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





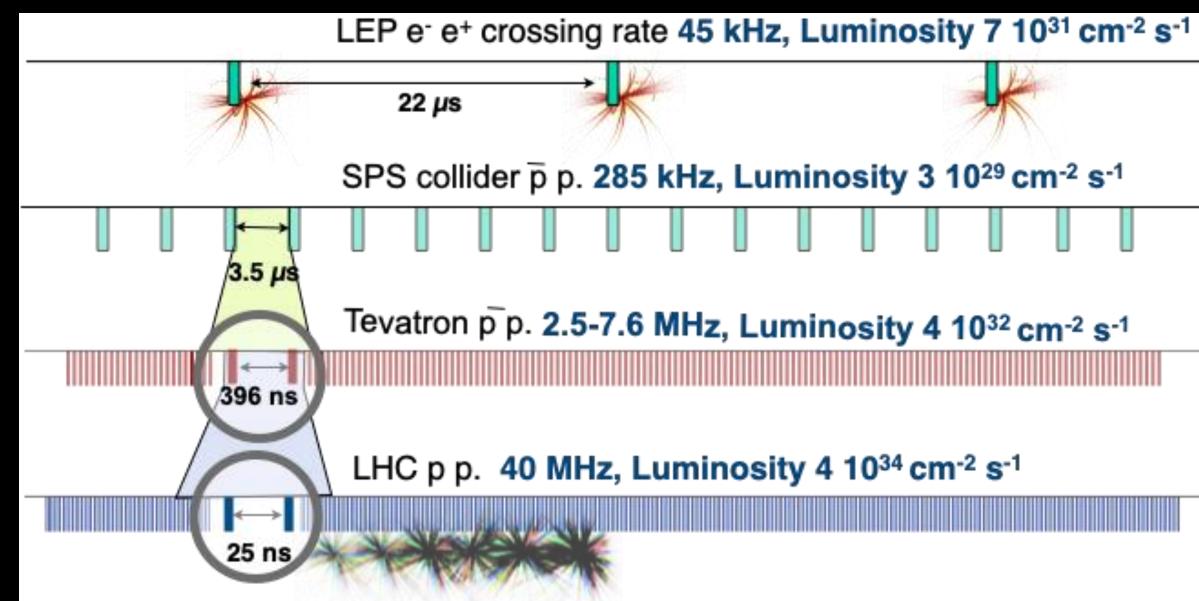
So... I should probably justify those statements...

RECALL FROM TRIGGER & DAQ LECTURES



- 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 ulletchannels
- What technology can we use?

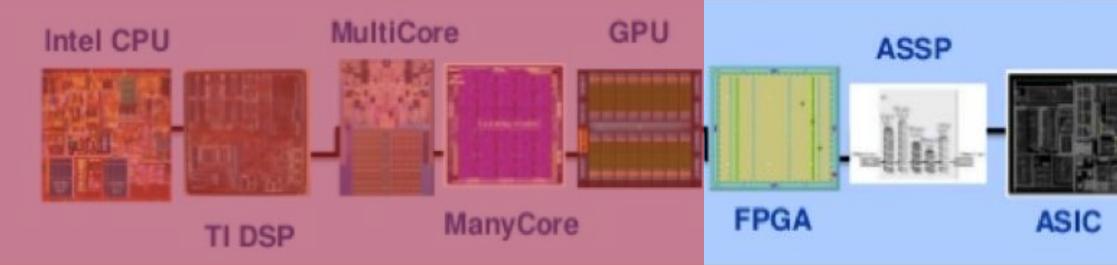
A NOTE ON TIMESCALES





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

PROGRAMMABLE DEVICES



Flexibility, Programming Abstraction

Performance, Area and Power Efficiency

CPU:

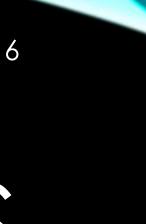
- Market-agnostic
- Accessible to many
- programmers (C++)
- Flexible, portable

FPGA:

- Somewhat Restricted Market
- Harder to Program (Verilog)
- More efficient than SW
- More expensive than ASIC

ASIC

- Market-specific
- Fewer programmers
- Rigid, less programmable
- Hard to build (physical)



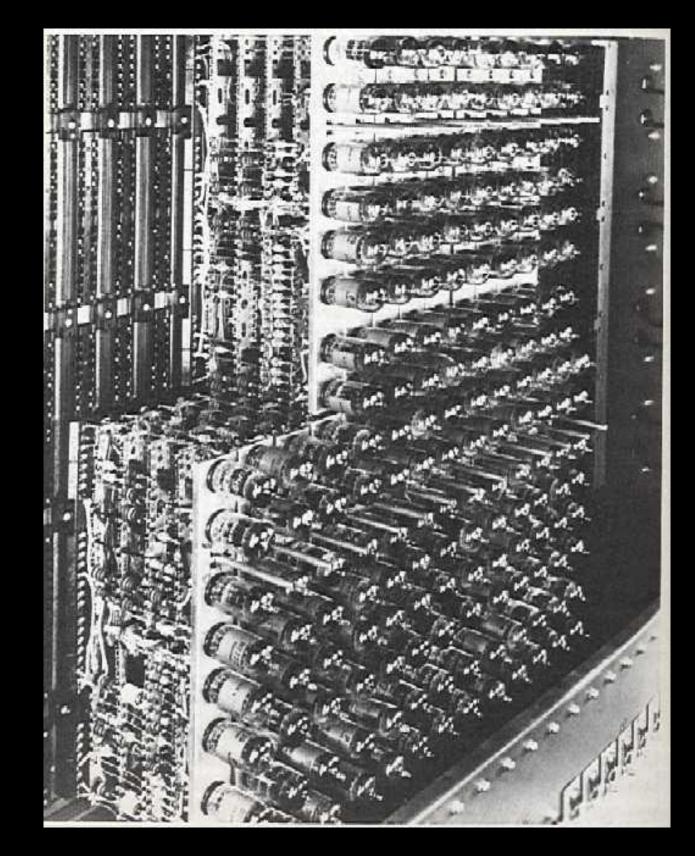


AN ASIDE: THE HISTORY OF ELECTRONICS

 Digital electronics really started with (colloquially, the "vacuum tube")

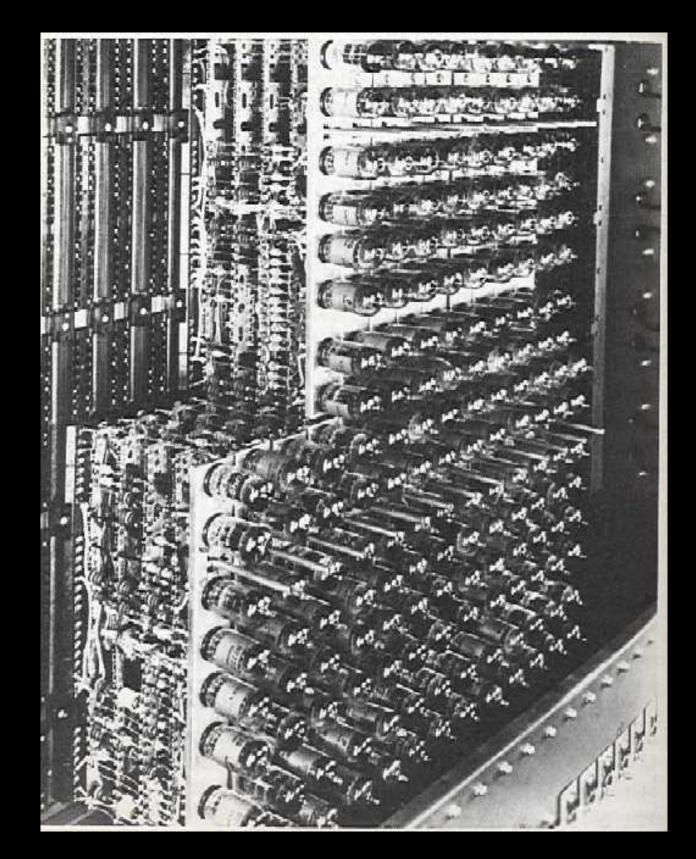


Digital electronics really started with the advent of the thermionic valve



Valve transistors

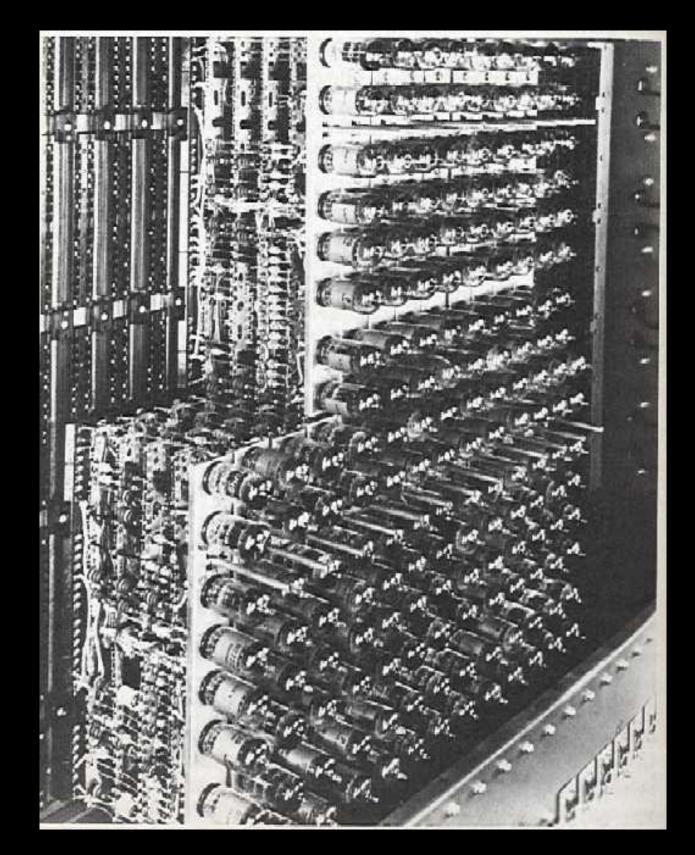




Valve transistors

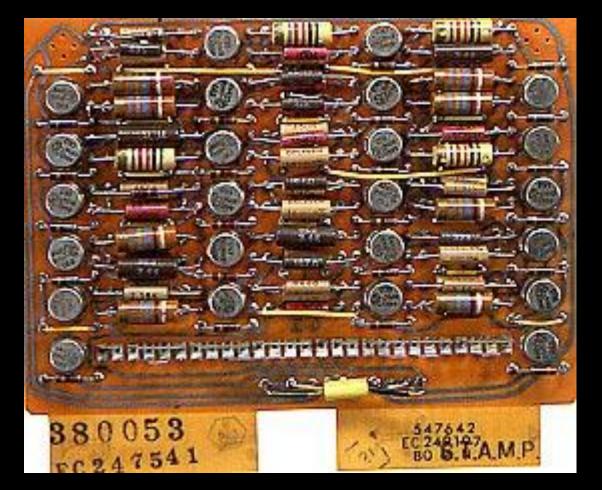


First solid-state transistors



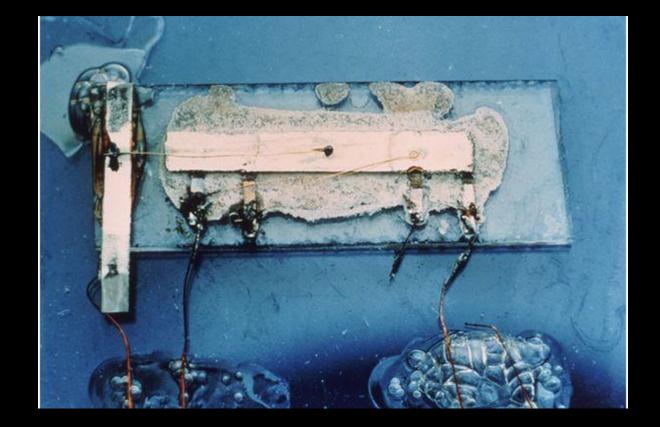
Valve transistors



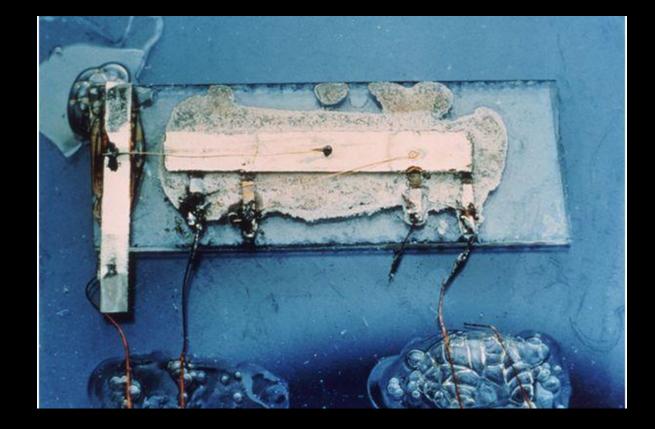


First solid-state transistors

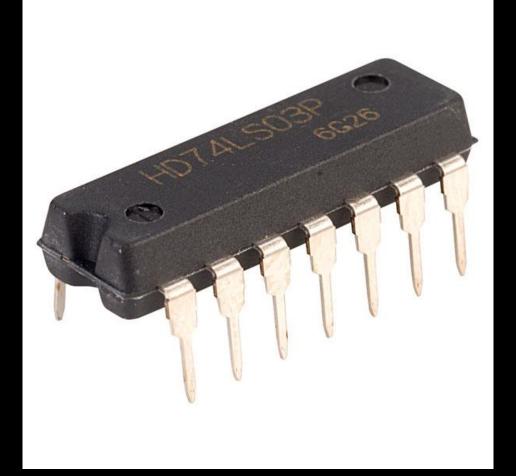
Solid-state transistors



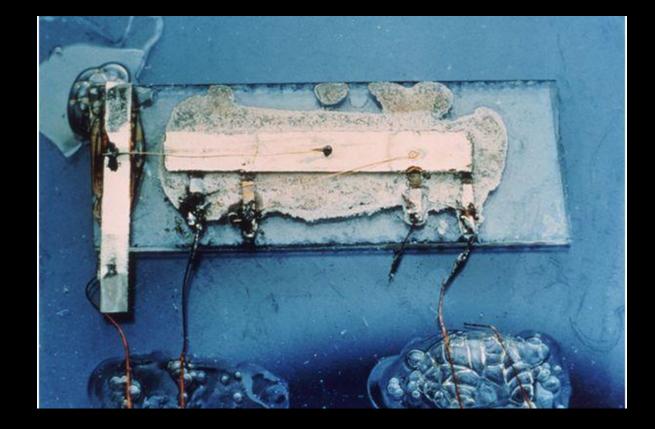
First multi-transistor silicon



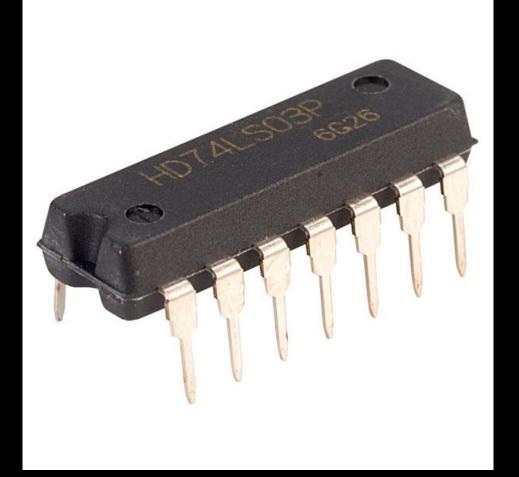
First multi-transistor silicon



Packaged Logic



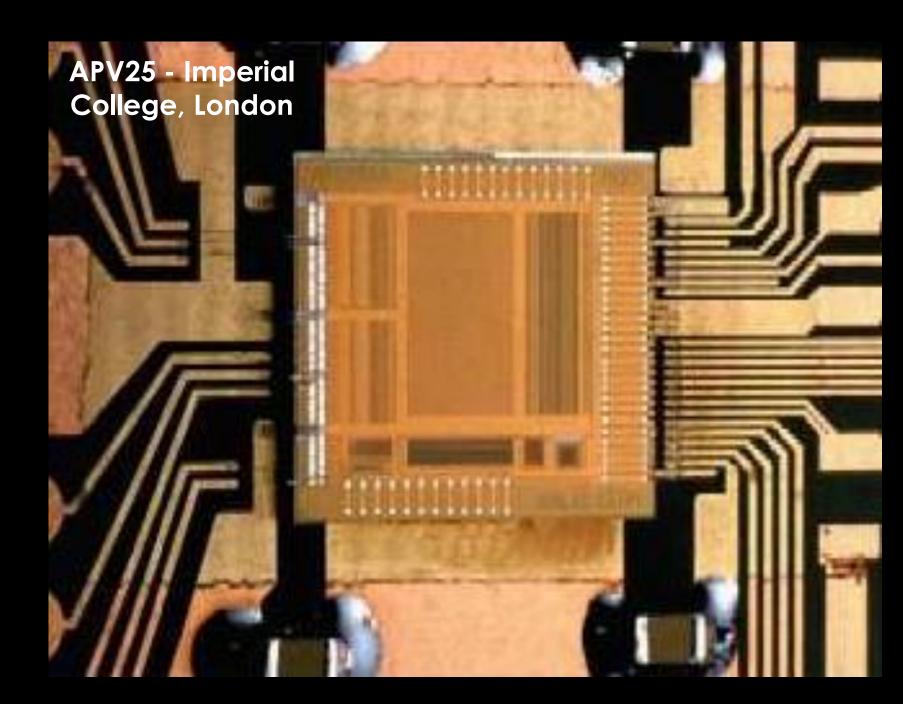
First multi-transistor silicon





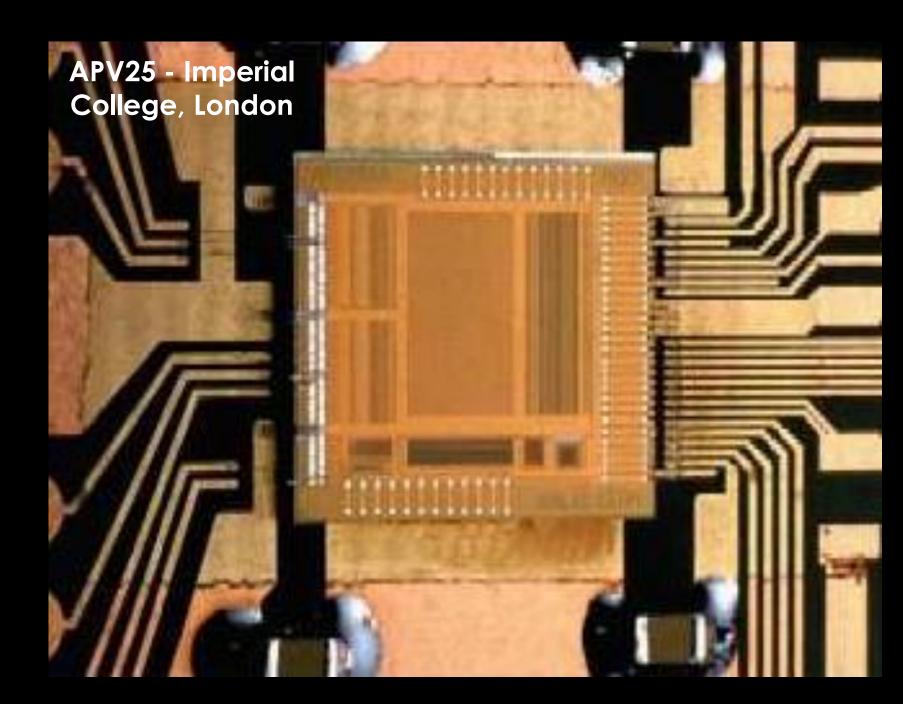
Packaged Logic

"Mini" processor board



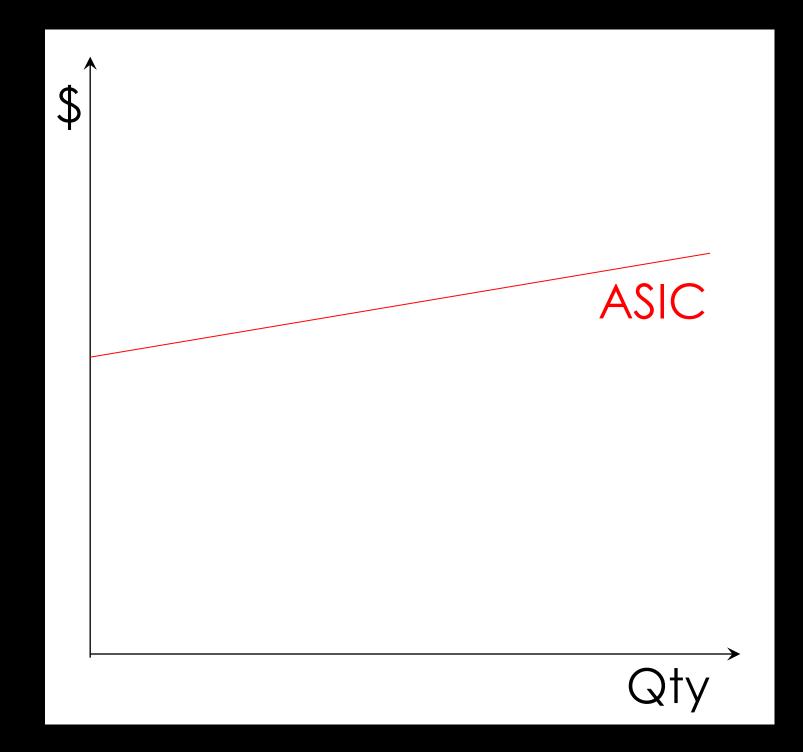
Application Specific Integrated Circuit (ASIC)

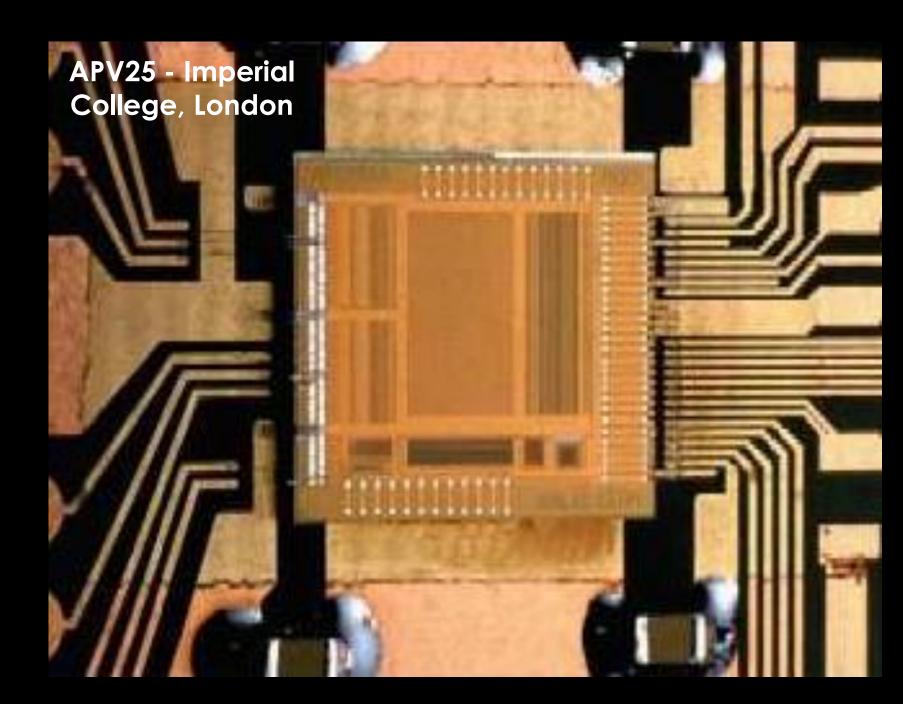




Application Specific Integrated Circuit (ASIC)

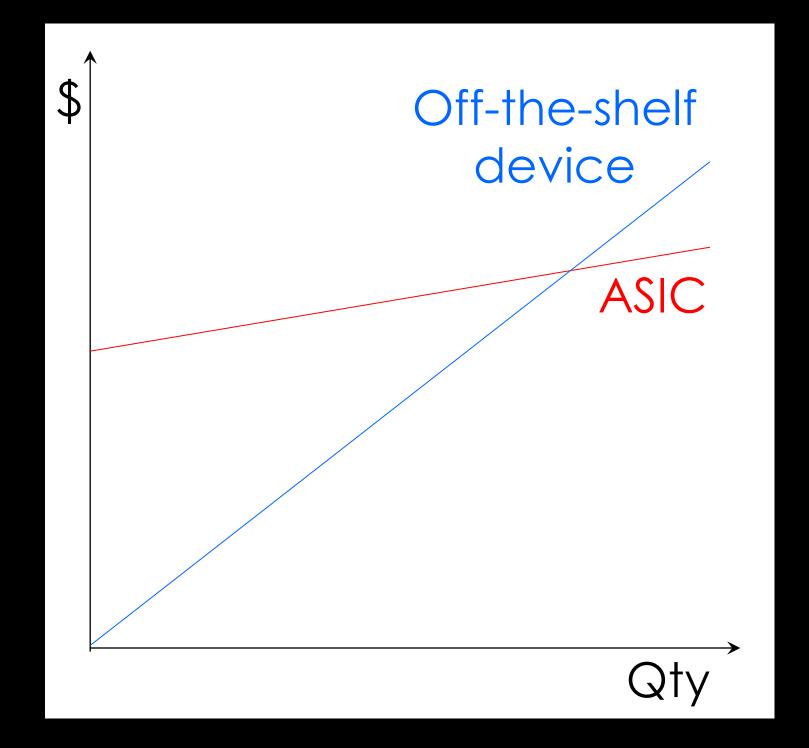






Application Specific Integrated Circuit (ASIC)

ASICS



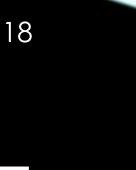
Have each operation performed by dedicated logic and let signal propagate as a wave through the logic

Combinatorial

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



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

Pipelined/Parallel

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

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

19



02am | 03am

Have each operation performed by Have each operation performed by dedicated logic the same logic and performing do that same operation a different operation on every clock cycle on every clock cycle

Pipelined/Parallel

Sequential





Have each operation performed by Have each operation performed by dedicated logic the same logic performing and a different operation do that same operation on every clock cycle on every clock cycle

Pipelined/Parallel

A debate as old as electronic computing itself

Sequential





Have each operation performed by Have each operation performed by dedicated logic the same logic performing and do that same operation a different operation on every clock cycle on every clock cycle

Pipelined/Parallel

"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.

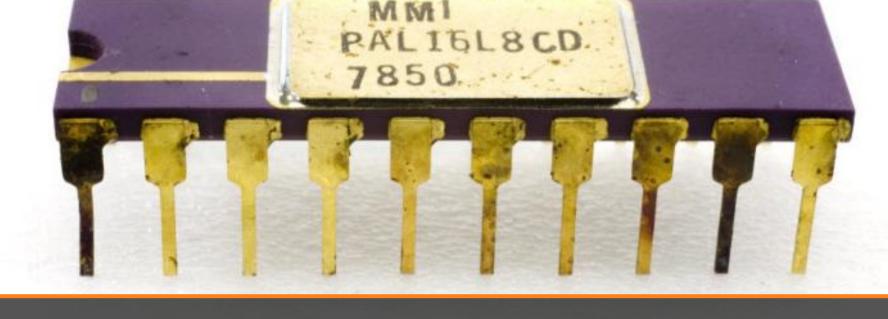
In an environment which has represented the absence of the need to think as the highest virtue, this is a decided disadvantage"

Daniel Slotnick, 1967

Sequential





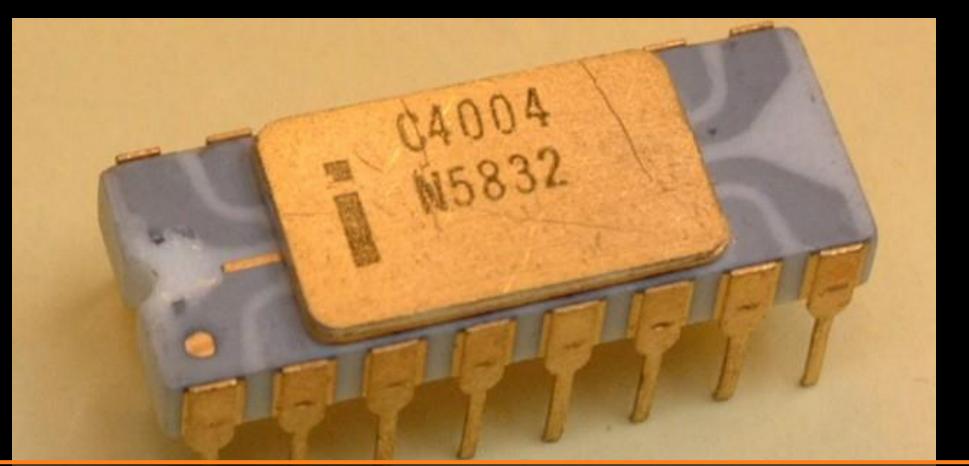


Pipelined/Parallel

Programmable Array Logic

Pack entire logic circuits in a chip

AND THE STORY DIVERGES...

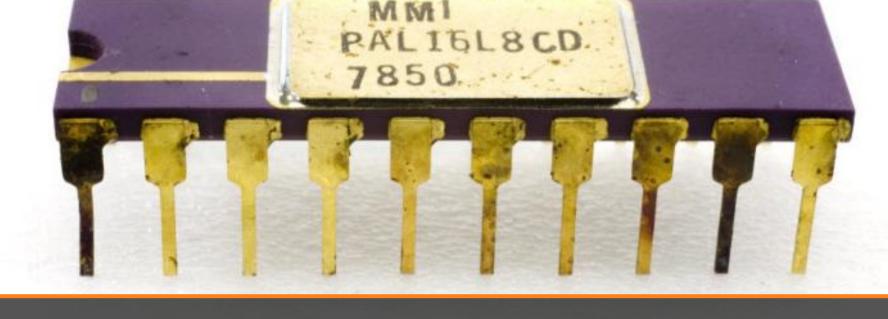


Sequential

Microprocessor

Perform all logical operations in one location, but sequentially



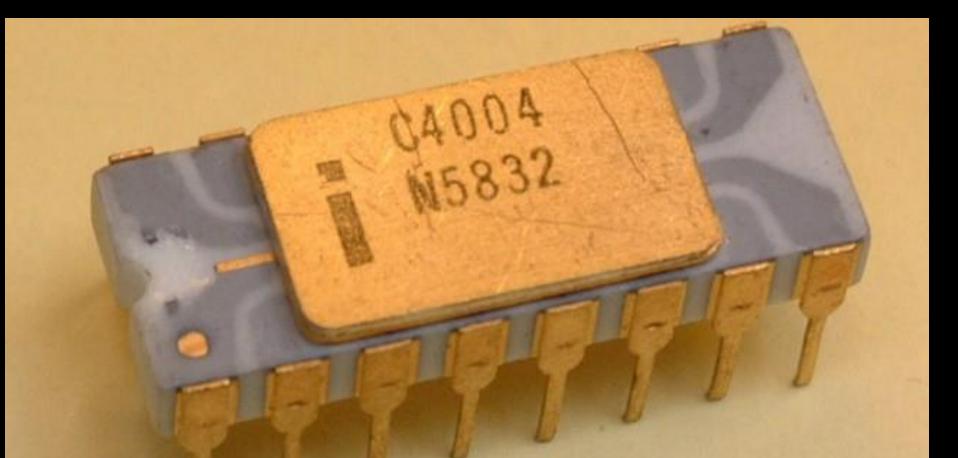


Pipelined/Parallel

Programmable Array Logic

Pack entire logic circuits in a chip

AND THE STORY DIVERGES...

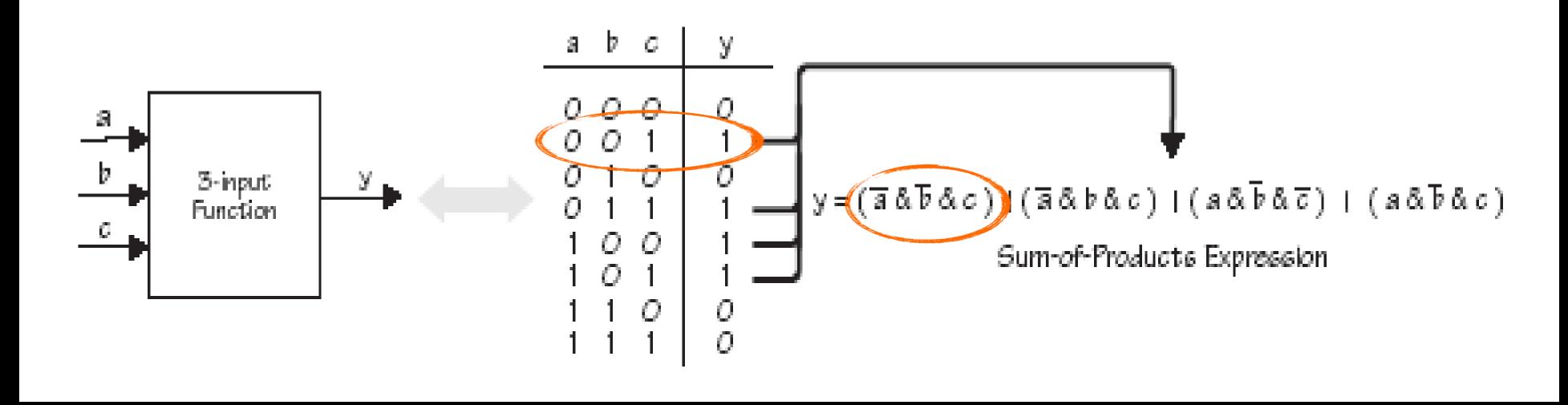


Sequential

Linviteorotessoner Performation logical operations in one location, but sequentially microprocessors



SUM-OF-PRODUCTS THEOREM

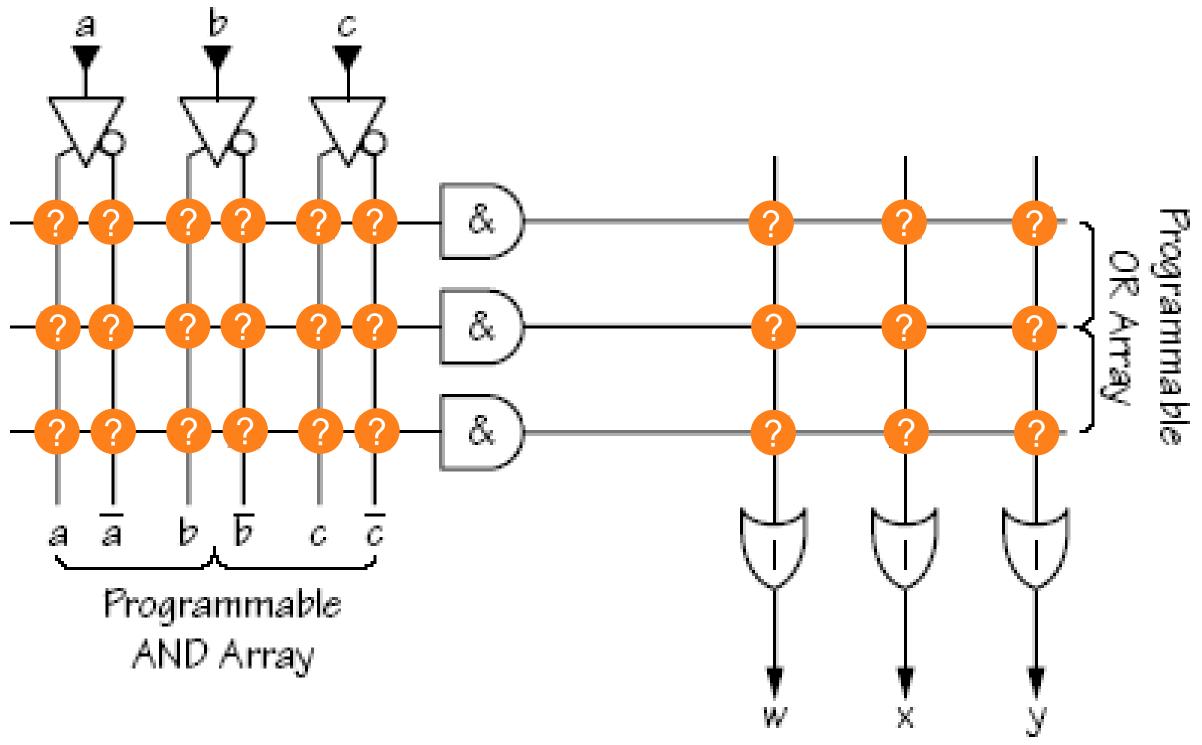


- Any Boolean operation may be expressed as
- Or

the OR of AND operations (Sum of products form)

the AND of OR operations (Product of sums form)

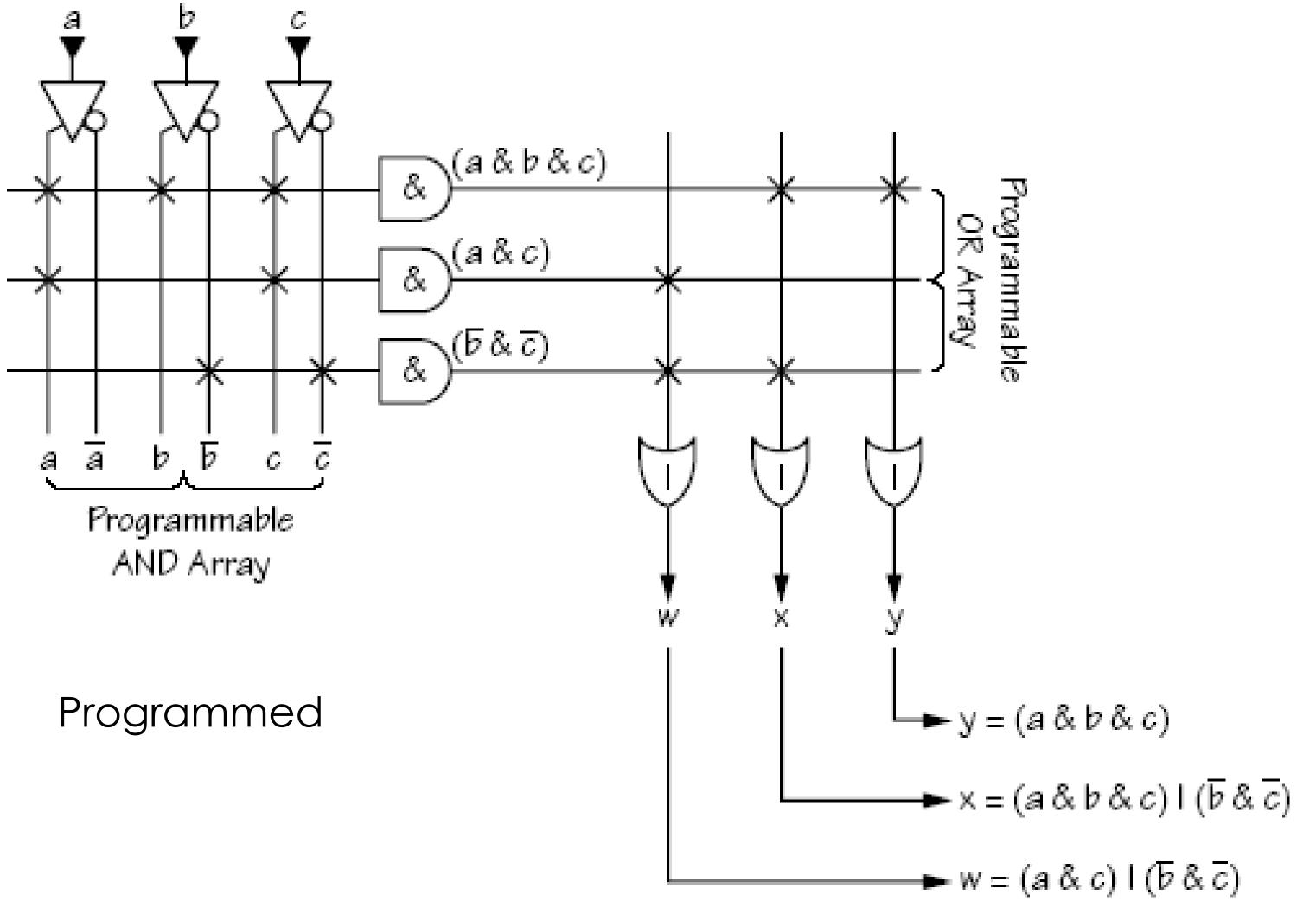
PROGRAMMABLE LOGIC DEVICES (PLDS) b



Unprogrammed



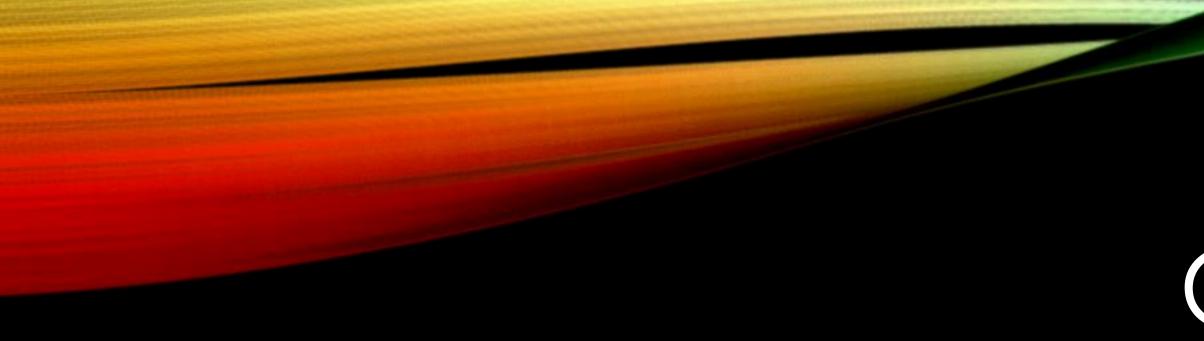
PROGRAMMABLE LOGIC DEVICES (PLDS) b

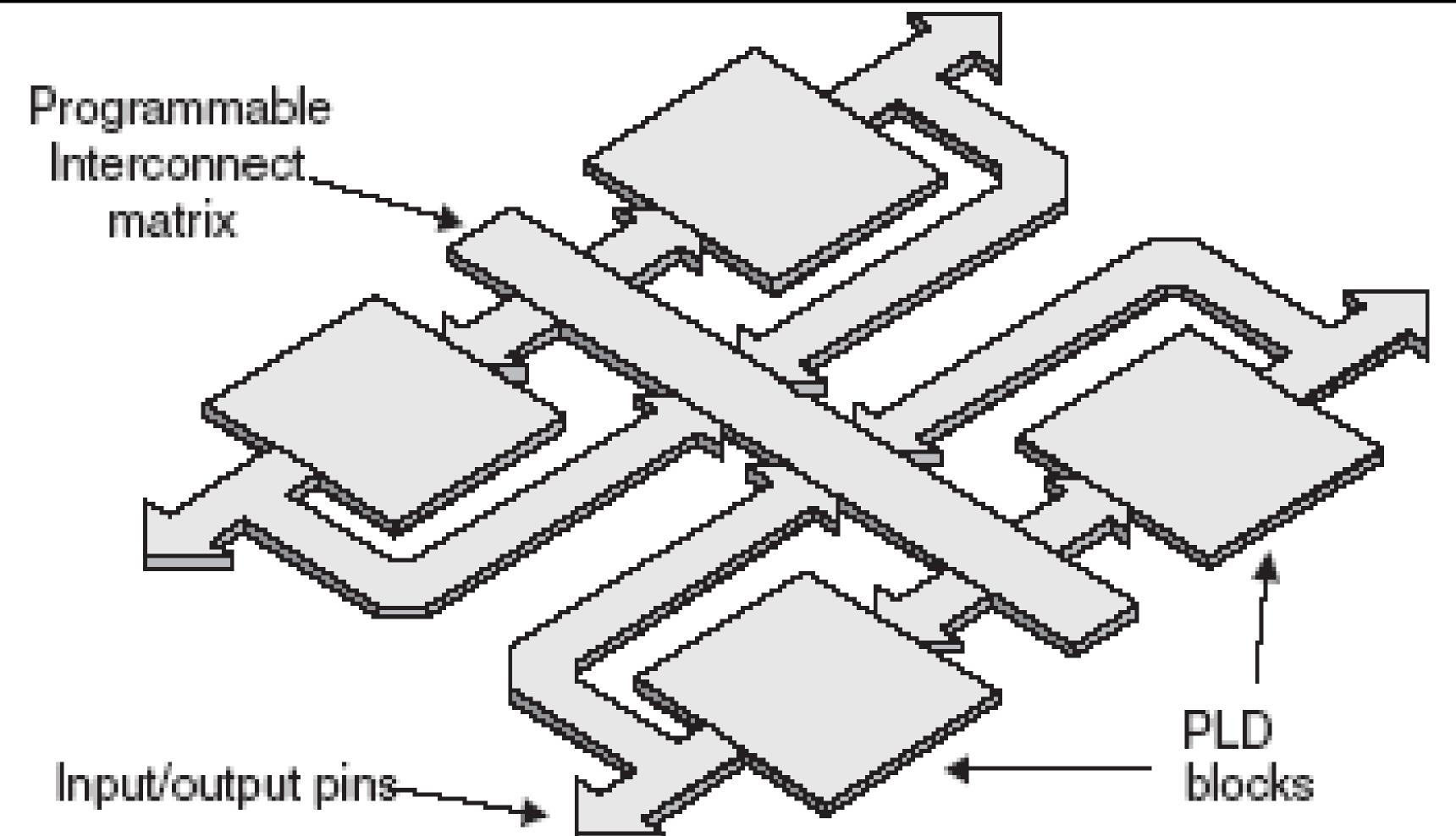


PROGRAMMABLE LOGIC DEVICES (PLDS)

- Originally one-time programmable
- Later field reprogrammable
- What did people do? Build boards with many PLDs...

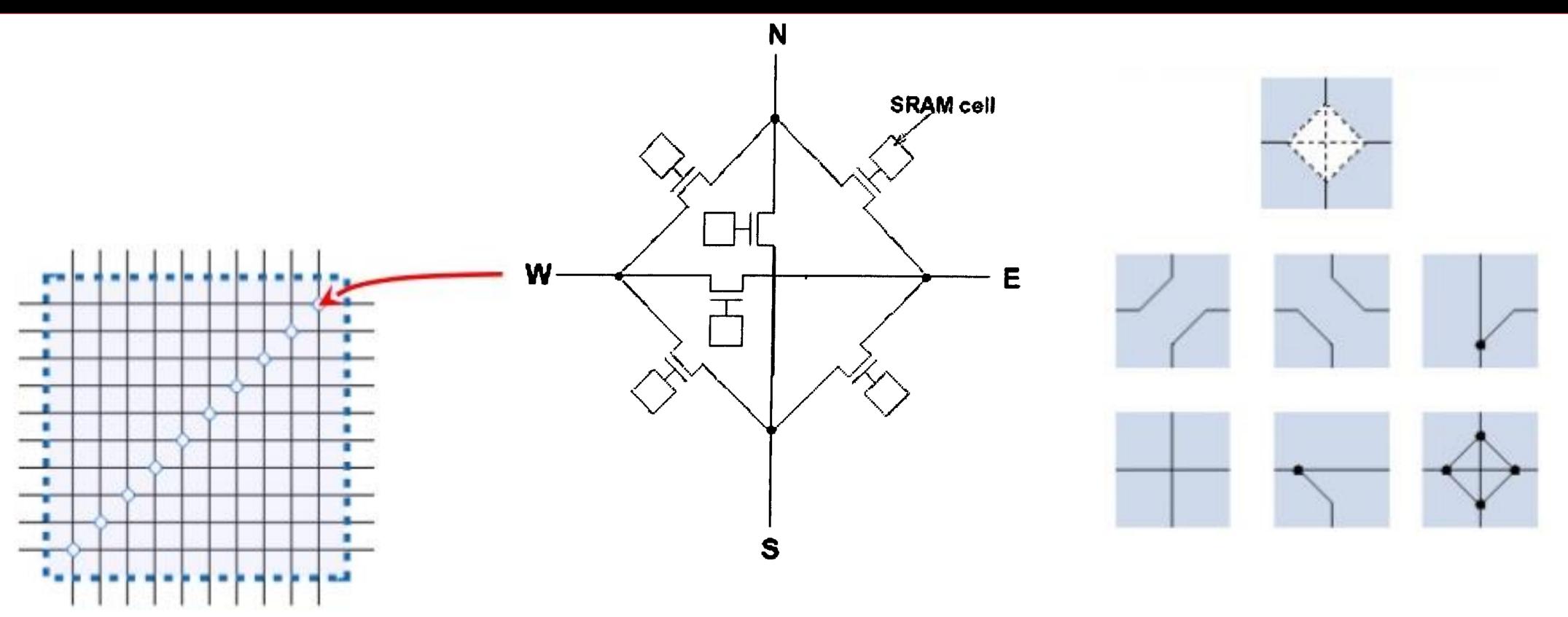


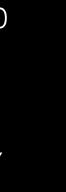




COMPLEX PLDS (CPLDS)

PROGRAMMABLE INTERCONNECT MATRIX





AN ALTERNATIVE APPROACH

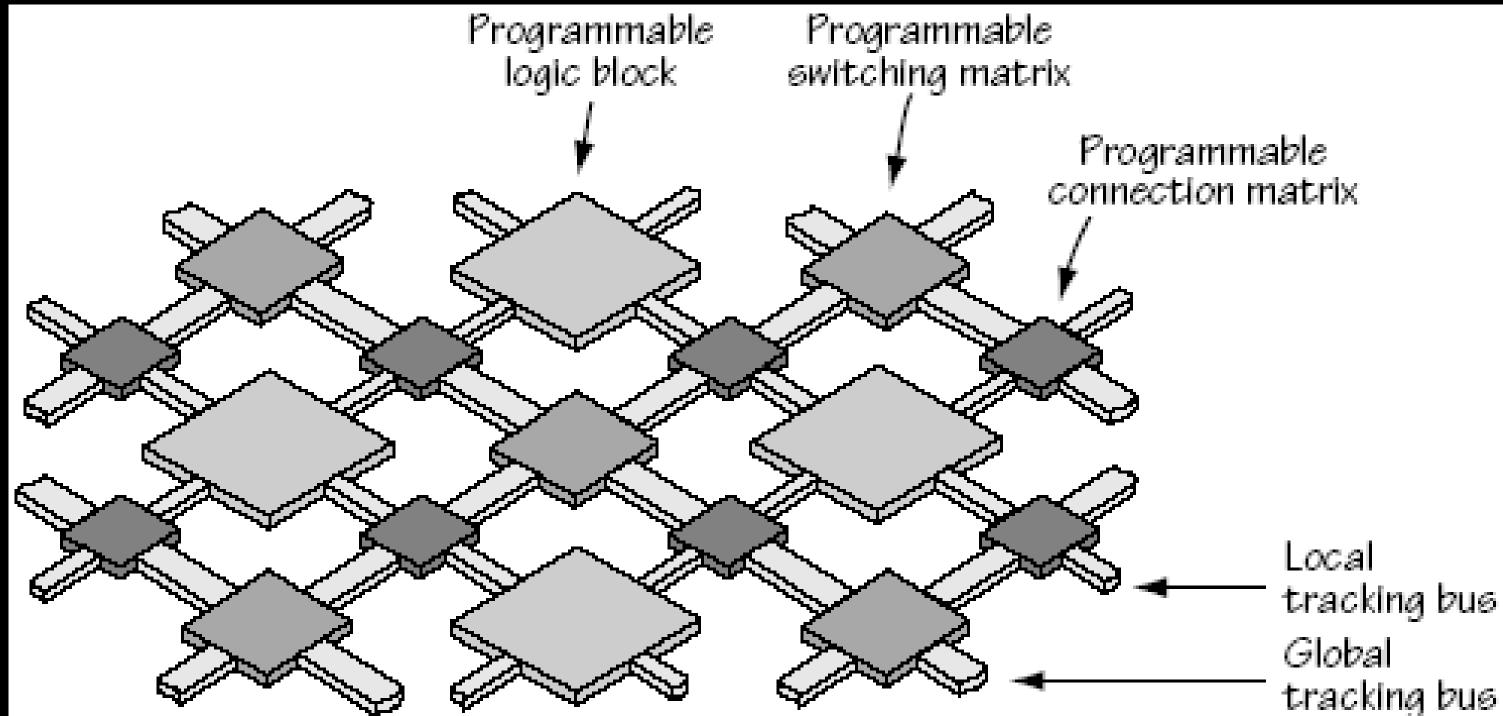
- Why bother with the complexity of the PLD cell?
- Replace the PLD cell with a simple SRAM:
 - Data-in becomes the "address"
 - Outputs the preloaded value for a given input

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)

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

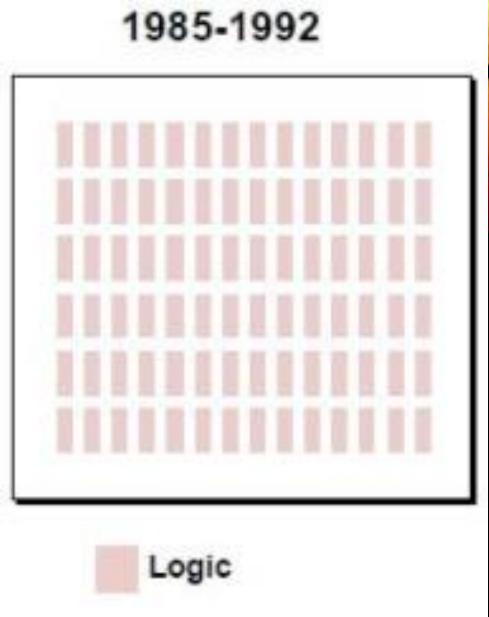


FIELD PROGRAMMABLE GATE ARRAYS (FPGAS)

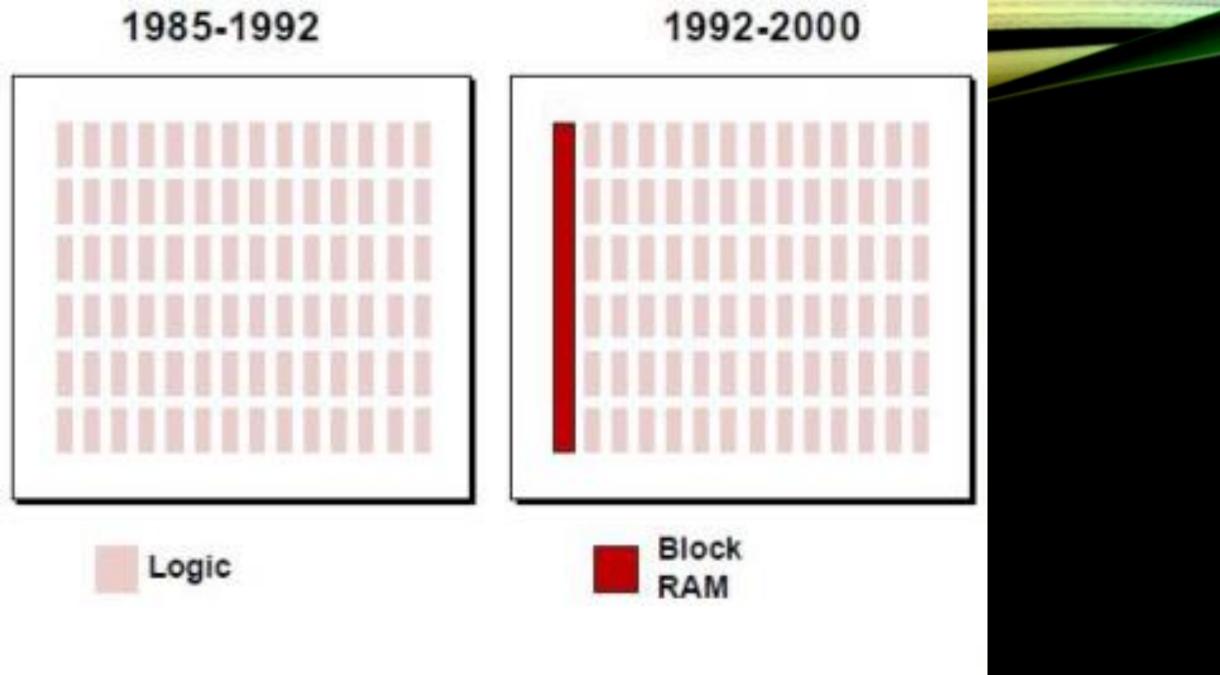
33



tracking bus

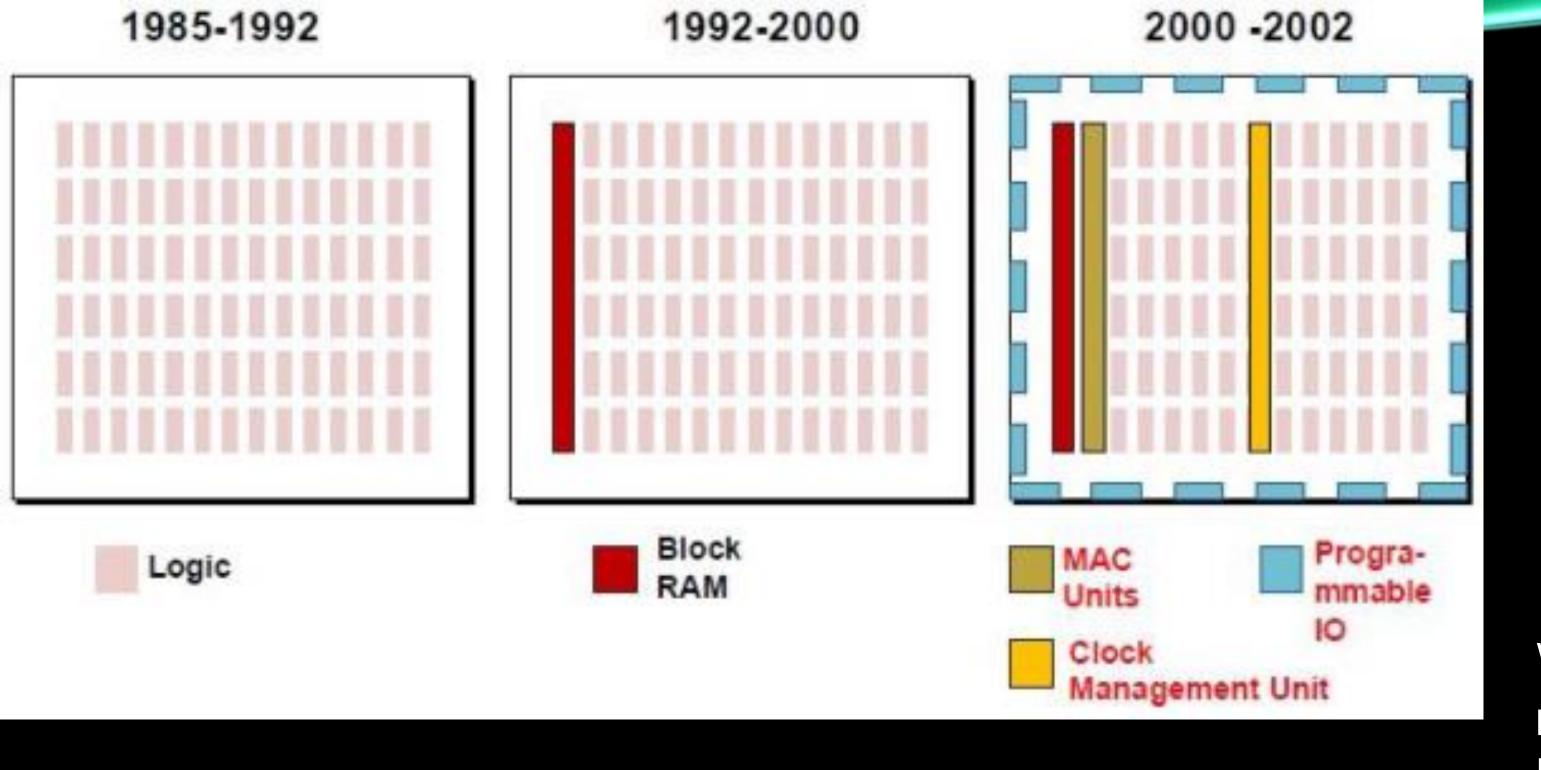


EVOLUTION OF FEATURES IN FPGAS



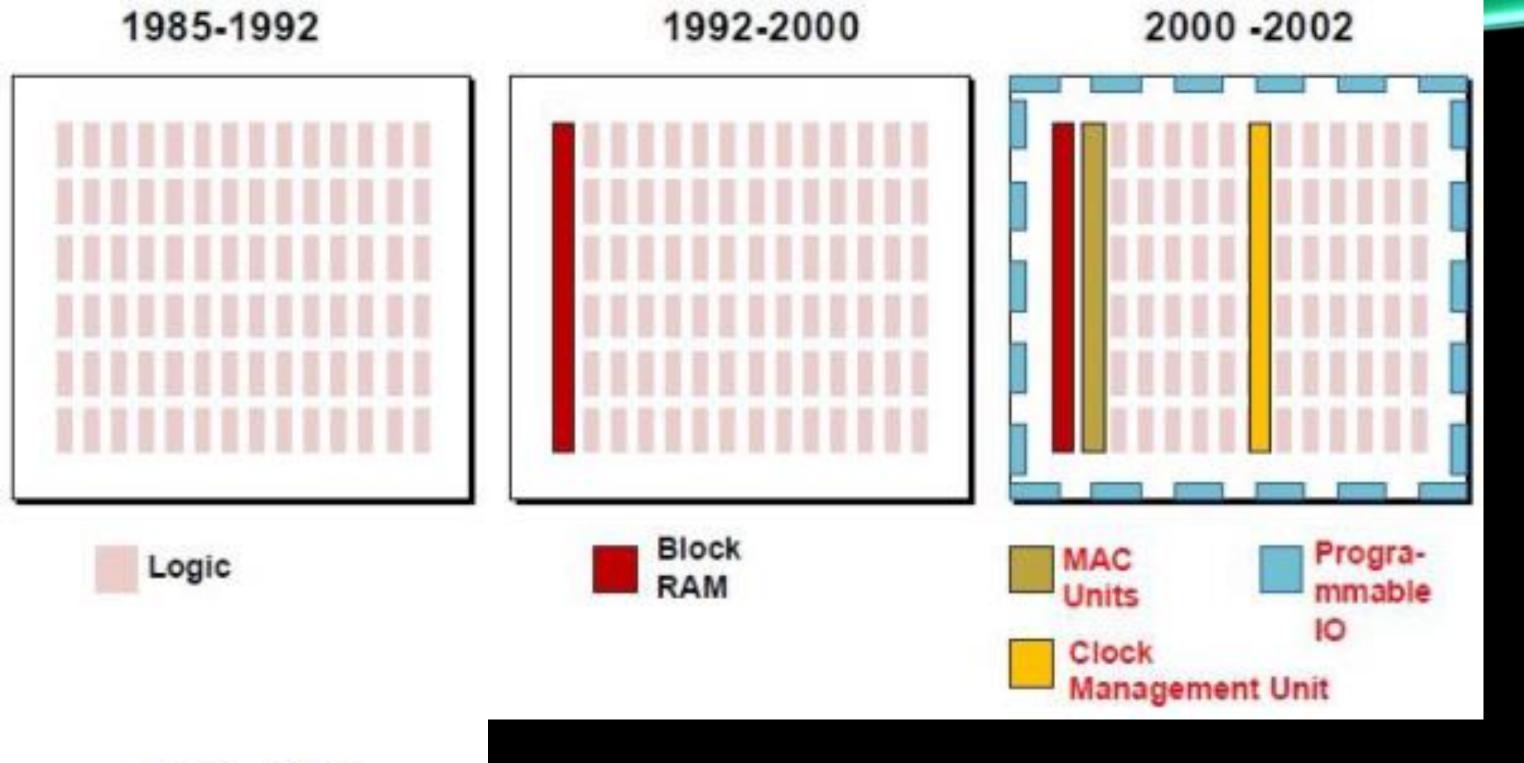
EVOLUTION OF FEATURES IN FPGAS

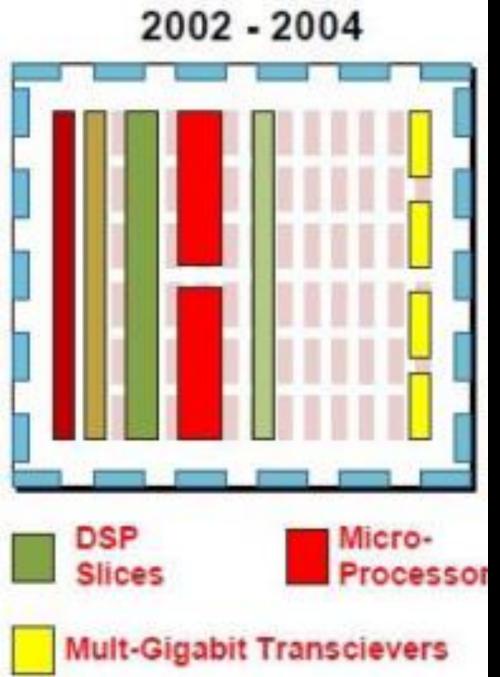
Who wants to waste all the LUTs as RAM?



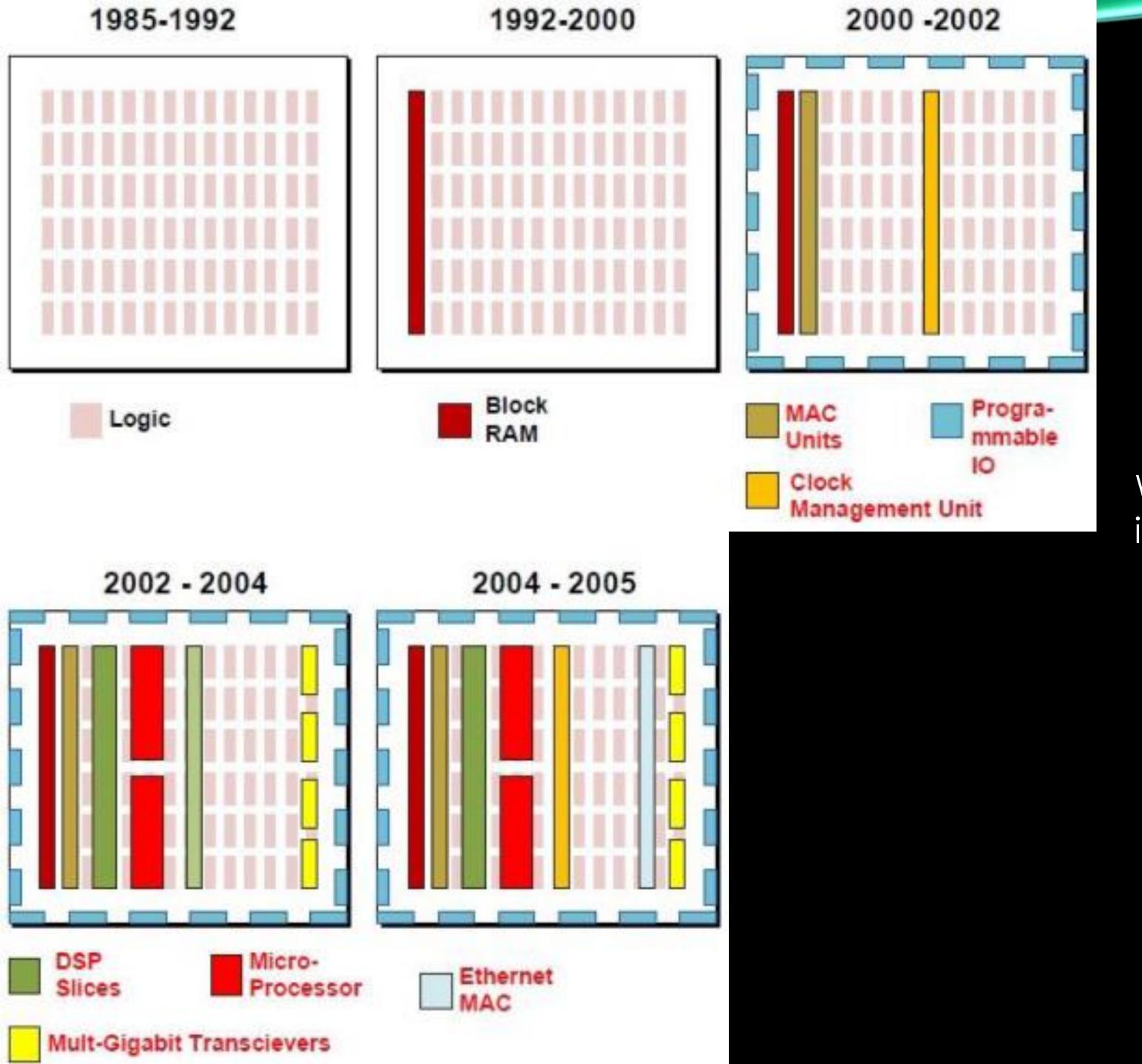
EVOLUTION OF FEATURES IN FPGAS

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





EVOLUTION OF FEATURES IN FPGAS

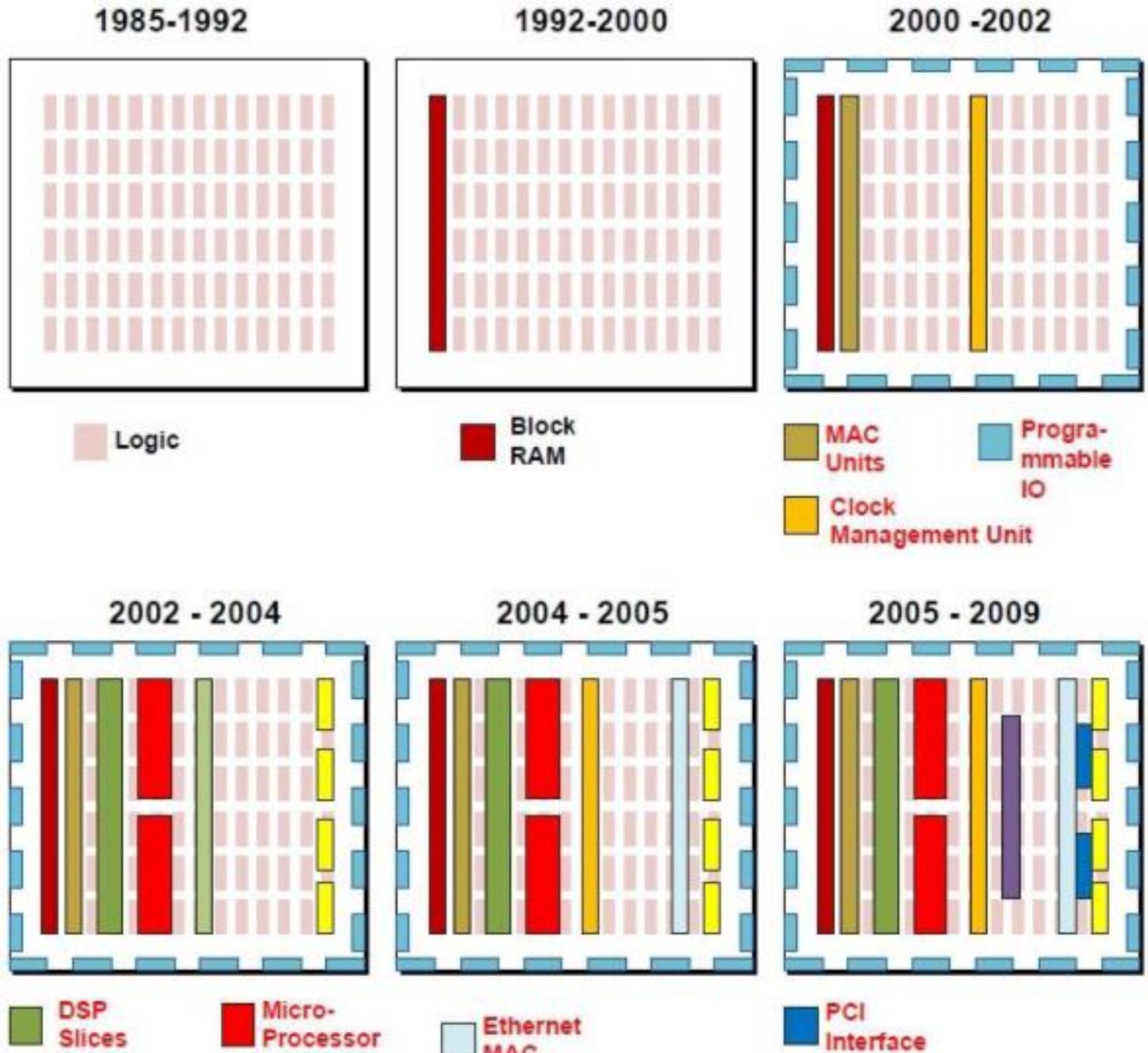


EVOLUTION OF FEATURES IN FPGAS

Who wants to waste LUTs AND re-inventindustry-standard blocks?

38

nt



Mult-Gigabit Transcievers

MAC

Interface System Monitor

EVOLUTION OF FEATURES IN FPGAS

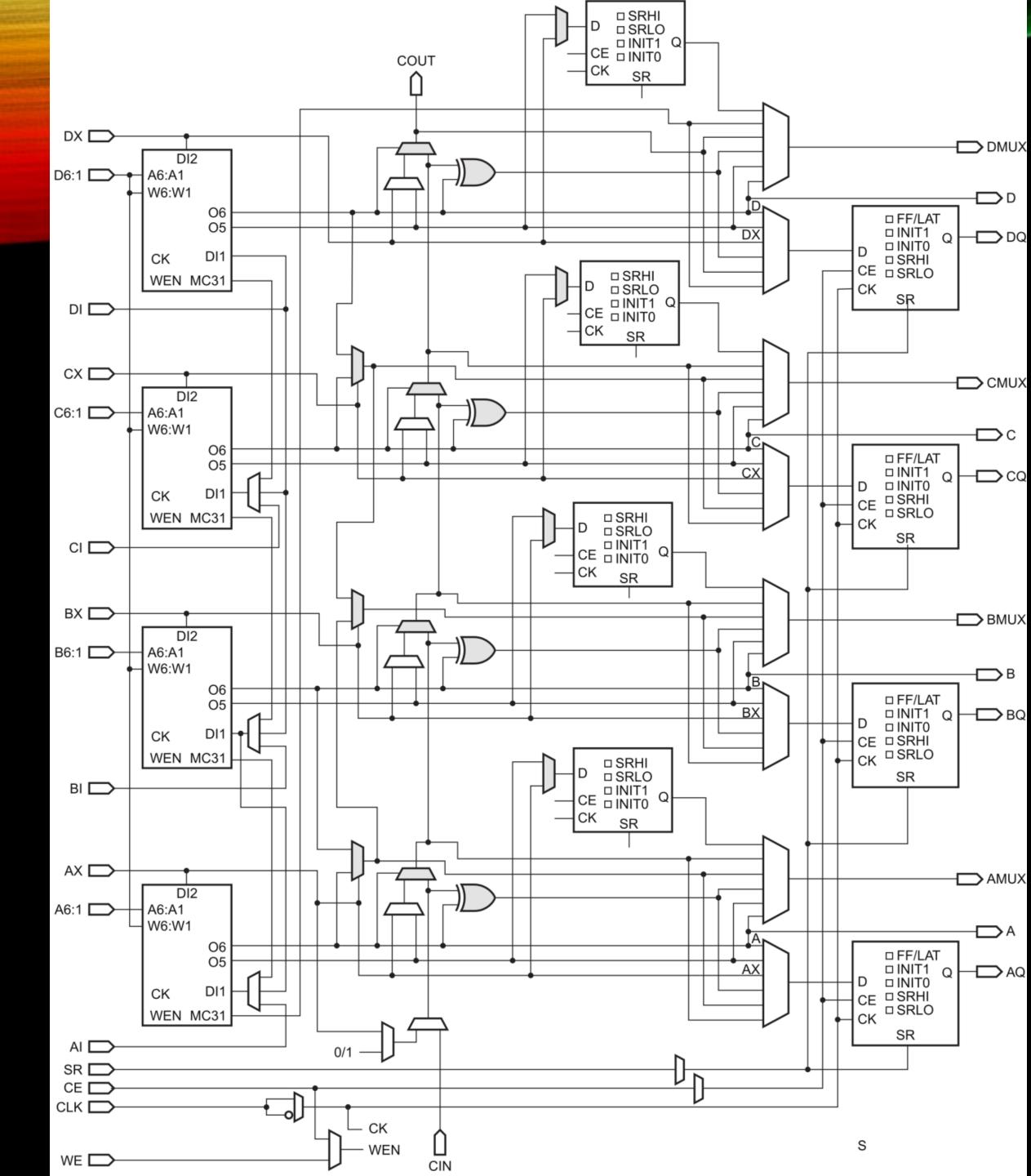
Who wants to waste LUTs AND re-invent industry-standard blocks?

- hundred MHz
- Latest generation Gen I/O up to 1.8Gbps
- Programmable logic standards \bullet
- Arms race Ever more and ever faster

A NOTE ON I/O

Traditionally, many hundreds of general-purpose pins (Gen I/O) up to a few

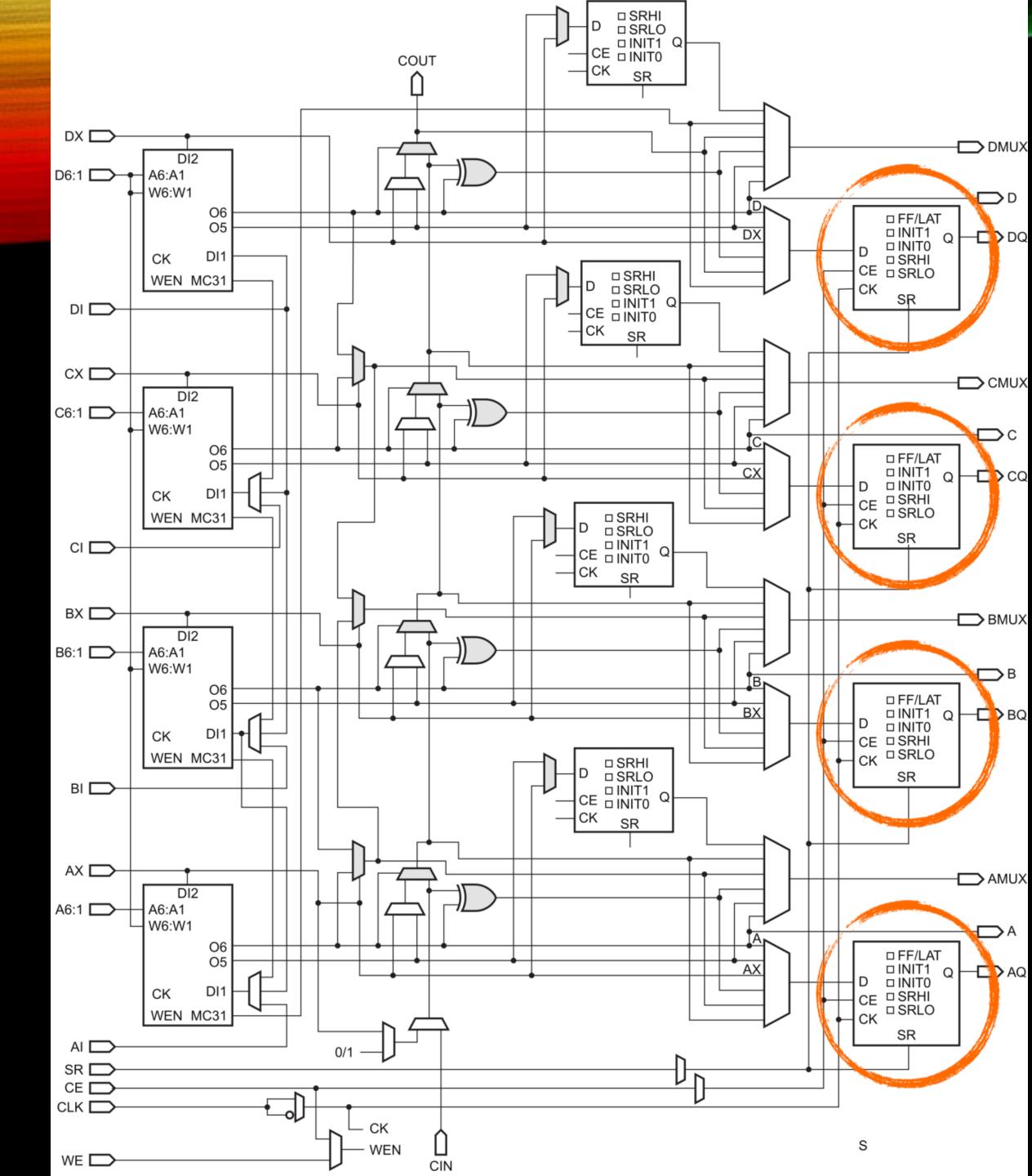
Since 2002, FPGAs have been adding dedicated Multi-gigabit transceivers



COMBINATORIAL LOGIC BLOCK

41

.

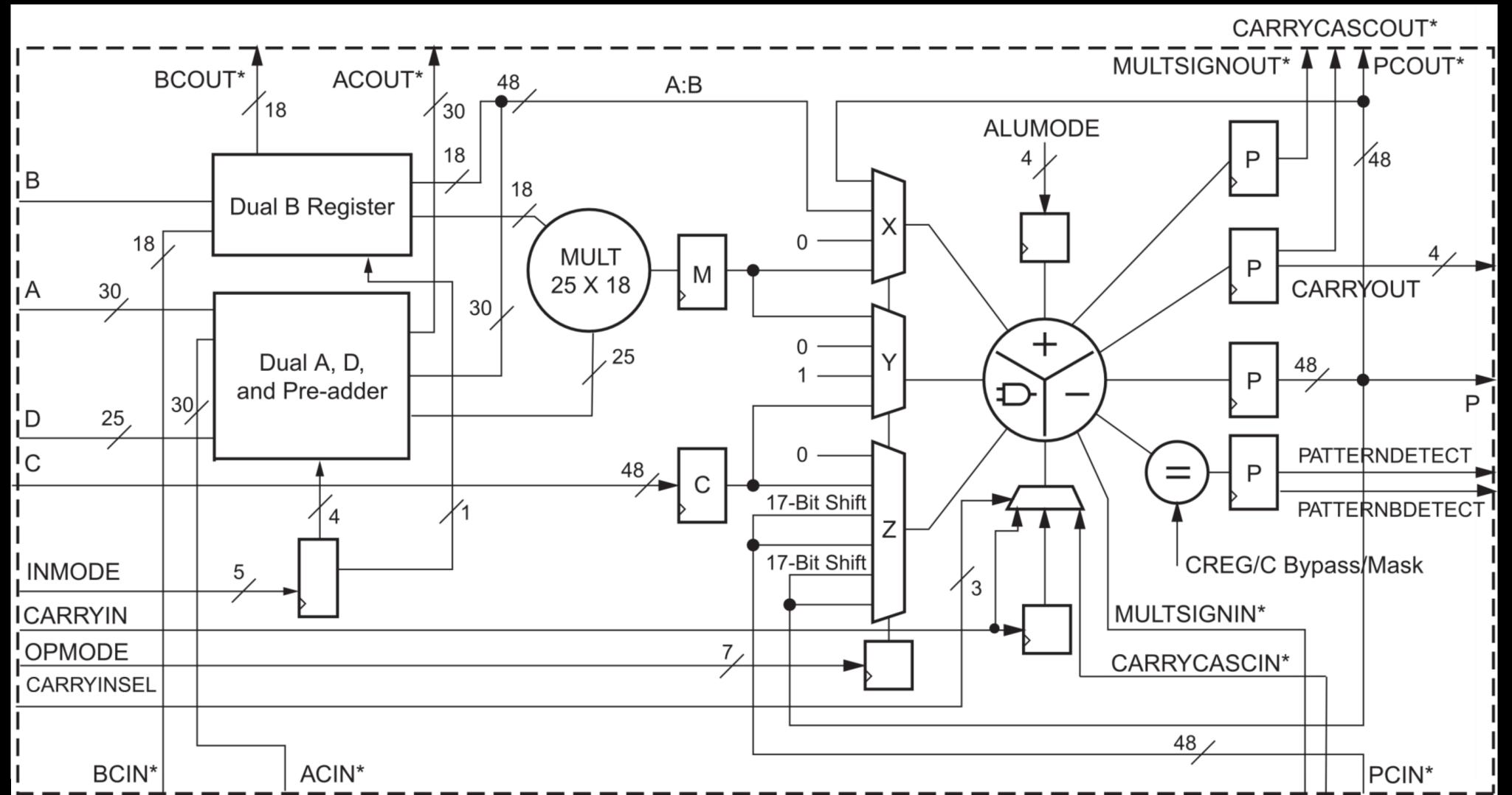


COMBINATORIAL LOGIC BLOCK

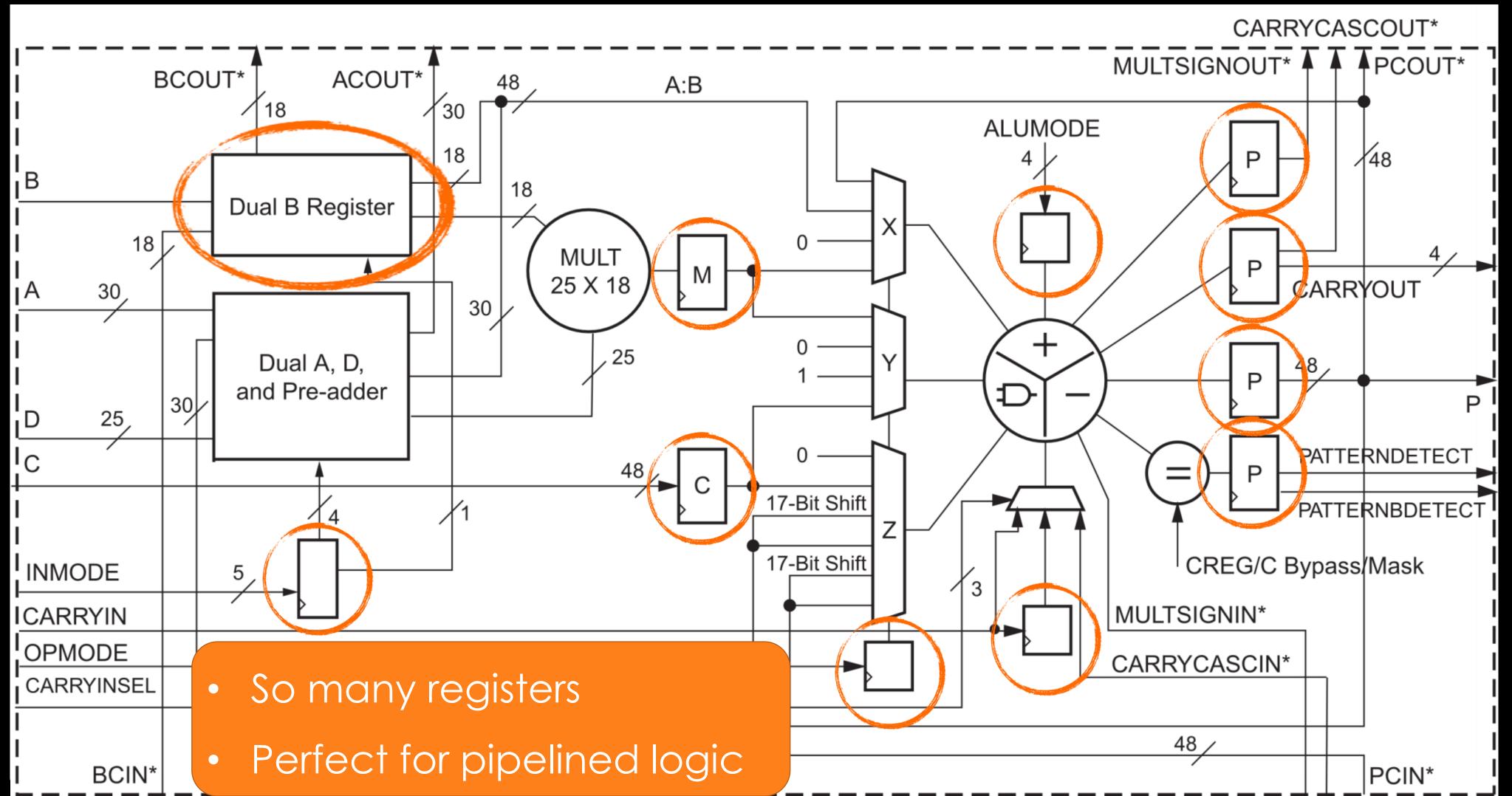
• Registers on the output of every cell

• Perfect for pipelined logic

INTEGRATED DIGITAL SIGNAL PROCESSING



INTEGRATED DIGITAL SIGNAL PROCESSING



BIGGEST XILINX "ULTRASCALE+" DEVICES

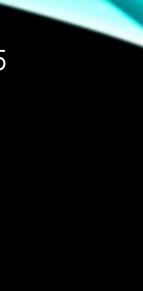
- Upwards of 2million logic cells
 - All clocked at up to 500MHz
 - Up to O(10¹⁵) operations/second
 - 1 PetaBOp ullet
- Upwards of 6000 DSPs
- All pipelined
- Fully programmable \bullet

PCle Gen

100G E Max Max GTY 3 GTM 58Gb

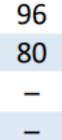
45

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
n3 x16/Gen4 x8 / CCIX ⁽¹⁾	-	_	_	8
150G Interlaken	9	6	8	0
Ethernet w/ KR4 RS-FEC	9	9	12	0
x. Single-Ended HP I/Os	832	624	832	1,976
x. Single-Ended HD I/Os	0	0	0	96
32.75Gb/s Transceivers	120	96	128	80
6b/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	-



VU19P 8,938 8,172 4,086 58.4

75.9 90.0 3,840 10.4 0 8 0 0 1,976



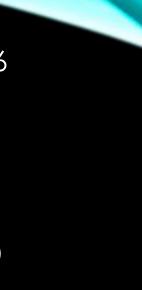
BIGGEST XILINX "ULTRASCALE+" DEVICES

- Upwards of 2million logic cells
 - All clocked at up to 500MHz
 - Up to O(10¹⁵) operations/second
 - 1 PetaBOp ullet
- Upwards of 6000 DSPs
- All pipelined
- Fully programmable \bullet

PCle Gen3

100G Et Max Max GTY 3 GTM 58G

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
13 x16/Gen4 x8 / CCIX ⁽¹⁾	_	_	_	8
150G Interlaken	9	6	8	0
Ethernet w/ KR4 RS-FEC	9	9	12	0
x. Single-Ended HP I/Os	832	624	832	1,976
x. Single-Ended HD I/Os	0	C	0	96
32.75Gb/s ransceivers	4.2 Tb	S 96	128	80
b/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	-



VU19P 8,938 8,172 4,086 58.4 75.9 90.0 3,840 10.4 0 8 0 0 1,976 96

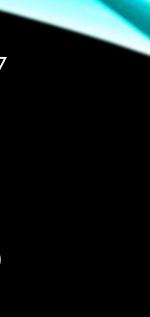
BIGGEST XILINX "ULTRASCALE+" DEVICES

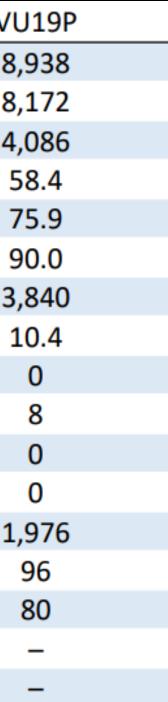
- Upwards of 2million logic cells
 - All clocked at up to 500MHz
 - Up to O(10¹⁵) operations/second
 - 1 PetaBOp
- Upwards of 6000 DSPs
- All pipelined
- Fully programmable
- So what is the catch?

PCIe Gen3

100G Et Max Max GTV 3 GTM 58G

Device Name	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)		2,835	3,780	8,938
CLB Flip-Flops (K)	2,364	2,592	3,456	8,172
CLB LUTs (K)		1,296	1,728	4,086
Max. Dist. RAM (Mb)	36.1	36.2	48.3	58.4
Total Block RAM (Mb)		70.9	94.5	75.9
UltraRAM (Mb)		270.0	360.0	90.0
DSP Slices	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)		28.7	38.3	10.4
PCle [®] Gen3 x16		3	4	0
3 x16/Gen4 x8 / CCIX ⁽¹⁾	_	_	_	8
150G Interlaken	9	6	8	0
Ethernet w/ KR4 RS-FEC	9	9	12	0
x. Single-Ended HP I/Os	832	624	832	1,976
. Single-Ended HD I/Os	0		0	96
32.75Gb/s Transceivers	4.2 Tb	S 1 96	128	80
b/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	-





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 ullet



FPGAS: WHAT'S THE CATCH?

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

"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

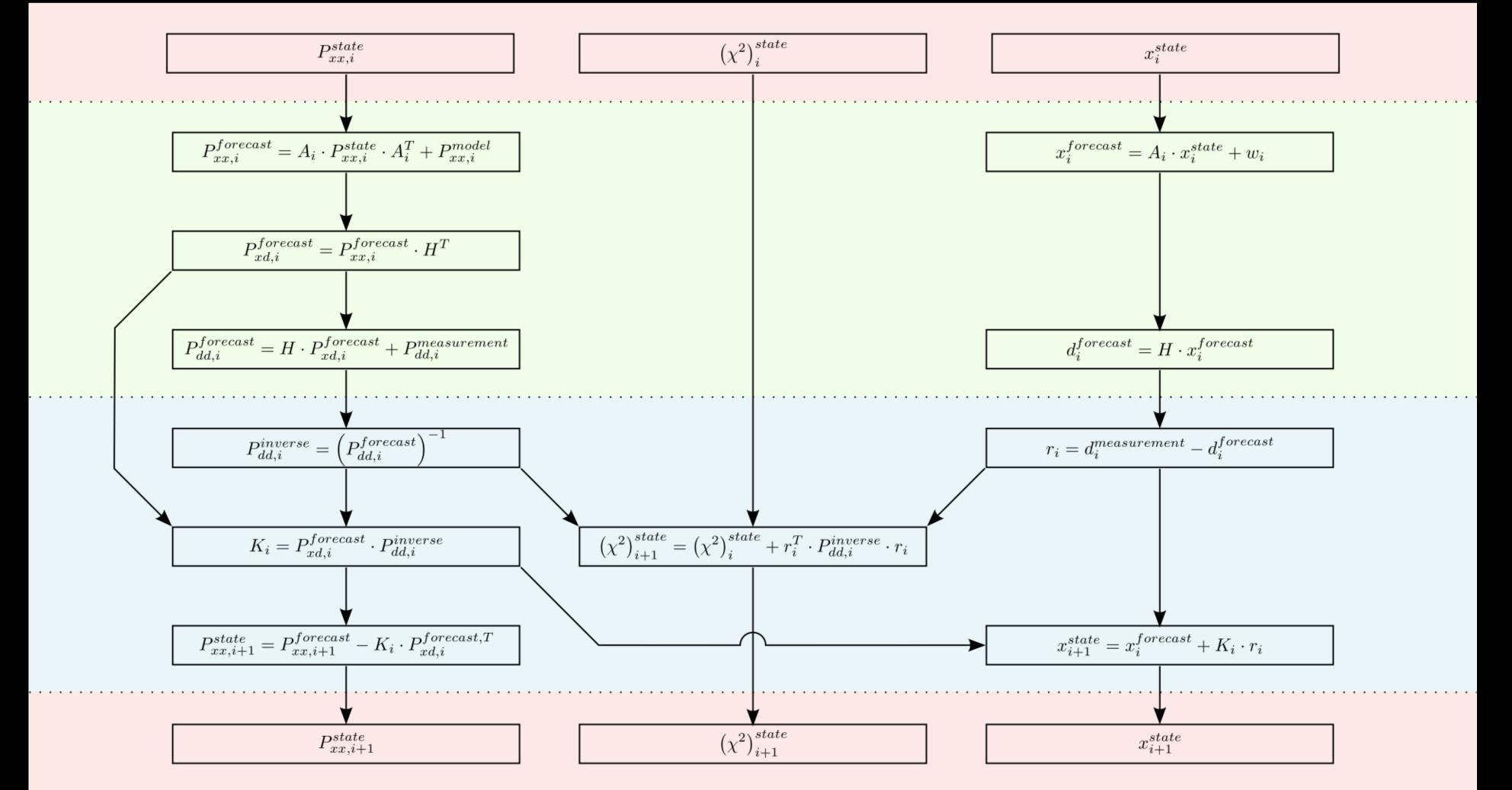
FPGAS: WHAT'S THE CATCH?

- 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 \bullet
- The chip is just the start needs to be attached to something ullet
- You are also responsible for the infrastructure \bullet

• Keep your data-flow fully flow-forwards

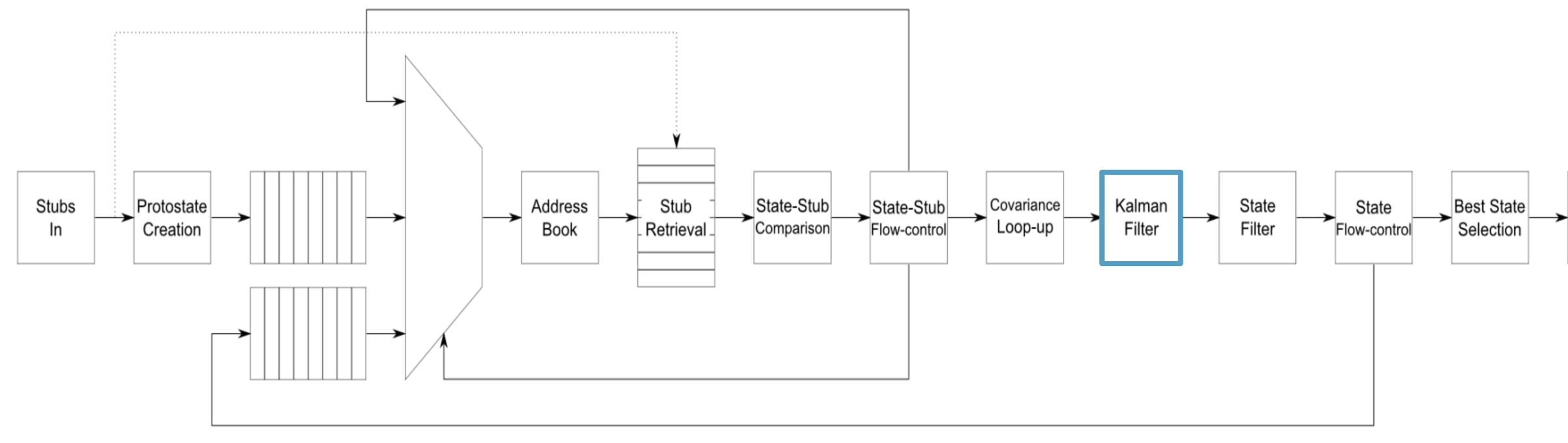
- No iterations
- or at least
- Flatten your loops

HOW TO PRESERVE YOUR SANITY USING FPGAS





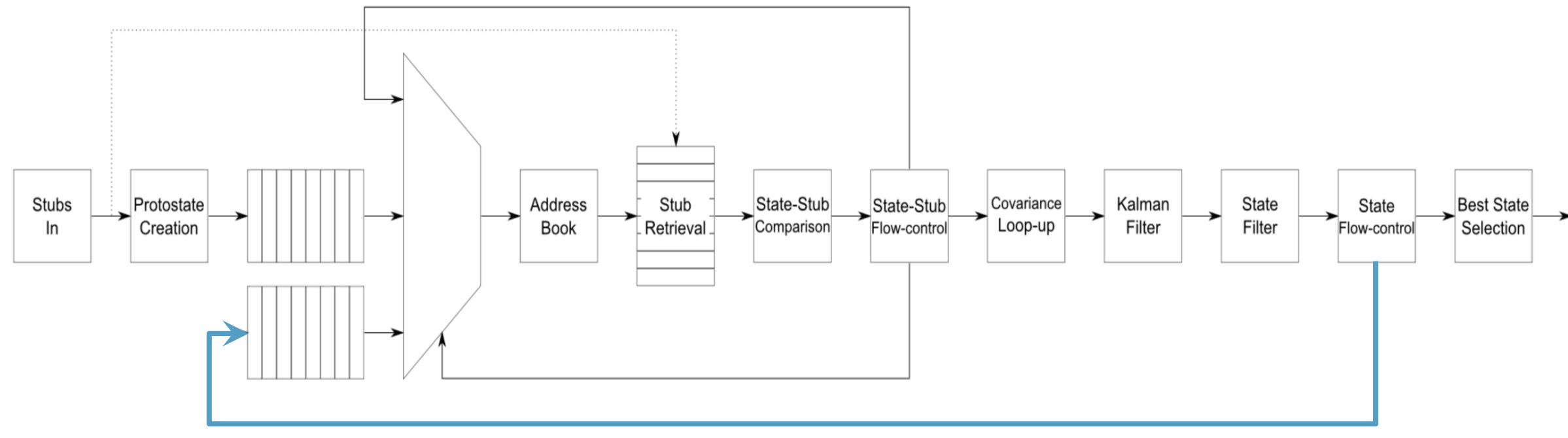




The maths is a relatively simple part of a more complex whole



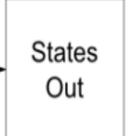
States Out

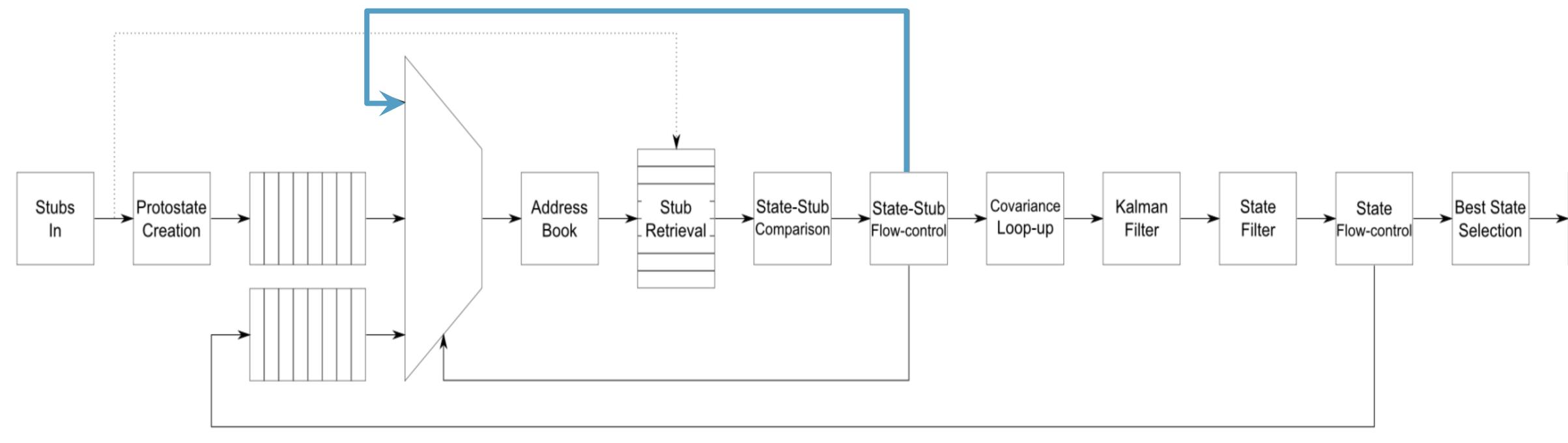




Kalman Filter is iterative



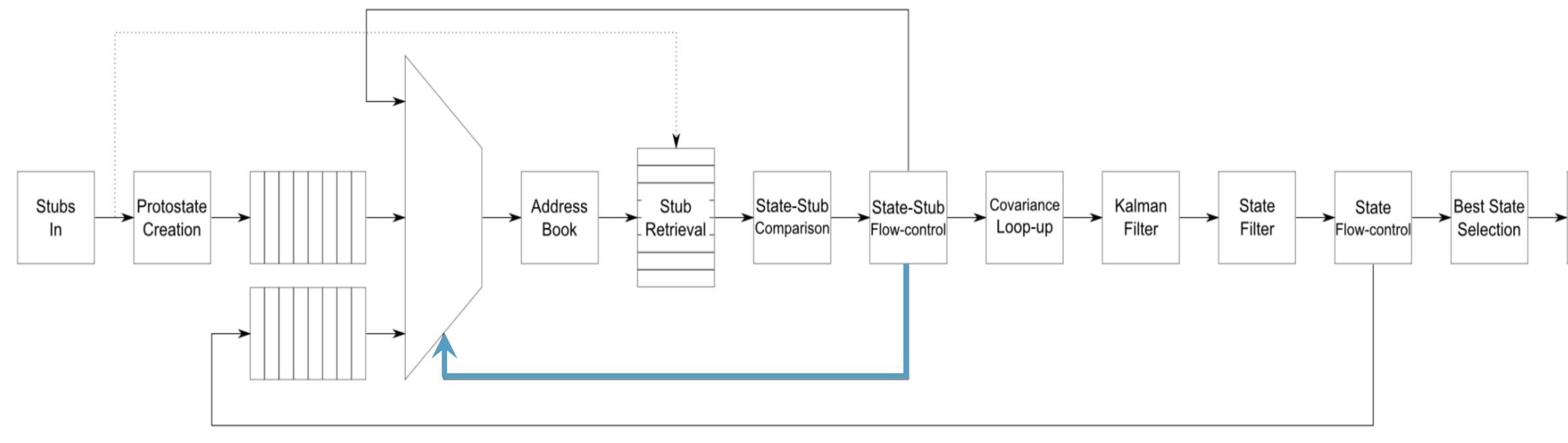




Kalman Filter must handle combinatorics







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?

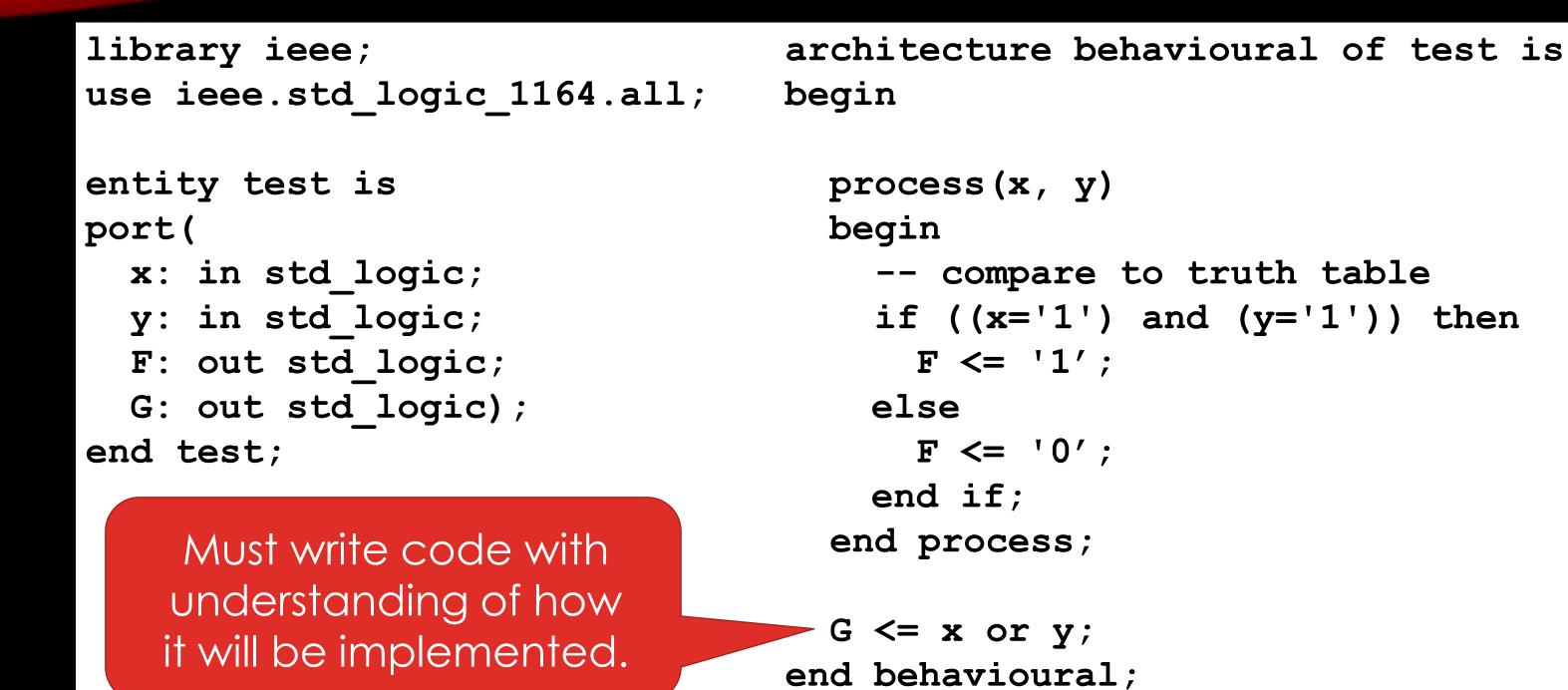


HARDWARE DESCRIPTION LANGUAGES

- 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!



- Can also enter code via schematic entry:
 - Easier to navigate, but not vendor independent
 - \bullet

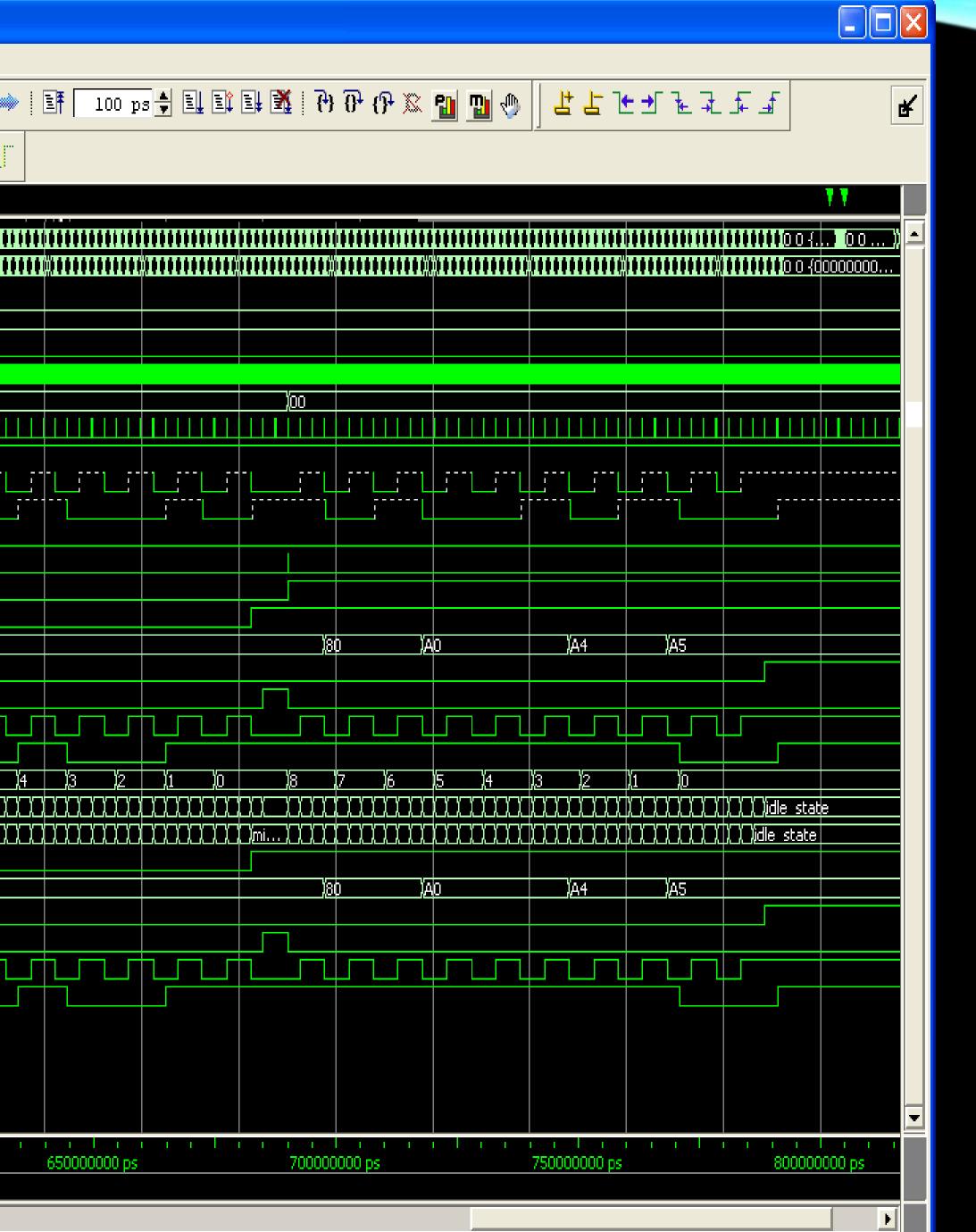
EXAMPLE

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			
<u>File E</u> dit <u>V</u> iew <u>A</u> dd F <u>o</u> rmat <u>T</u> ools Debu	<u>g W</u> indow		
🗋 • 🚅 🔲 🎲 🚭 🐰 🖻 🛍 🏠 :	∷∦ #4 ⊡−⊡	д і 🕵 💻 🤹 і	🔄 🔛 🛺 🎇 🕺 🛉 🔶 👒
	D : (7)	3 3 3 3 3 3 3 1 1 1	
🖧 - 🐔 🗳 🥰 📐 🖳 🏦 👫	3+ ∰ ℃		
Messages			
	0 0 {ZZZZZZZZZ		***********************
➡-◆torcommfpga/ttcrx_i2c/comm_reply	0 0 {000000000	<u>U</u>	
	0000100	0000100	
pga/ttcrx_i2c/control/async_reset	0		
atorcommfpga/ttcrx_i2c/control/clkommfpga/ttcrx_i2c/control/data_w	0	A5	
commfpga/ttcrx_i2c/control/enable	0		
orcommfpga/ttcrx_i2c/control/r_nw	1		
torcommfpga/ttcrx_i2c/control/scl_i	H	,	
 orcommfpga/ttcrx_i2c/control/sda_i mfpga/ttcrx_i2c/control/start_add 	н 0		
fpga/ttcrx_i2c/control/start_data	0		
🔹 \landtorcommfpga/ttcrx_i2c/control/stop	1		
torcommfpga/ttcrx_i2c/control/ack	1		
commfpga/ttcrx_i2c/control/data_r	A5 1	00	
mfpga/ttcrx_i2c/control/mid_trans	0		
orcommfpga/ttcrx_i2c/control/scl_o	1		
rcommfpga/ttcrx_i2c/control/sda_o	1		
tratorcommfpga/ttcrx_i2c/control/iga/ttcrx_i2c/control/current_state	0 idle_state	0 idle state ()))))	<u>)8 _]7 _]6 _]5</u>
ga/ttcrx_i2c/control/next_state	idle_state	idle state	
commfpga/ttcrx_i2c/control/ack_cld	1		
fpga/ttcrx_i2c/control/data_r_cld	A5	00	
 commfpga/ttcrx_i2c/control/idle_cld ga/ttcrx_i2c/control/mid_trans_cld 	1	L	
mmfpga/ttcrx_i2c/control/scl_o_cld	1		
mfpga/ttcrx_i2c/control/sda_o_cld	1		
AR Now	835368850 ps		60000000 ps
Cursor 1	0 ps		
559897291 ps to 816361126 ps Now:	835,368,850 ps	Delta: 5 1	



"Event display"

ModelSim SE-64 10.1b	
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>C</u> ompile <u>S</u> imulate A <u>d</u> d T<u>r</u>anscript Tools Layo<u>u</u>t Boo<u>k</u>marks <u>W</u>indow	Help
🖹 - 🖆 🗟 🐇 🖄 🆓 🖄 🖄 🖄 🖄 🖓 👔 🕼 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘 👘	
🔍 🔍 🔍 🐧 ColumnLayout Default 💌 📃 🛄 🛄 👔 🧊 🗍 🕽 🖓 🚓 🍖 Search:	
🕇 🎨 🛊 👔 🔝 👔 🕹 Layout Simulate 🔽 🛛 🦓 🦓 - 🥰 💽 🖓 🆓 - 🥰 💽 🚯 🔛 🚯 🔛 🖓	
Image: Second	
Msgs	
- JET SUMS	
- JET VETO	
王 🔷 /testbench/r {{{TRUE TRUE } TRUE TRUE} {TRUE TRUE} {TRUE} {TR	
$$ JET VETO FILTERED $$ JET VETO FILTERED $+$ /testbench/f {{{{0} 0 FALSE} {{0} 0 FALSE} {{0} 0 FALSE} {{0} 0 } } } / } / } / } / } / } / } / } /	
Let Now 2000 ns 1800 ns 400 ns 600 ns 800 ns 1000 ns 1200 ns 1400 ns 1600 ns 1800 ns 2000	IIIIII ns
6 Cursor 1 1999,462 ns	2 ns
Image:	× 5 + ==
# PHI = 131 58 475 427 98 55 24 0 439 390 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 # v · · · · · · · · · · · · · · · · · ·	
# PHI = 101 117 491 400 220 408 82 496 263 210 35 255 323 338 318 155 459 273 226 324 465 466 144 322 297 495 176 182 232 124 471 105 194 410 71 482 438 17 215 487 251 # • • • • • • • • • • • • • • • • • • •	
# PHI = '9 i 7 61 49 9 201 466 30 190 395 475 94 444 198 355 278 7 449 419 136 394 i 78 115 39 17 320 478 64 474 129 327 155 30 326 123 428 108 474 274 62 277	
# PHI = 8 168 137 495 237 178 82 375 89 267 494 317 239 400 9 129 238 411 8 384 11 99 235 266 432 300 211 179 465 267 301 419 432 475 144 100 169 164 260 153 447 # PHI = 7 91 276 374 13 320 463 65 72 74 389 17 472 6 110 369 315 346 420 482 207 309 481 8 294 130 498 151 237 435 140 15 45 211 427 114 159 117 151 444 354	
# PHI = 71 91 276 374 131 320 4631 651 721 74 3891 171 4721 61 1101 3691 3151 3461 4201 4821 2071 1 3091 4811 81 2941 1301 4981 1511 2371 4351 1401 151 451 2111 4271 1141 1591 1171 1511 4441 3541 #	
# ++++++++++	
<pre># PHI = 5 257 438 453 498 395 7 379 238 188 314 384 226 292 41 499 146 245 14 413 252 346 146 195 316 412 71 16 82 25 216 226 380 392 145 305 309 313 506 268 423 # +++++++++++++-</pre>	
<pre># PH1 = 41 1 431 271 3931 111 3301 171 4701 4011 3061 4311 551 5021 3781 1351 2491 1281 4351 2971 2021 3741 1 2721 1921 2111 941 3181 3491 5071 4561 3731 3111 1921 1991 2401 2441 31 761 321 1991 3711 4761 #</pre>	
<pre># PH1 = 31 + 3011 461 4221 3231 3001 1951 461 3821 2931 4851 1841 4301 3251 1851 1981 1921 2781 3271 141 4341 + 611 861 4661 2551 741 3351 821 3971 241 4751 1981 3971 351 4781 51 4911 4651 1941 4921 4471 #</pre>	
<pre># PHI = 21 2591 1501 2781 4691 2821 3951 4111 4731 2841 2281 1501 3201 1851 1941 2801 4081 3501 2171 521 3701 2071 971 221 481 971 4421 2271 1881 2421 4181 4481 1791 4501 1971 401 01 4851 571 5781 2871 # ++++++++++++++++-</pre>	
<pre># PH1 = 11 + 412 + 3331 3881 4941 1331 3841 1331 3841 1331 381 4941 1331 381 4941 1331 381 4941 1331 381 4941 1331 381 4941 381 491 491 491 491 491 491 491 491 491 49</pre>	
Now: 2 us Delta: 8 sim:/testbench	/

TESTBENCH SUITE

Clock-by-clock summary

	Transcript 🗁
#	Jets 9x1 Sum[2] : latency 4 clks : Matches expected latency
#	<<<<<< Clock 8 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
	Jets 9x1 Sum[3] : latency 4 clks : Matches expected latency
	<<<<<< Clock 9 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
#	Jets 9x1 Sum[4] : latency 4 clks : Matches expected latency Jets 1x9 Sum[0] : latency 8 clks : Matches expected latency
# # #	<<<<<< Clock 10 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
# # #	<pre><<<<<< Clock 11 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>
# # #	<pre><<<<<< Clock 12 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>

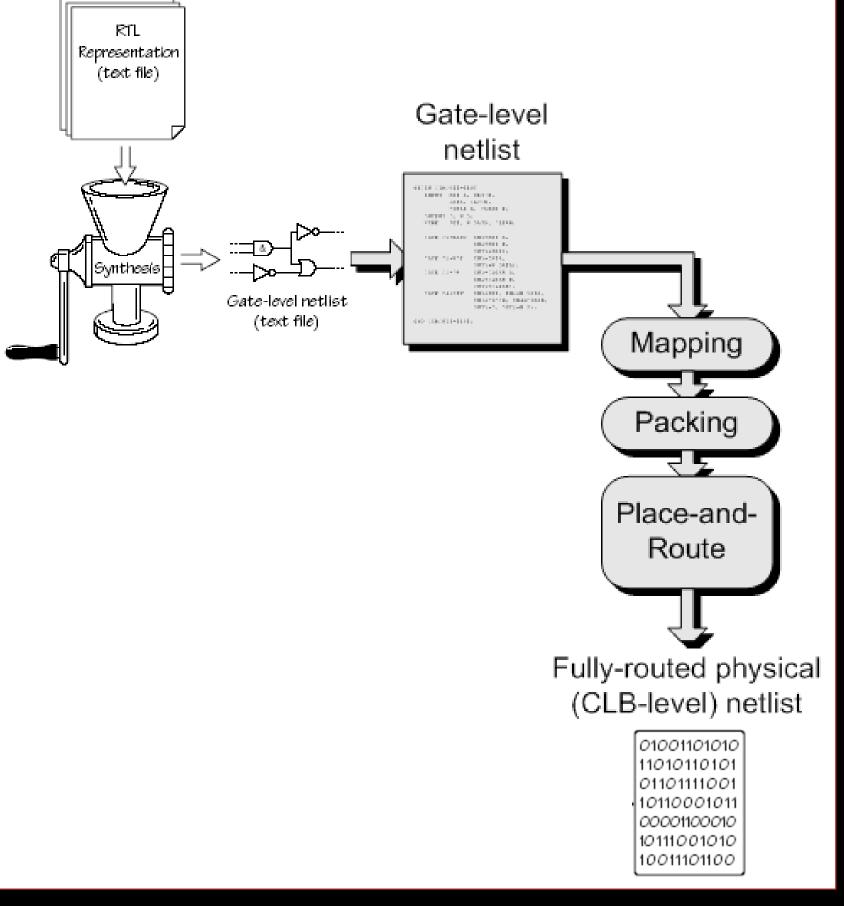
End-of-event summary

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

TESTBENCH SUITE

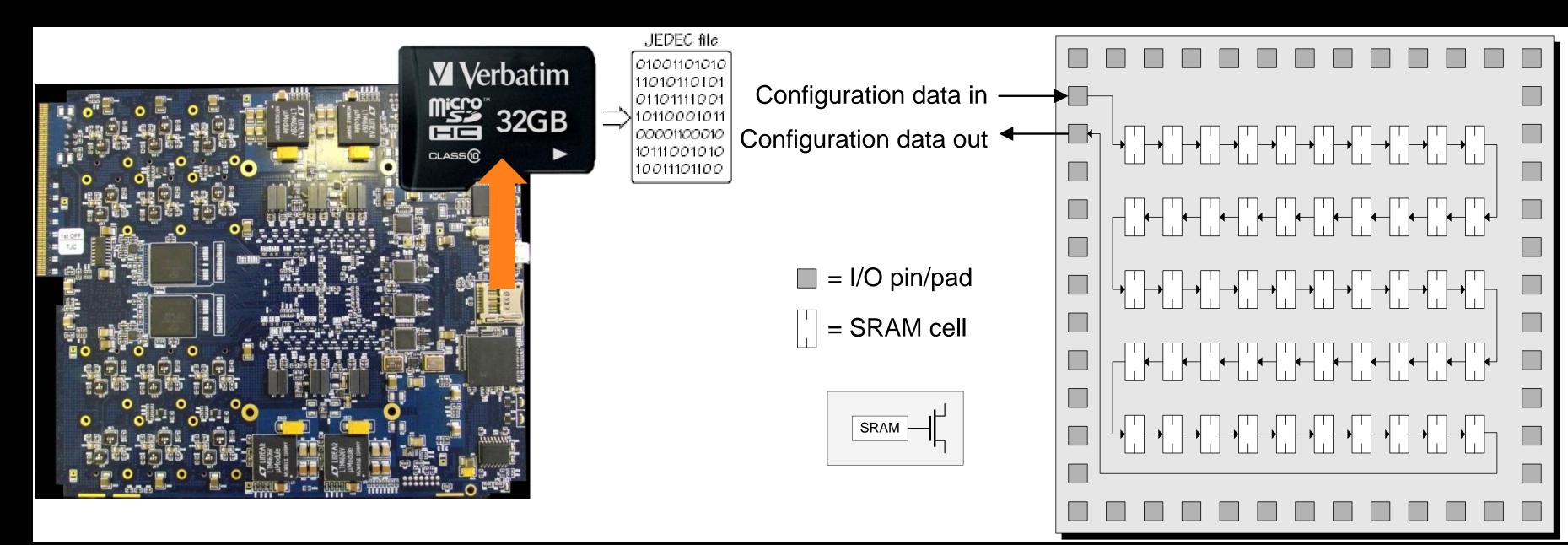
DESIGNING LOGIC WITH FPGAS

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



Millions of SRAM cells holding LUTs and Interconnect Routing

- Keep bit patterns describing the SRAM cells in non-Volatile Memory e.g. PROM or memory card
- Configuration takes ~ secs



CONFIGURING AN FPGA

Volatile Memory: Lose configuration when board power is turned off.

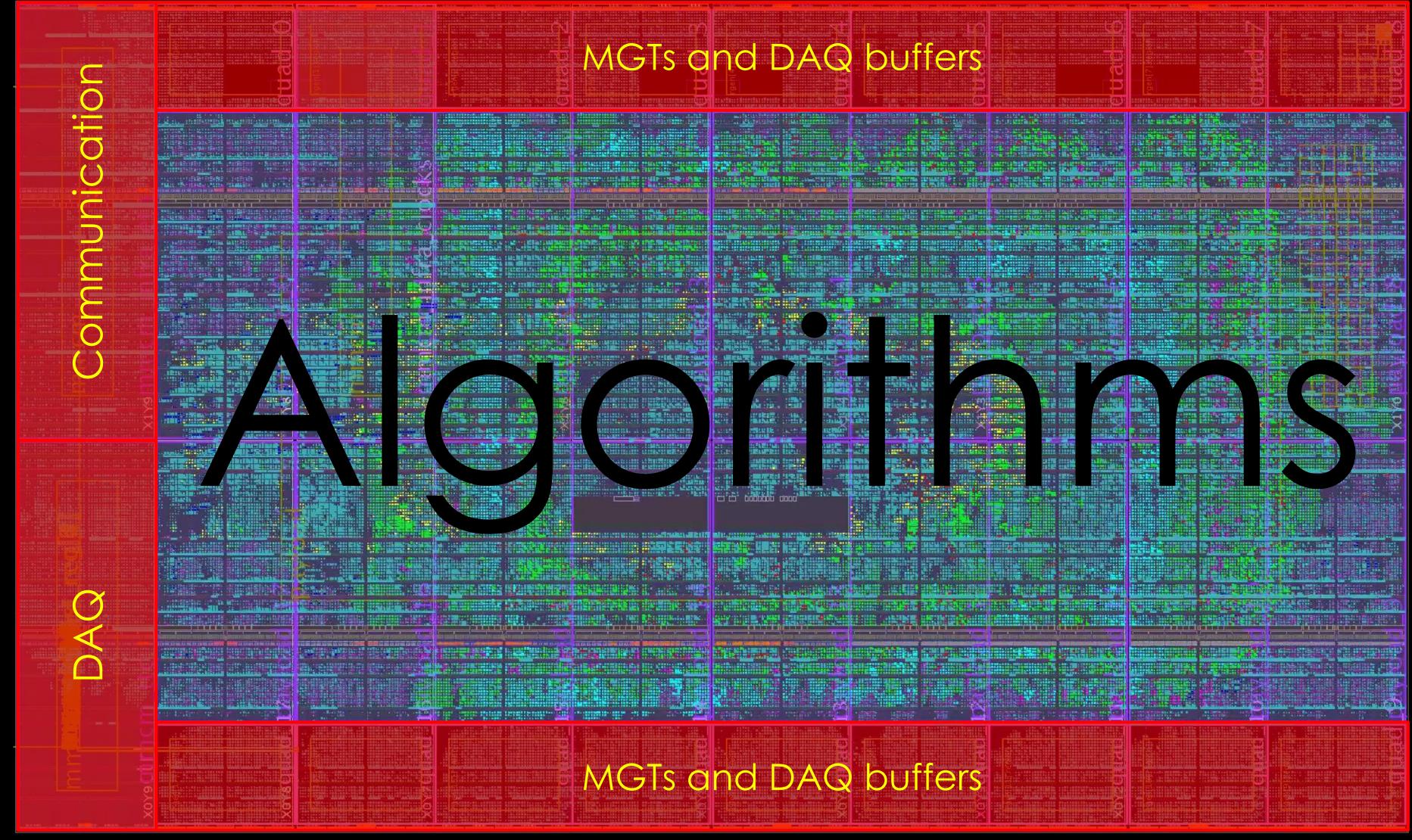
IT DOESN'T WORK: HOW TO DEBUG

- 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) ightarrow
- Use chipscope
 - Rebuild design with embedded logic analyser
 - Can be a bit like quantum mechanics ightarrow
 - 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

	d G G G				
Ę,					
XIX					
			Managering solding the party of		
Xov9 Cut					

FLOORPLAN OF FIRMWARE IN MP7

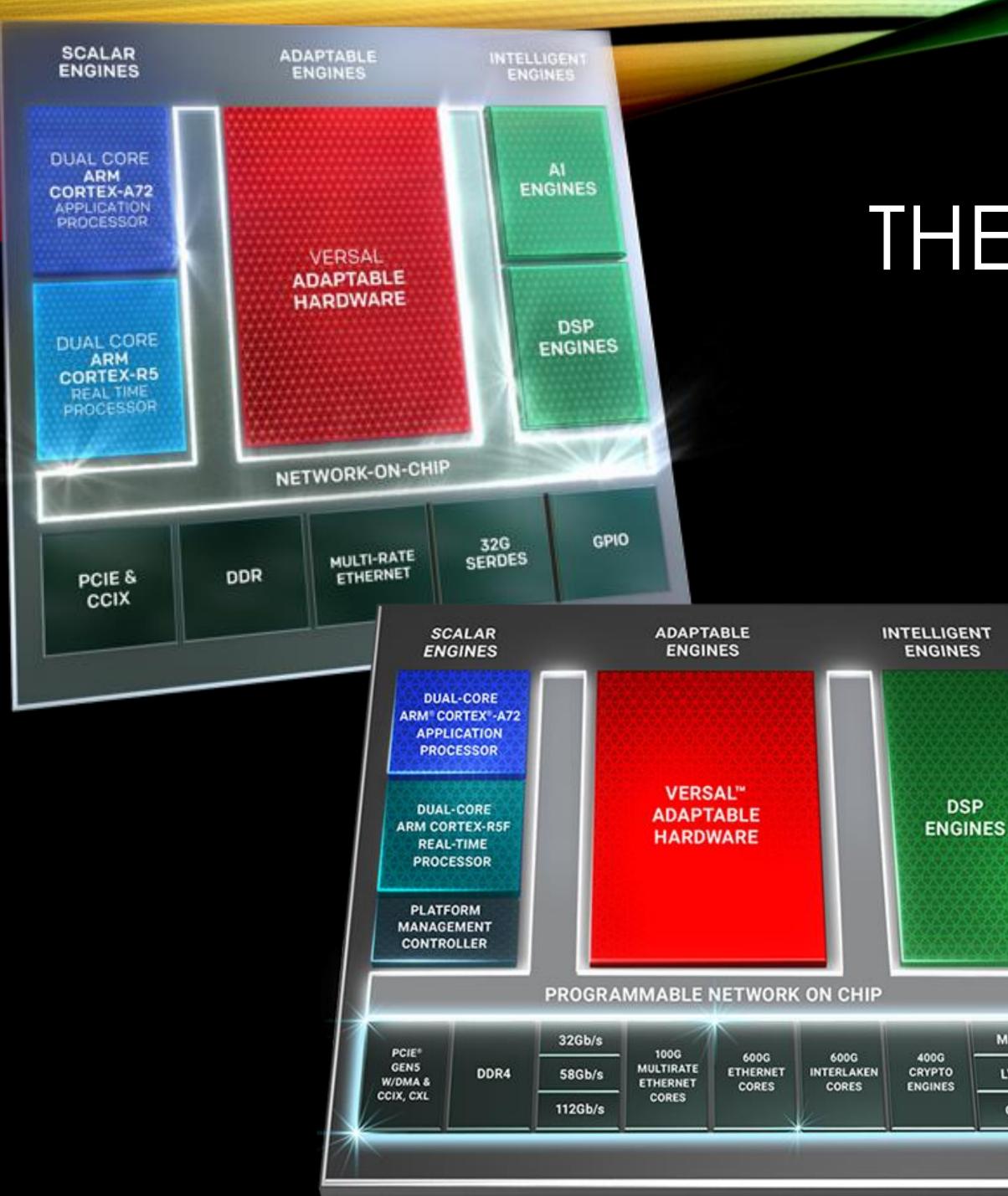


- FPGAs are expensive (high-end £10k-100k cf. £100)
- FPGAs are power-hungry
- Programming FPGAs is like designing logic circuits not like programming sequential microcontrollers
- Large firmware build-times are tens of hours or days
- Floating-point ops and iterative algorithms awkward in FPGAs (That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

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

- 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

CONCLUSION



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...

MIPI

LVDS

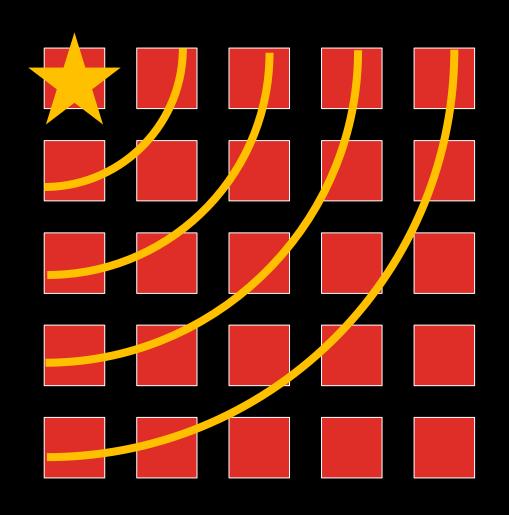
GPIO



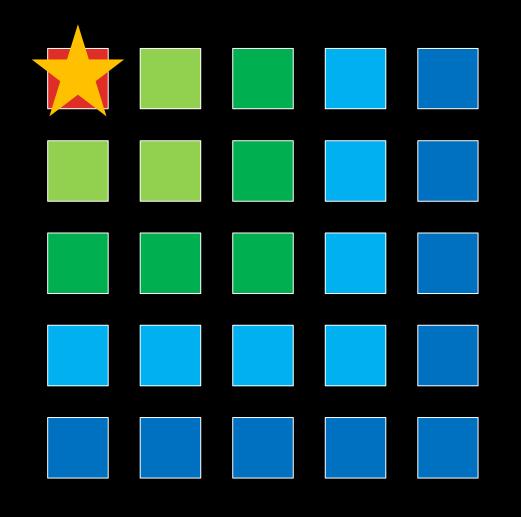


THANK YOU Any questions?

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