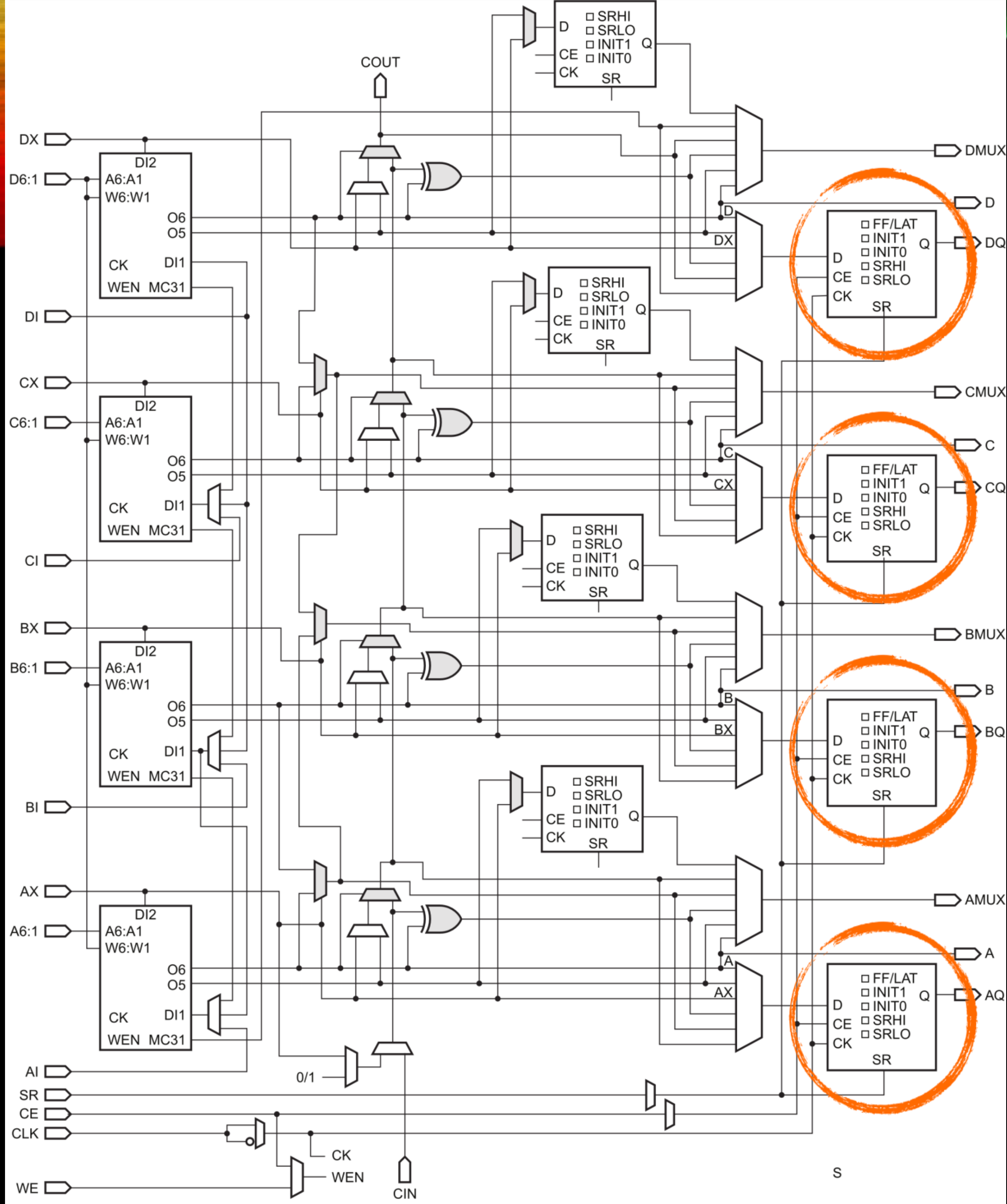
A NOTE ON I/O

- 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



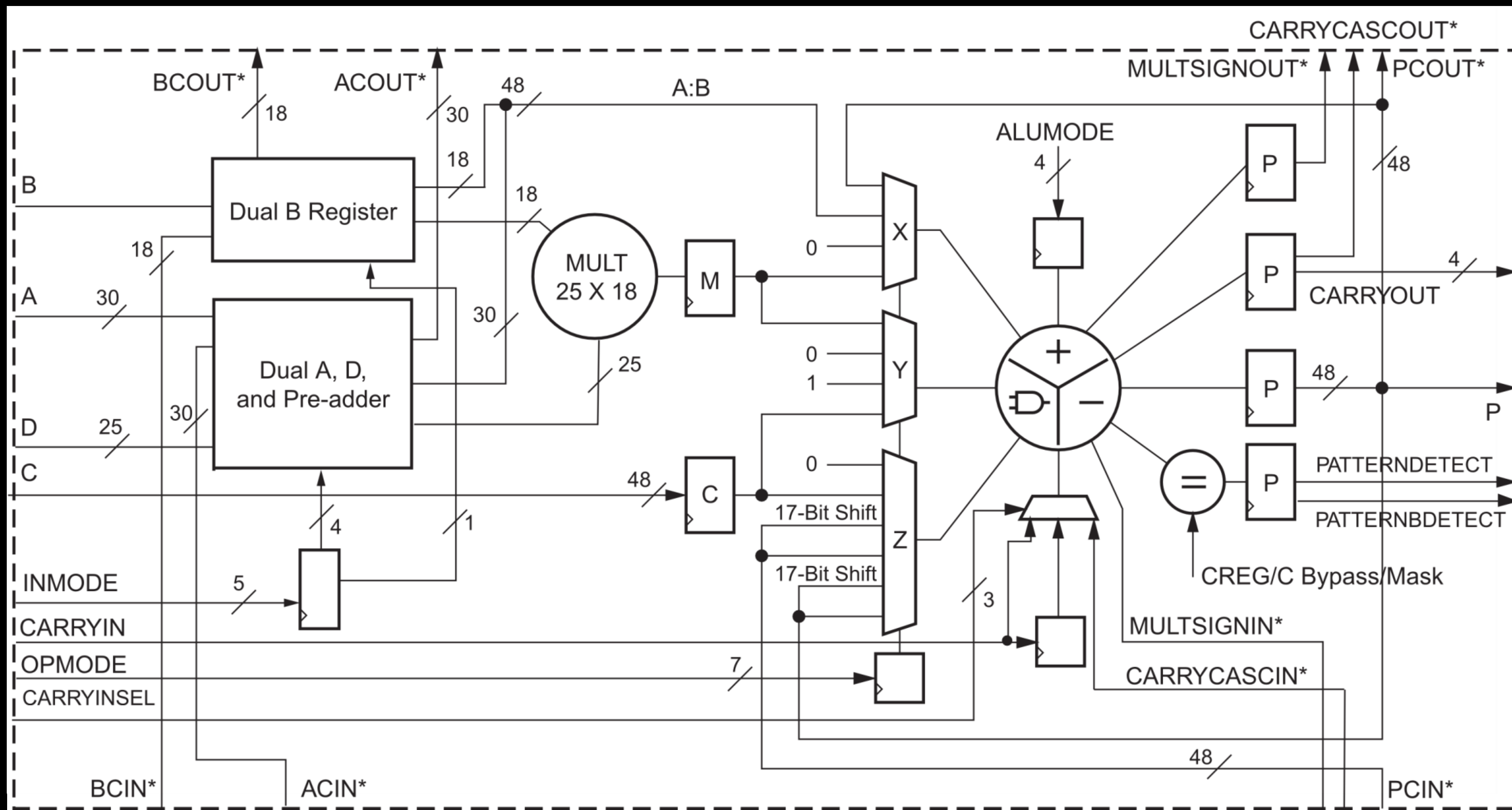
COMBINATORIAL LOGIC BLOCK



COMBINATORIAL LOGIC BLOCK

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

INTEGRATED DIGITAL SIGNAL PROCESSING



BIGGEST XILINX “ULTRASCALE+” DEVICES

- Upwards of 8.9 million logic cells
 - All clocked at up to 500MHz
 - Over $O(10^{15})$ operations/second
 - 1 PetaBOP
- Upwards of 12,000 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 8.9 million logic cells
 - All clocked at up to 500MHz
 - Over $O(10^{15})$ operations/second
 - 1 PetaBOP
- Upwards of 12,000 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	20	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 8.9 million logic cells
 - All clocked at up to 500MHz
 - Over $O(10^{15})$ operations/second
 - 1 PetaBOP
- Upwards of 12,000 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	20	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

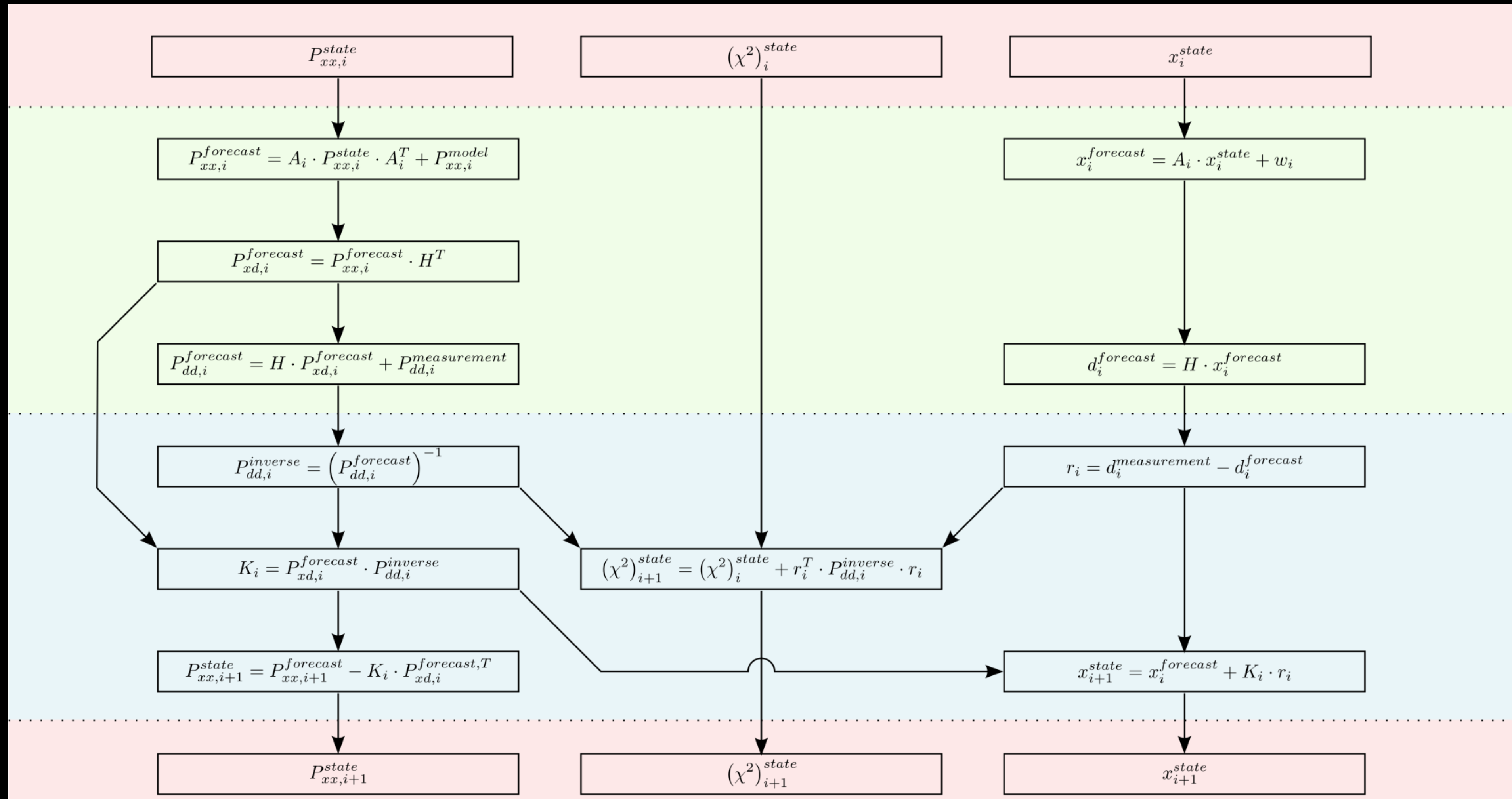
FPGAS: WHAT'S THE CATCH?

- 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**

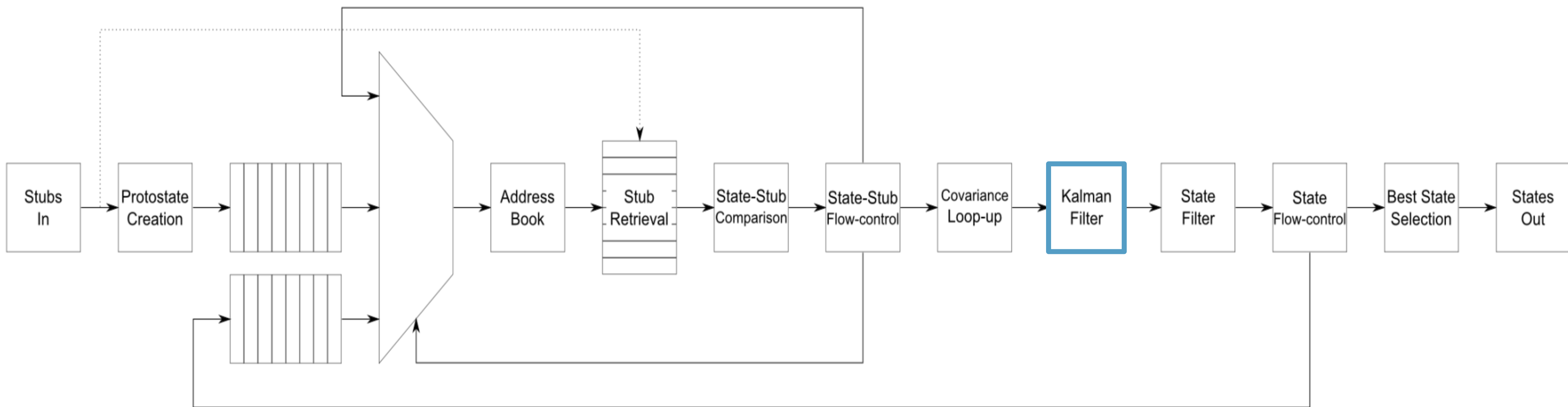
PROTOTYPE CMS TRACKING TRIGGER: KALMAN FILTER



PROTOTYPE CMS TRACKING TRIGGER: KALMAN FILTER

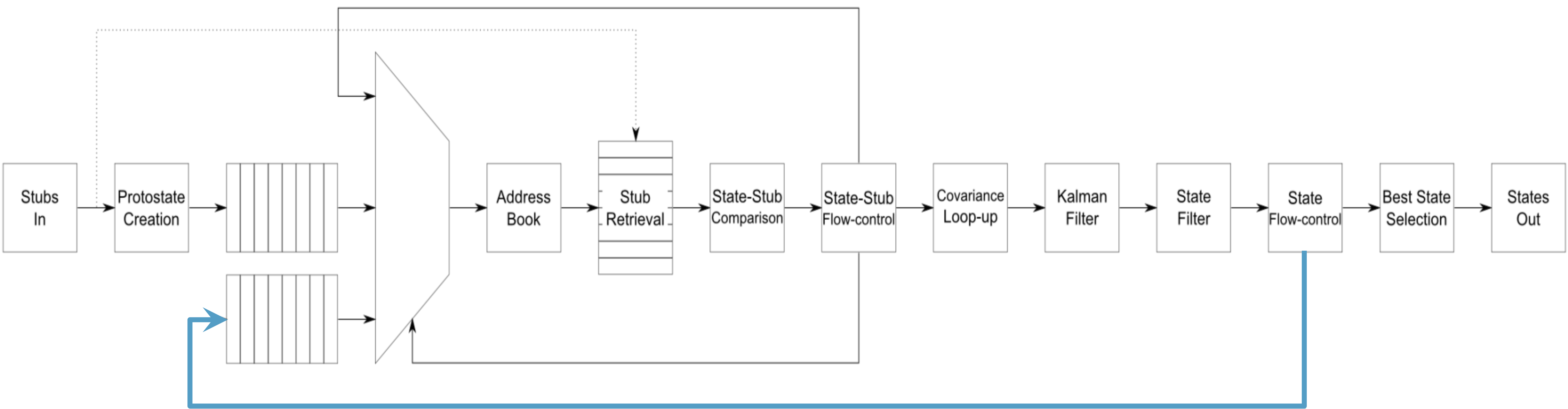


PROTOTYPE CMS TRACKING TRIGGER: KALMAN FILTER



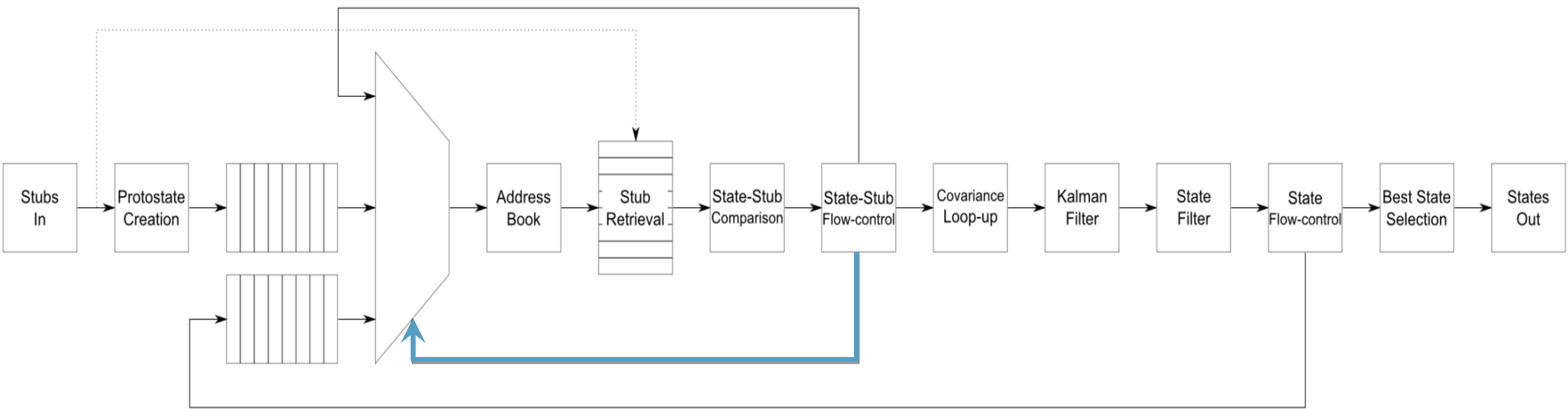
The maths is a relatively simple part of a more complex whole

PROTOTYPE CMS TRACKING TRIGGER: KALMAN FILTER



Kalman Filter is iterative

PROTOTYPE CMS TRACKING TRIGGER: KALMAN FILTER



Kalman Filter 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
- Novelty – 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;

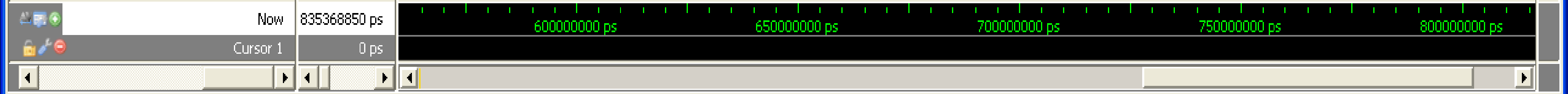
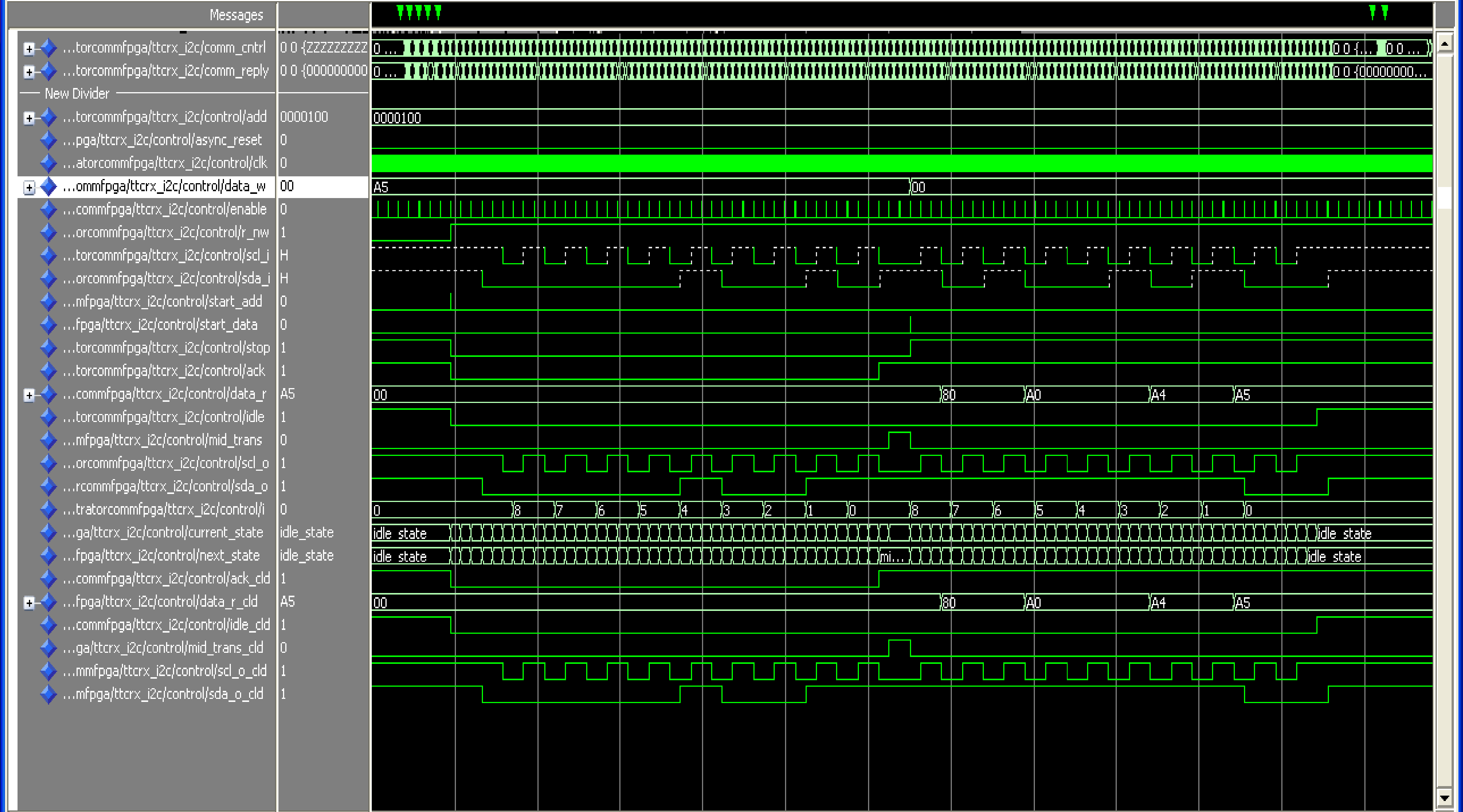
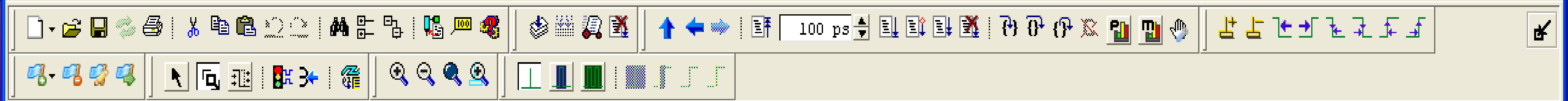
  G <= x or y;
end behavioural;
```

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



“Event display”

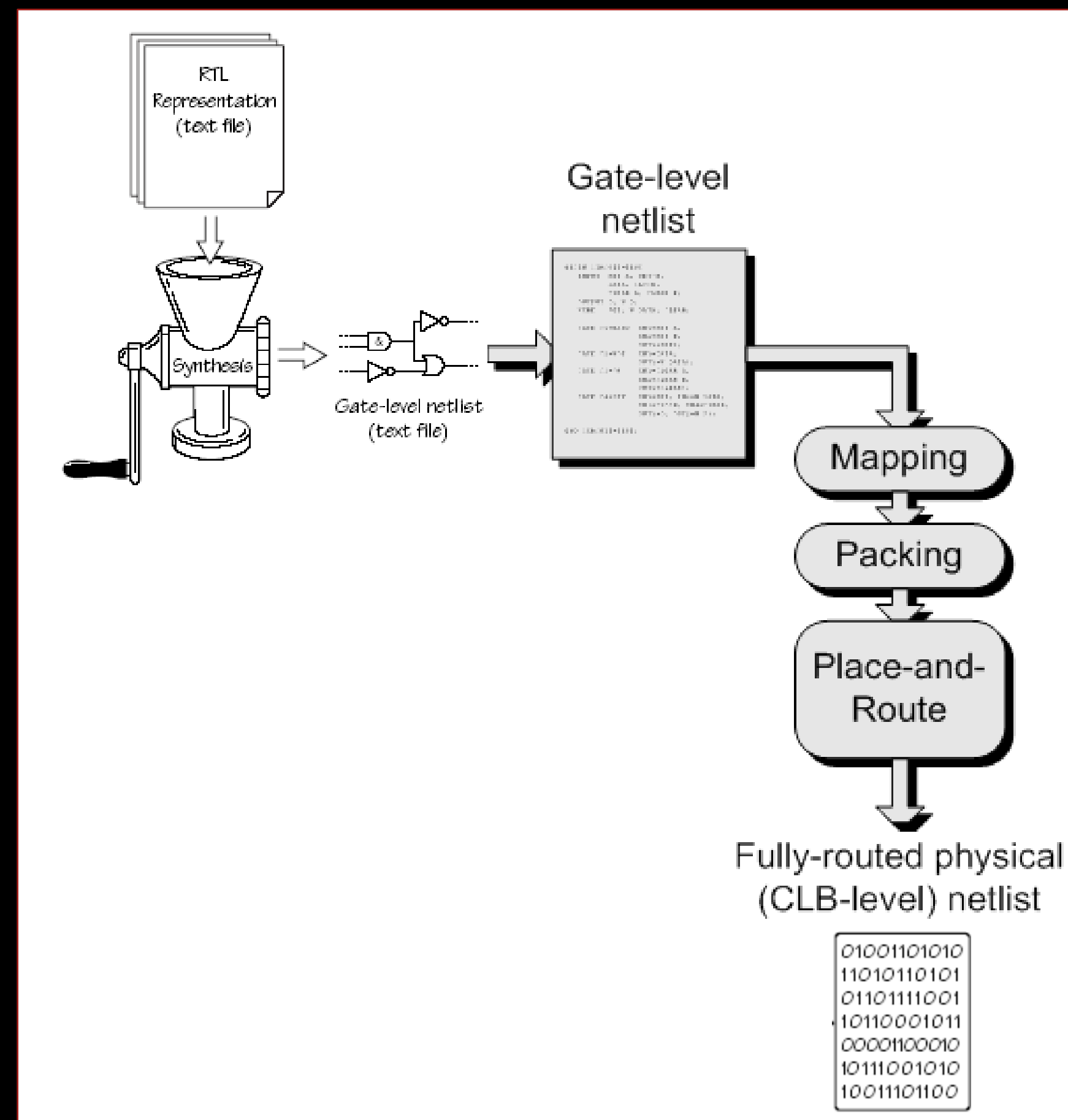
TESTBENCH SUITE

The screenshot shows the ModelSim SE-64 10.1b interface. The top window, titled 'Wave - Default', displays a timing diagram for a testbench suite. The signals are organized into groups: JET SUMS, JET VETO, PILEUP EST., JET VETO FILTERED, and PILEUP SUB. JETS. Each signal's value is shown as a series of bits over time, with a cursor at 1999.462 ns. The bottom window, titled 'Transcript', shows the simulation output for various PHI values (131, 121, 111, 101, 91, 81, 71, 61, 51, 41, 31, 21, 11, 01) and their corresponding results. The transcript is formatted as follows:

```
# PHI = 131 | 58| 475| 427| 98| 55| 24| 01| 439| 390| 44| 48| 394| 85| 493| 370| 30| 448| 116| 417| 350| 248| 457| 69| 84| 92| 62| 195| 441| 498| 314| 268| 386| 356| 315| 422| 343| 235| 342| 86| 99|
# PHI = 121 | 261| 456| 384| 219| 54| 120| 462| 231| 471| 77| 426| 62| 189| 142| 198| 401| 157| 138| 315| 391| 56| 283| 142| 146| 269| 478| 435| 244| 428| 415| 59| 376| 81| 372| 331| 491| 123| 403| 501| 185|
# PHI = 111 | 242| 121| 310| 85| 53| 363| 263| 487| 85| 141| 56| 216| 258| 132| 122| 112| 147| 140| 125| 461| 60| 52| 234| 467| 17| 469| 91| 182| 318| 62| 463| 329| 197| 226| 338| 78| 504| 388| 196| 493|
# PHI = 101 | 117| 491| 400| 220| 408| 82| 496| 263| 210| 35| 255| 323| 338| 318| 155| 489| 273| 226| 324| 465| 466| 144| 322| 297| 495| 176| 182| 232| 124| 471| 105| 194| 410| 71| 482| 438| 17| 215| 487| 251|
# PHI = 91 | 71| 61| 491| 91| 201| 466| 30| 190| 395| 475| 94| 444| 198| 355| 278| 71| 449| 419| 136| 394| 78| 115| 39| 17| 320| 478| 64| 474| 129| 327| 155| 30| 326| 123| 428| 108| 474| 274| 62| 277|
# PHI = 81 | 168| 137| 495| 237| 178| 82| 375| 89| 267| 494| 317| 239| 400| 91| 129| 238| 411| 81| 384| 111| 99| 235| 266| 432| 300| 211| 179| 465| 267| 301| 419| 432| 475| 144| 100| 169| 164| 260| 153| 447|
# PHI = 71 | 91| 276| 374| 131| 320| 463| 65| 72| 74| 389| 17| 472| 61| 110| 369| 315| 346| 420| 482| 207| 309| 481| 81| 294| 130| 498| 151| 237| 435| 140| 151| 45| 211| 427| 114| 159| 117| 151| 444| 354|
# PHI = 61 | 137| 493| 288| 252| 310| 481| 173| 368| 485| 462| 169| 354| 81| 53| 226| 457| 464| 387| 360| 401| 95| 356| 260| 410| 163| 403| 54| 449| 280| 415| 244| 351| 449| 31| 69| 183| 112| 91| 142| 106|
# PHI = 51 | 257| 438| 453| 498| 395| 71| 379| 238| 188| 314| 384| 226| 292| 41| 499| 146| 245| 14| 413| 252| 346| 146| 195| 316| 412| 71| 161| 82| 251| 216| 226| 380| 392| 145| 305| 309| 313| 506| 268| 423|
# PHI = 41 | 43| 27| 393| 11| 330| 171| 470| 401| 306| 431| 55| 502| 378| 135| 249| 128| 435| 297| 202| 374| 272| 192| 211| 94| 318| 349| 507| 456| 373| 311| 192| 199| 240| 244| 31| 76| 32| 199| 371| 476|
# PHI = 31 | 301| 461| 422| 325| 300| 195| 461| 382| 293| 465| 184| 430| 325| 185| 198| 152| 278| 327| 141| 434| 61| 86| 466| 255| 74| 339| 82| 397| 241| 475| 198| 397| 351| 476| 51| 491| 469| 194| 492| 447|
# PHI = 21 | 259| 150| 278| 469| 282| 393| 411| 473| 284| 228| 150| 320| 185| 194| 260| 408| 330| 217| 52| 370| 207| 97| 221| 461| 97| 442| 227| 186| 242| 416| 448| 179| 430| 197| 401| 01| 485| 371| 378| 267|
# PHI = 11 | 412| 333| 366| 494| 155| 384| 198| 324| 321| 194| 133| 116| 114| 257| 135| 136| 473| 119| 90| 476| 301| 401| 381| 63| 300| 433| 274| 244| 287| 501| 143| 137| 211| 483| 501| 395| 275| 447| 327| 345|
# PHI = 01 | 470| 601| 389| 282| 73| 251| 305| 193| 450| 414| 186| 178| 211| 141| 307| 418| 471| 258| 91| 419| 78| 468| 328| 457| 441| 499| 179| 360| 496| 311| 1061| 871| 435| 469| 383| 465| 231| 96| 145| 308|
```

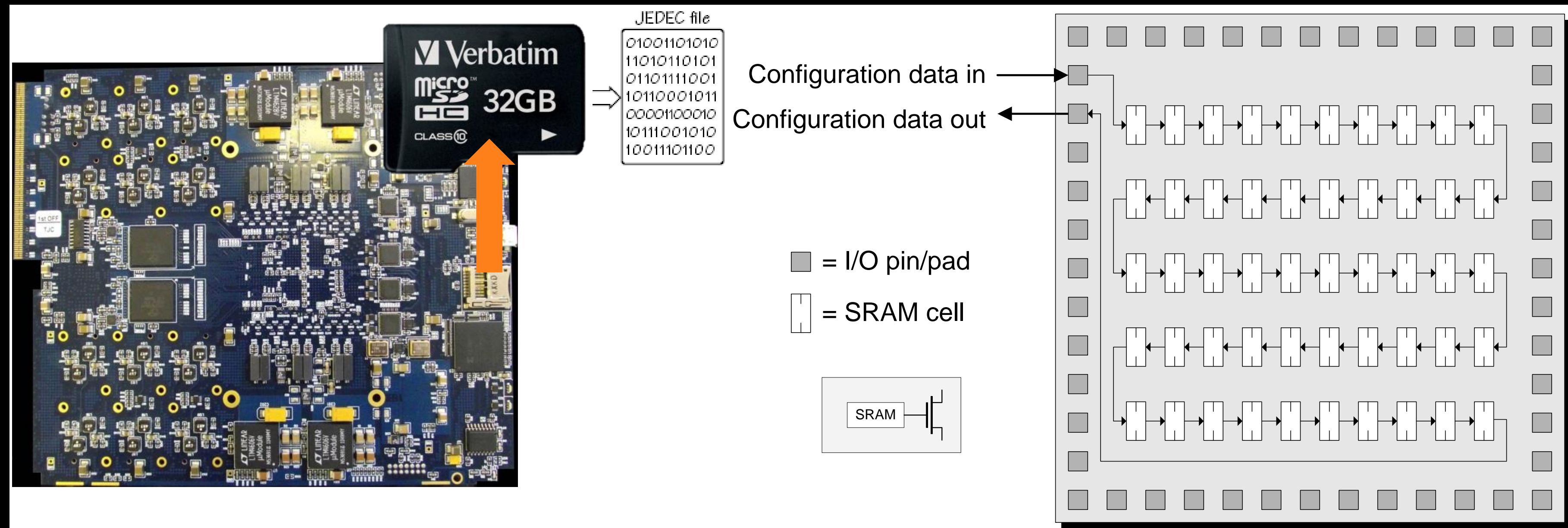

DESIGNING LOGIC WITH FPGAS

- 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

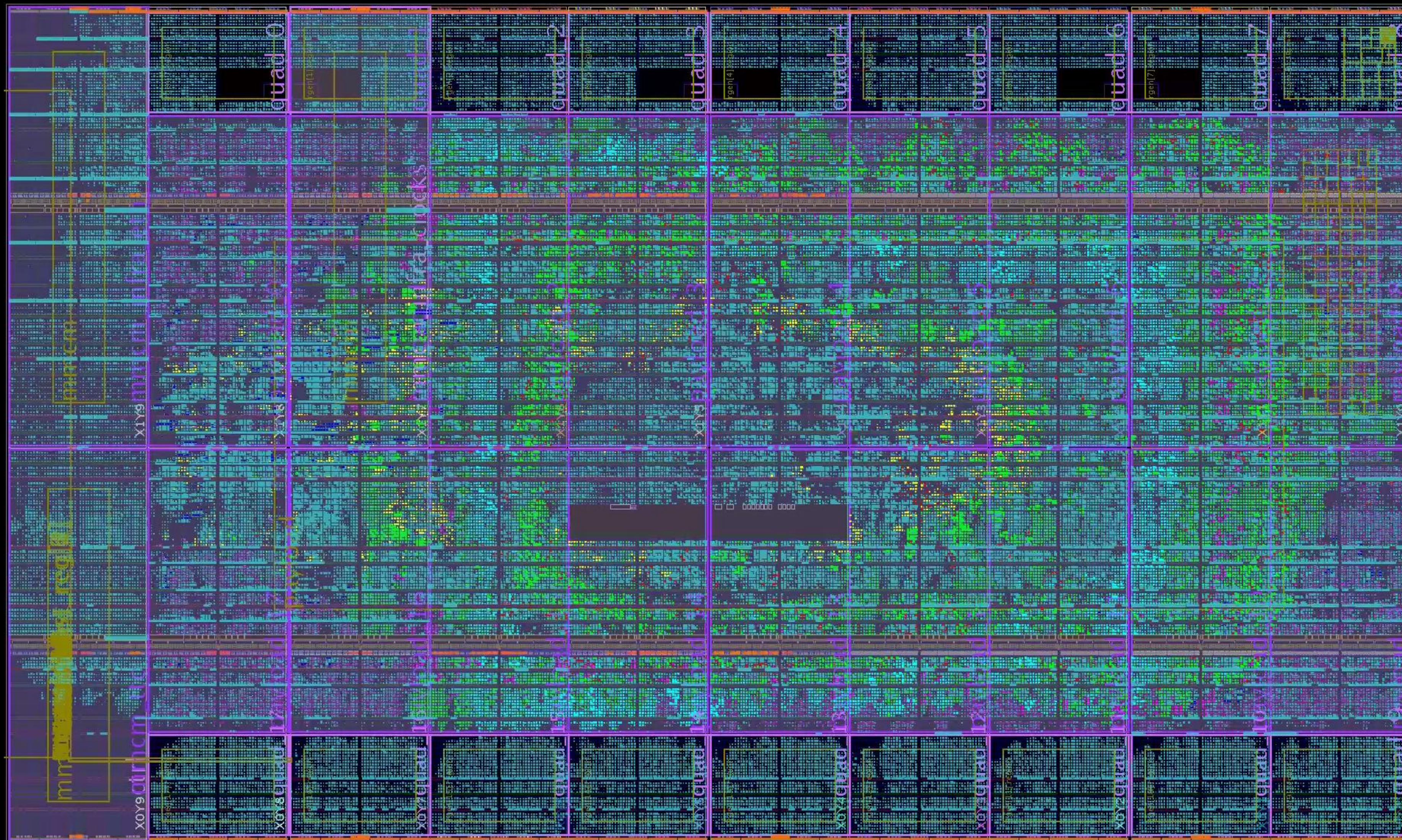
- Millions of SRAM cells holding LUTs and Interconnect Routing
- Volatile Memory: Lose configuration when board power is turned off.
- Keep bit patterns describing the SRAM cells in non-Volatile Memory e.g. PROM or memory card
- Configuration takes ~ secs



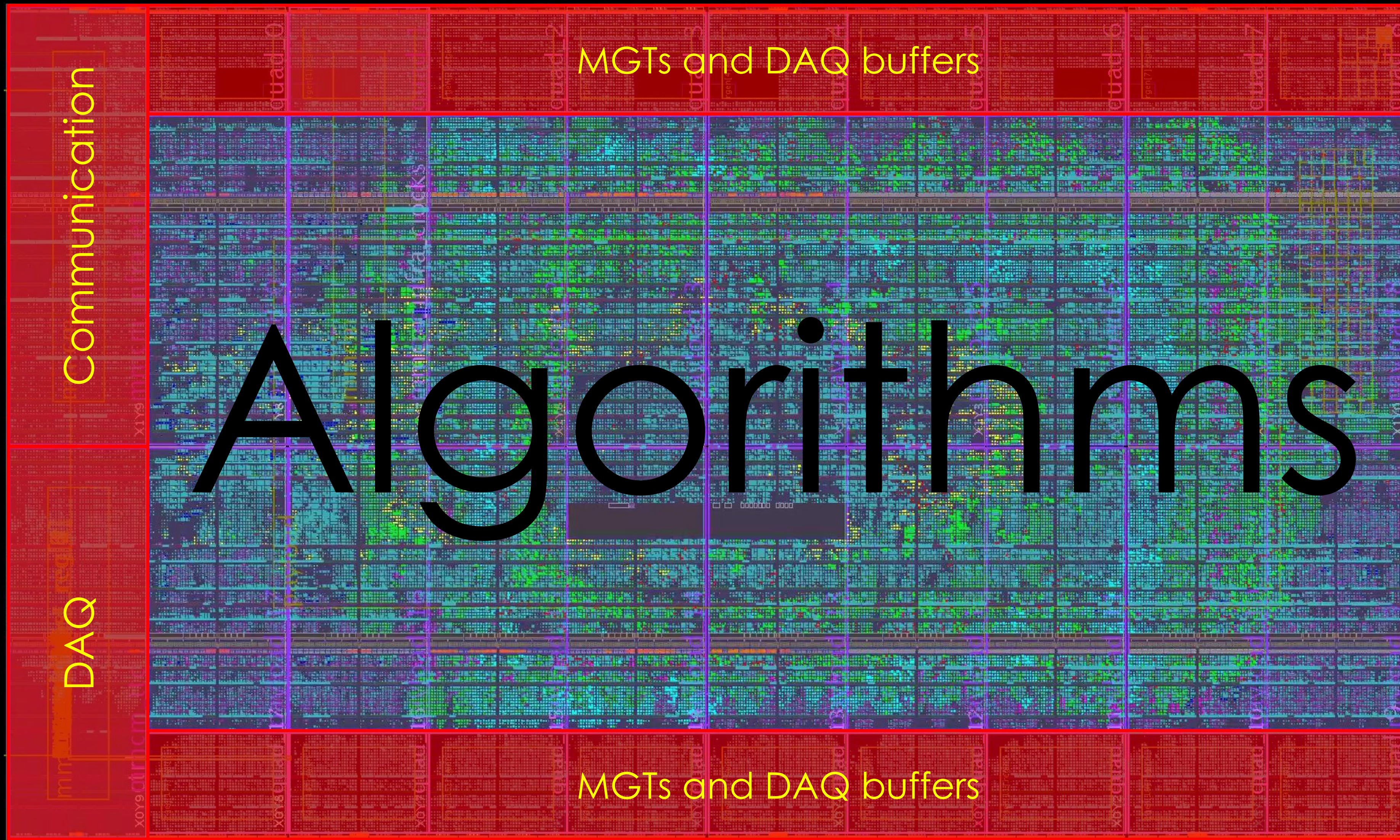
IT DOESN'T WORK: HOW TO DEBUG

- 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



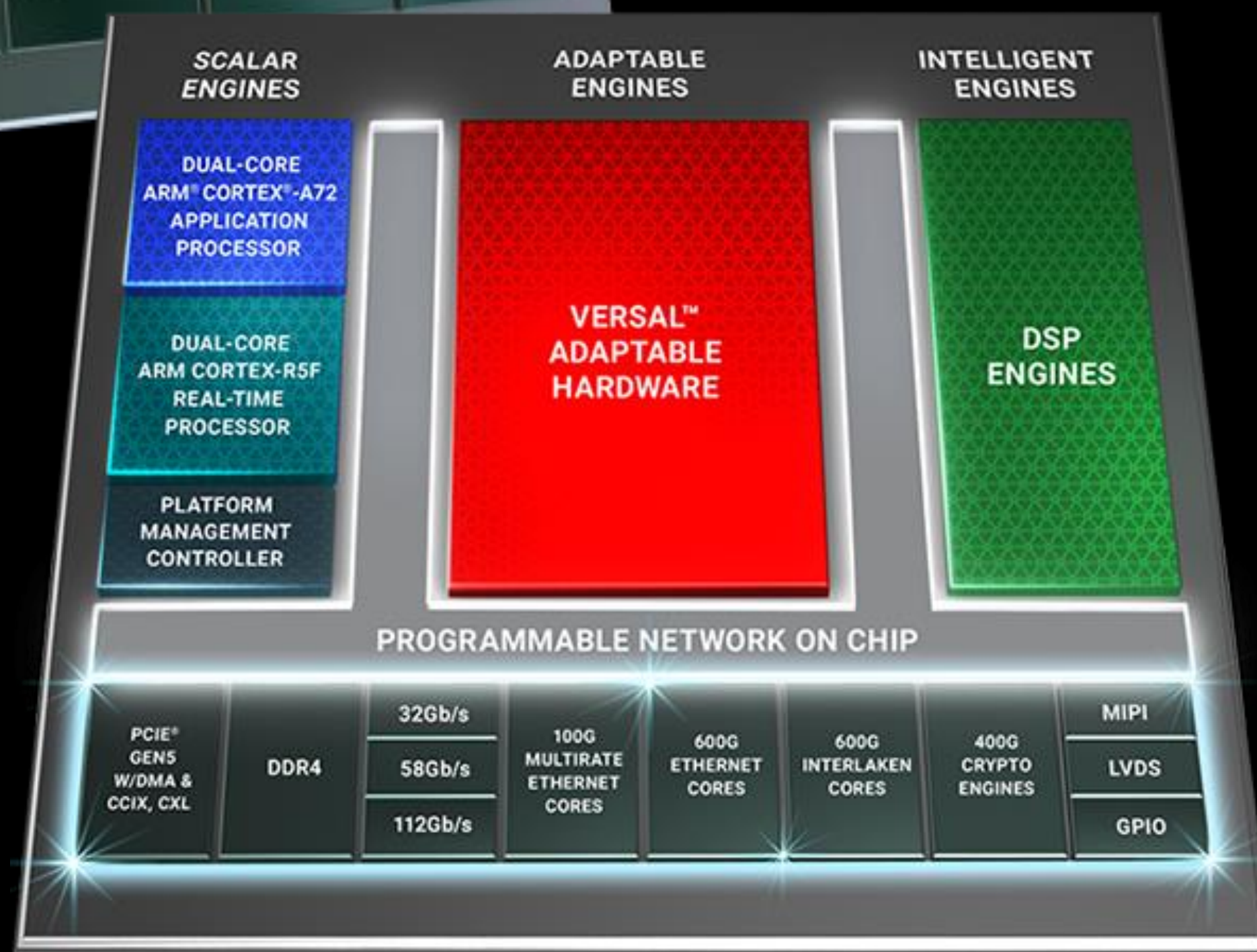
FLOORPLAN OF FIRMWARE IN MP7



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

- FPGAs are expensive (high-end £10k-100k cf. £100)
- FPGAs are power-hungry (Although also very efficient if used correctly!)
- 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**

THE FUTURE OF THE FPGA?



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

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 wrong and you unleashed a world of pain on yourself**
- **Get them right and you can do magic and have a lot of fun in the process**

THANK YOU

Any questions?

