CPE 462
VHDL: Simulation and Synthesis

Topic #01 - Introduction to reconfigurable computing

Goals of this class

1. **Learn principles of reconfigurable computing**

2. Master the principles of VHDL and understand its features and limitations

3. Acquire hands-on experience with synthesis tools, reconfigurable hardware and simulators

4. Strengthen engineering skills through tangible class projects that can be shown to potential employers

Methods for executing computations: software

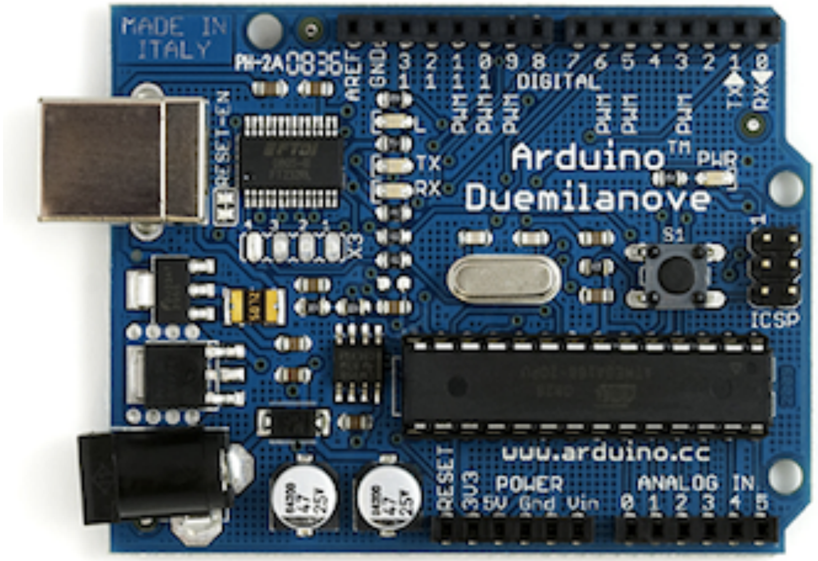
- Software programmed microprocessors

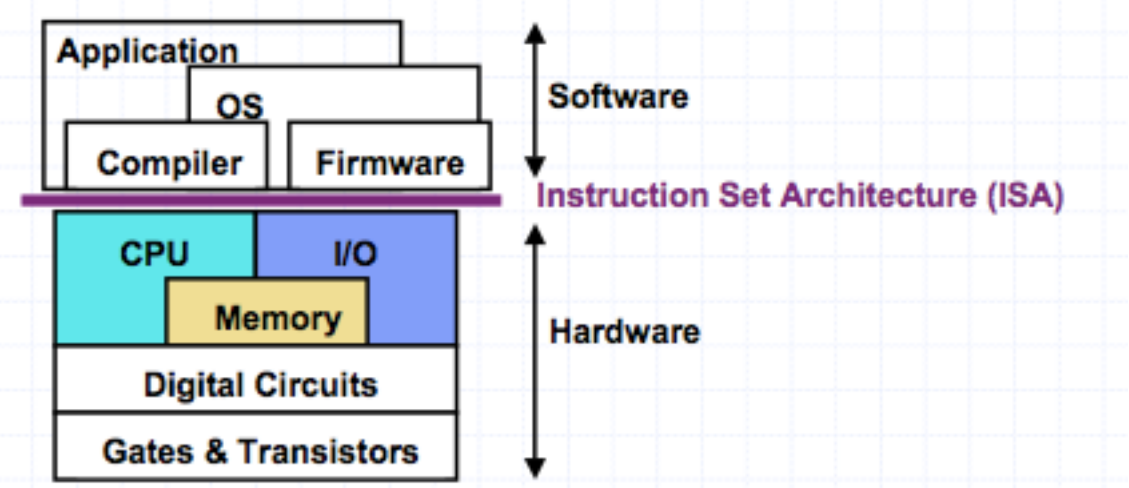
Advantages:

- Software is flexible to change

Disadvantages:

- Performance can suffer if clock is not fast
- Fixed instruction set by hardware





Methods for executing computations: ASIC

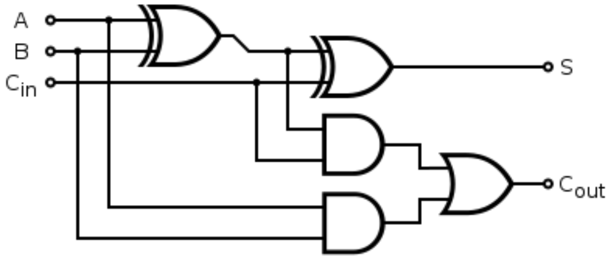
- Straight in hardware
- **A**pplication **S**pecific **I**ntegrated **C**ircuits

Advantages:

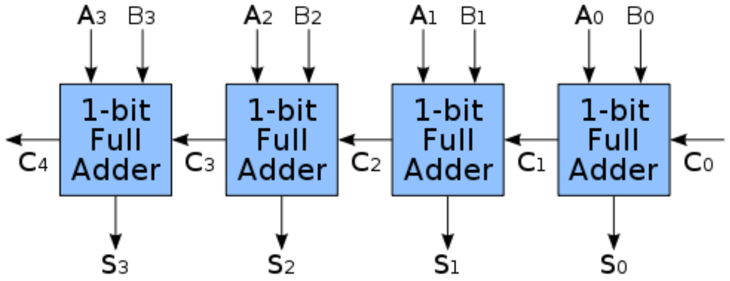
- Extremely fast

Disadvantages:

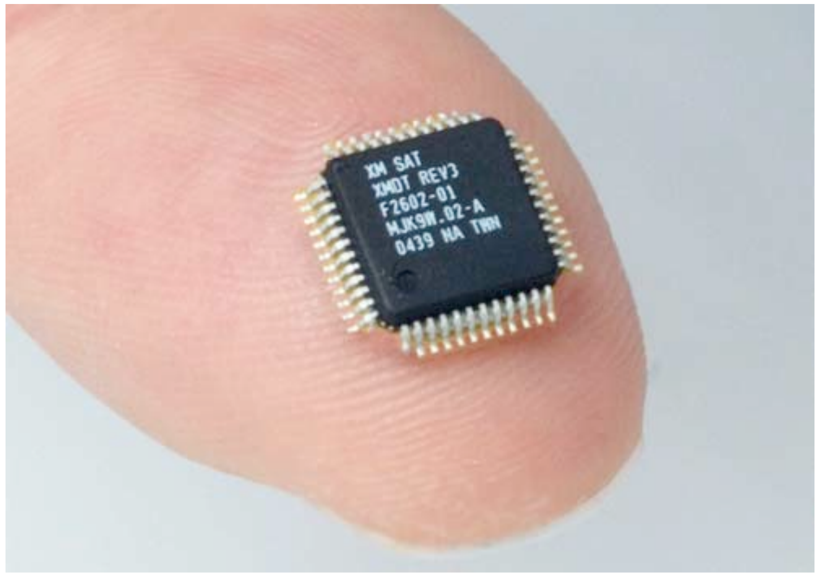
- Algorithms can't be modified once printed
- Expensive



1 Bit Adder



4 Bit Adder



Methods for executing computations: reconfigurable computing

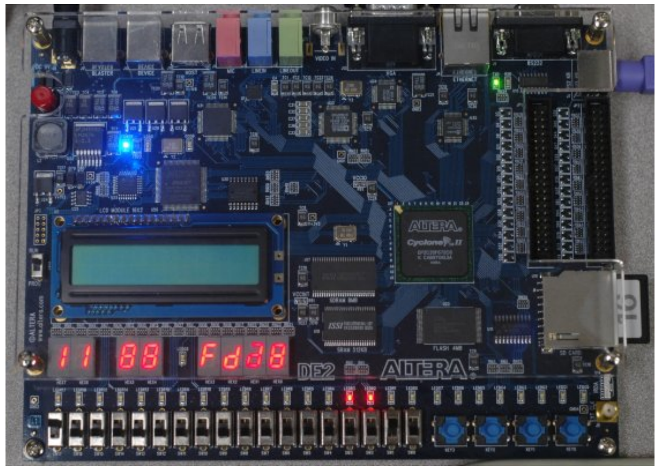
- Fills the gap between Hardware and Software
- Individual logic gates can programed

<ins>Advantages:</ins>

- Higher performance than software
- More flexible than ASIC hardware

<ins>Disadvantages:</ins>

- Programing is not trivial
- Labor intensive development

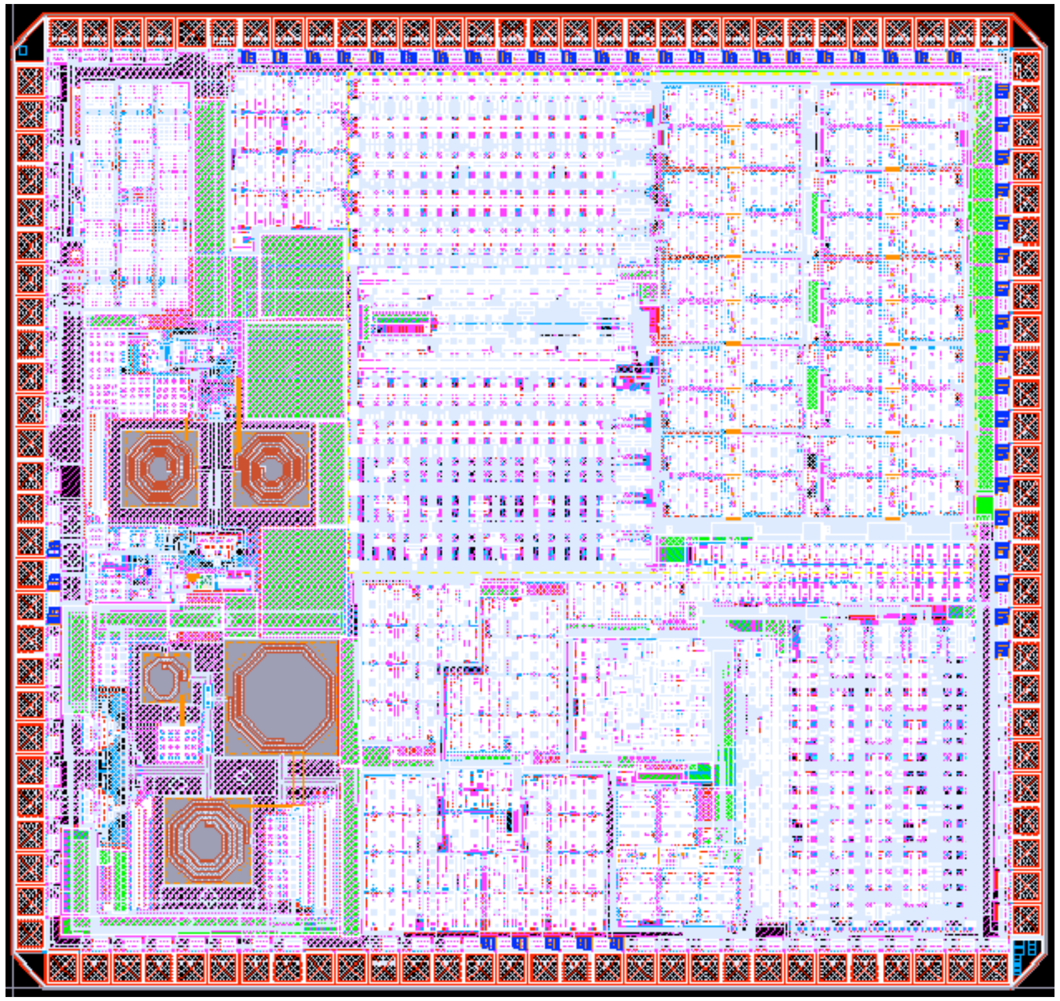


DELL
]FOR I = 0 TO 39 : COLOR = I : HLIN 0,I AT I : VLIN 0,39 AT I : NEXT I

Reconfigurable computing approach

1. You have a program / algorithm that you need to run in hardware
2. You develop that program using logic gates
3. You load these logic gates into a special piece of hardware
4. If you are unhappy with the end results, you can always modify your circuit and reload it into the same board

Hardware description languages



We can model / simulate a piece of hardware before it is created physically!

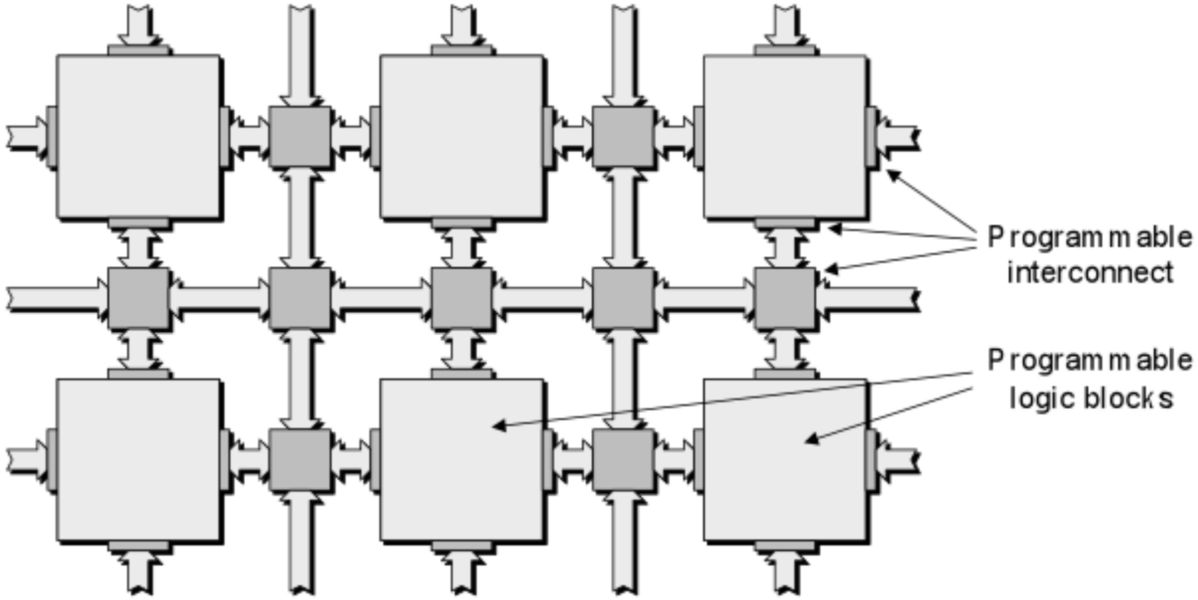
- Unpractical to manually draw each gate in a circuit
- A hardware description language (HDL) is any language which describes the circuit's operation
- Verilog, VHDL, SystemC, Matlab Simulink, LabView, MyHDL

Defining characteristics:

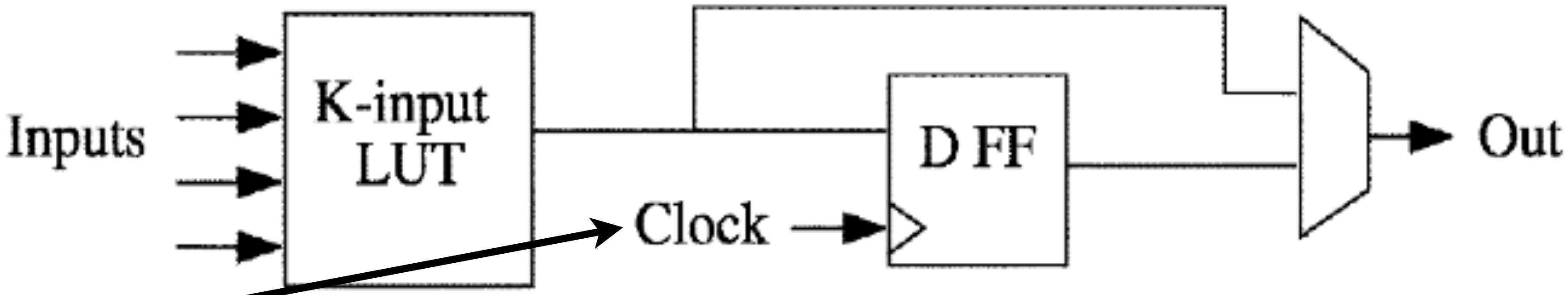
- Explicit notion of time (e.g. Wait 2 ms before sending signal)
- Notations for expressing concurrency

Reconfigurable devices

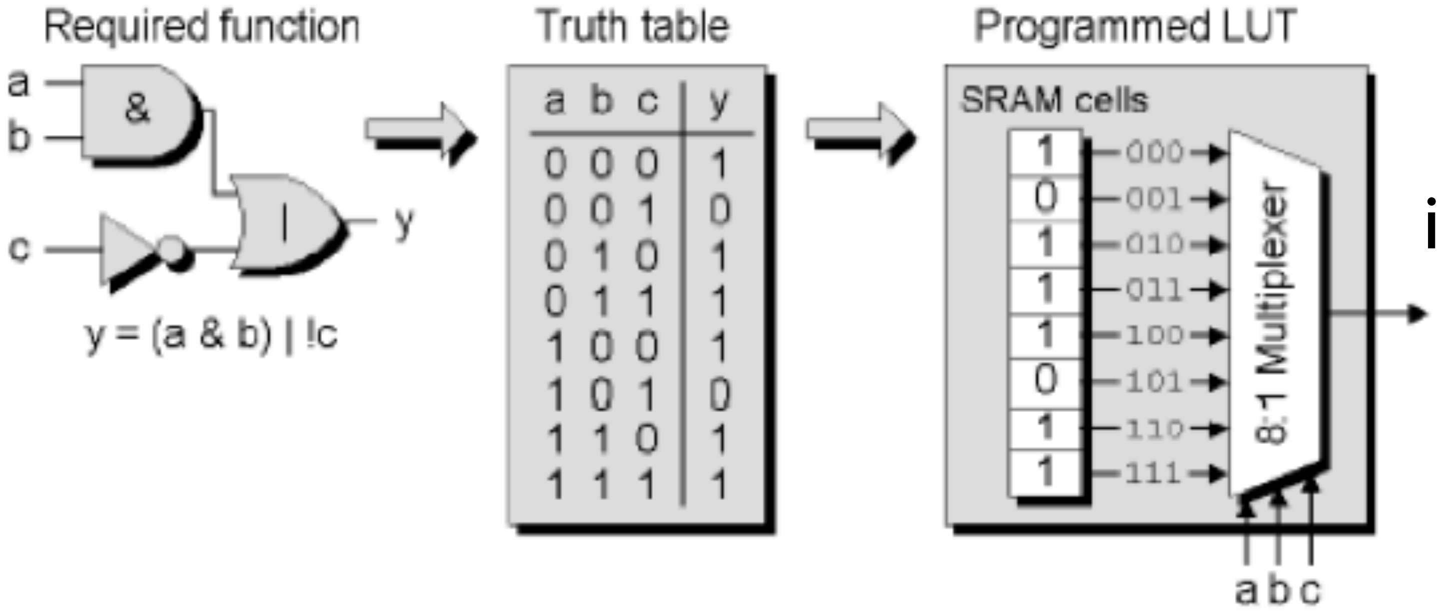
- In a reconfigurable device you configure the behavior of each logic block
- An FPGA is an example of a reconfigurable device
- FPGAs will be our VHDL prototype platform
- The logic blocks are connected by a set of routing resources that are also programmable
- Custom logic circuits can be mapped to the reconfigurable fabric



Block logic element



Everything is clocked!

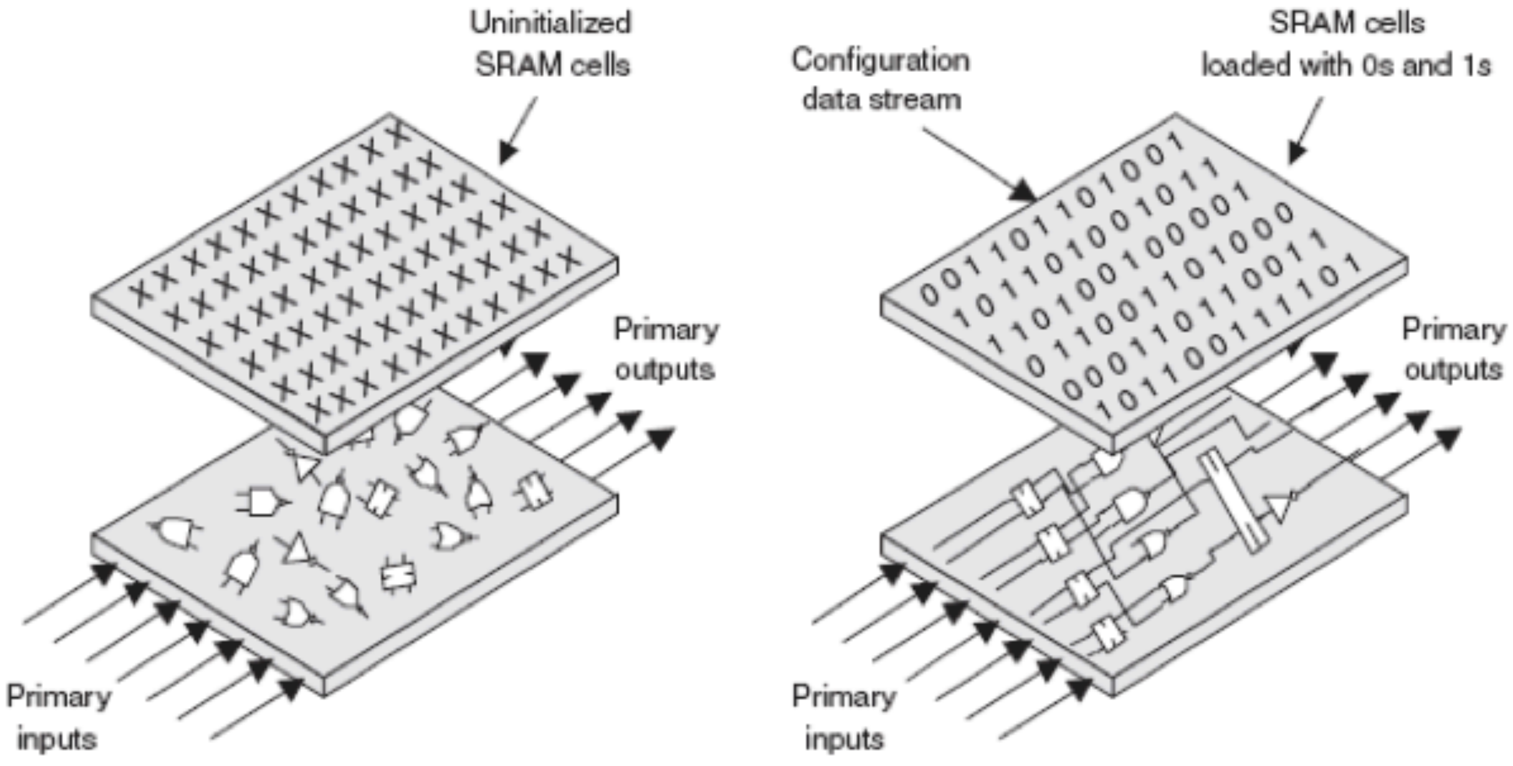


a	b	c	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

LUT is stored inside a SRAM cell

- Designs need to be decomposed and mapped to logic blocks
- FPGAs might contain non-reconfigurable elements that interface to the logic blocks

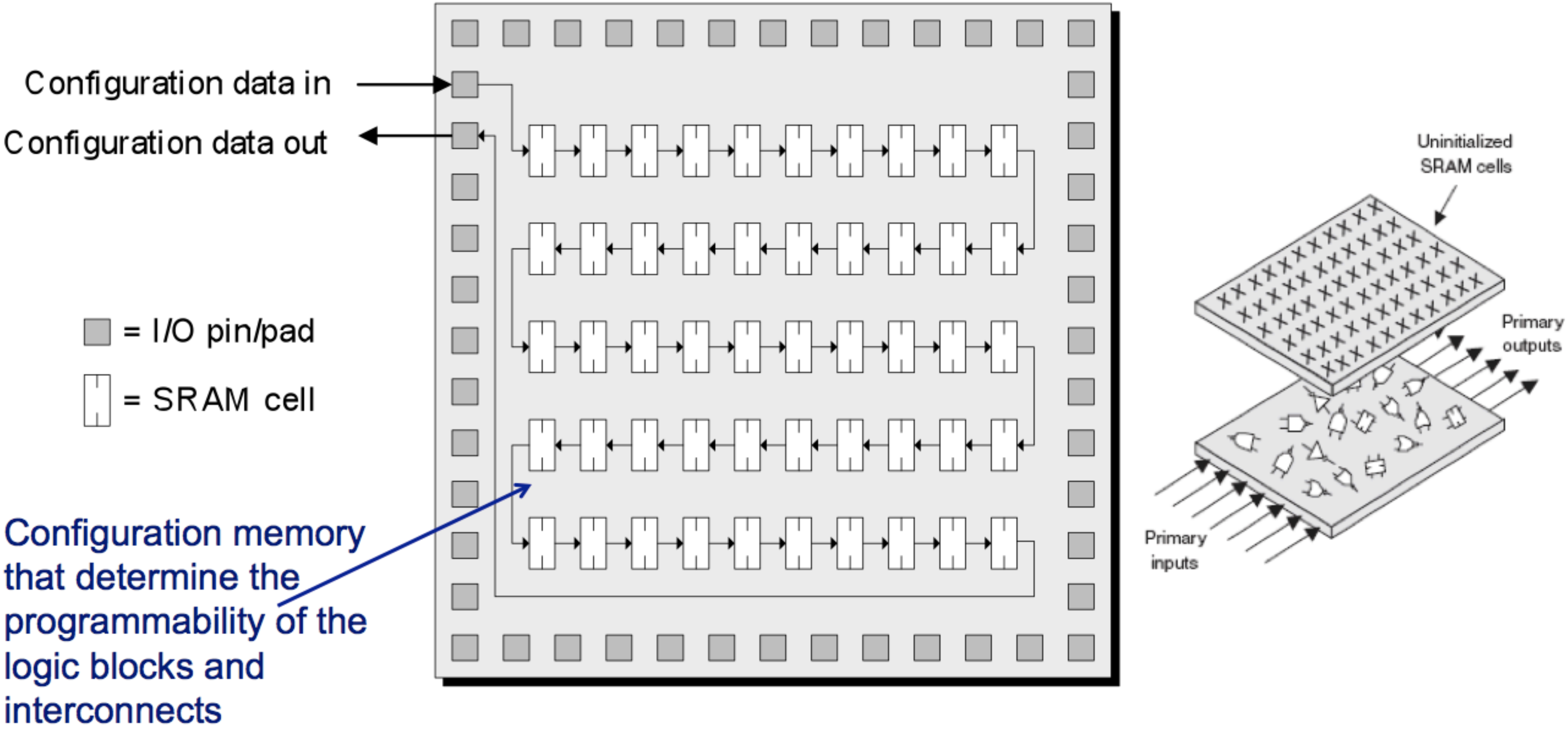
Configuring FPGAs



(a) Unconfigured (b) Configured

- FPGAs can be dynamically reprogrammed before or during runtime
- Slow to reprogram... order of seconds!
- Full or partial reconfiguration

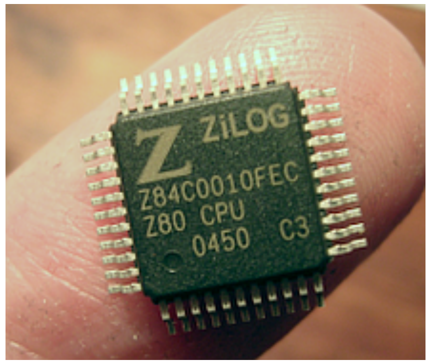
Why so slow to configure?

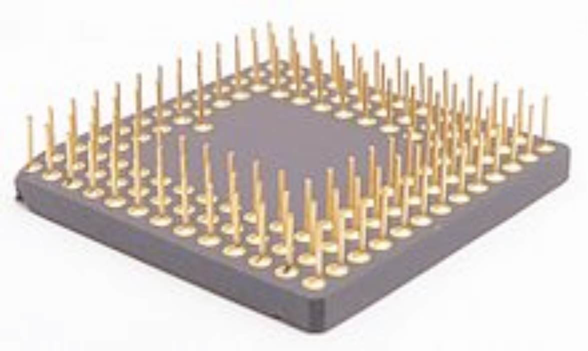


Configuration memory that determine the programmability of the logic blocks and interconnects

How does an FPGA look like?

There are several packaging types



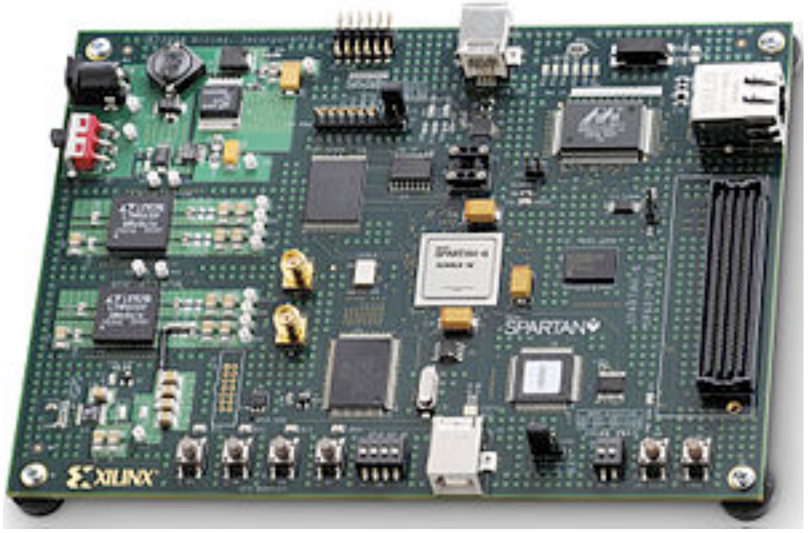


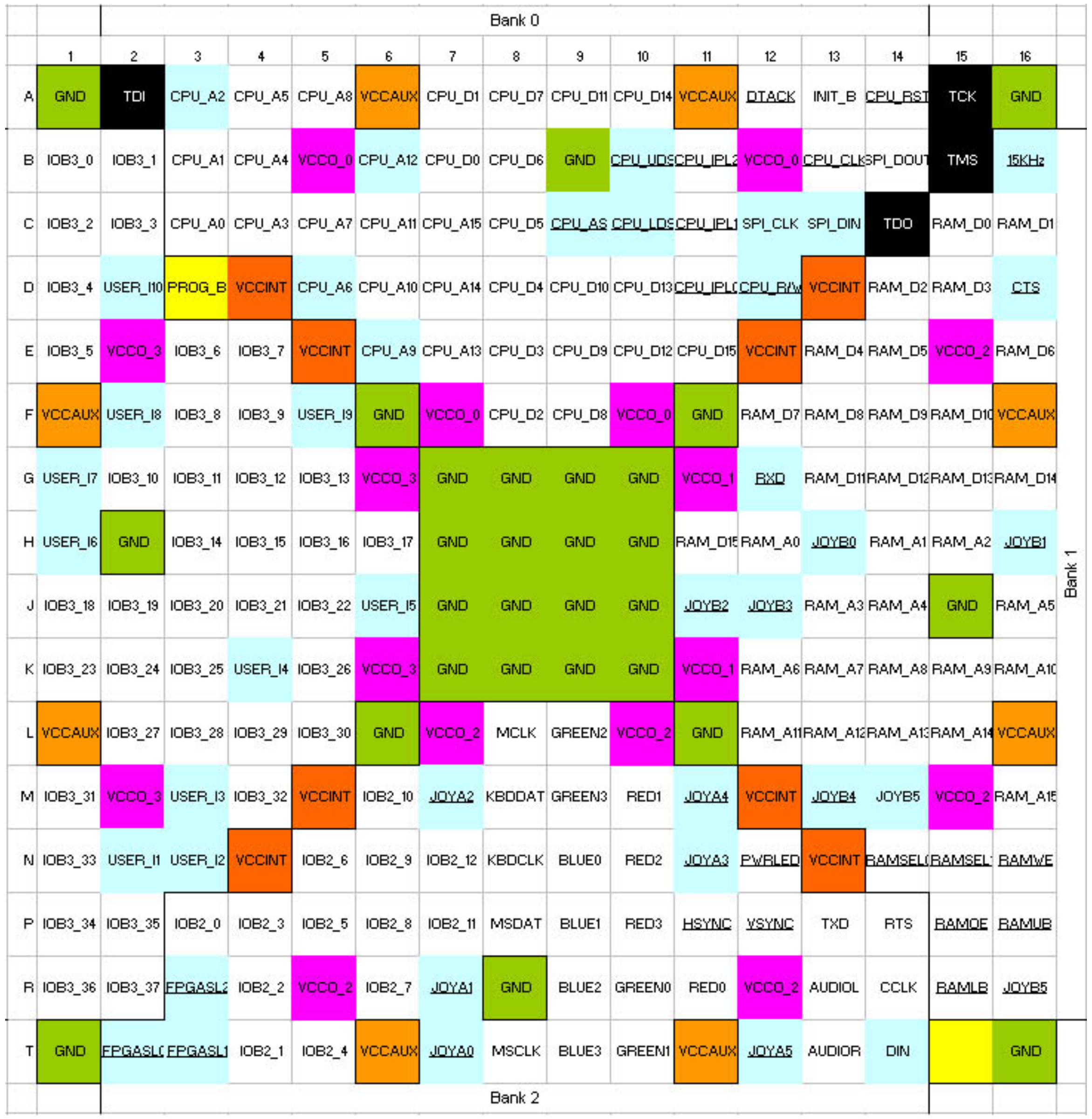
Type	Dual Inline Package DIP (70's)	Quad Flat Package QFP (80's)	Pin Grid Array PGA (90's)	Ball Grid Array BGA (00's)
+	- Easy to solder, handle and replace - Extremely mature technology (cheap)	- More available I/O pins than DIP	- More available I/O pins than QFP - Often mounted with through hole methods	- High density - Good heat conduction - Low inductance
-	- Low pin density - Signal propagates "slowly" through pins	- No socketing or hole mounting (only soldering)	- Long leads means loss of signal integrity	- Expensive testing equipment - Unreliable test sockets

FPGAs are mostly BGA packages

FPGA are chips with lots of I/O

With prototype boards we can fully utilize those I/O ports



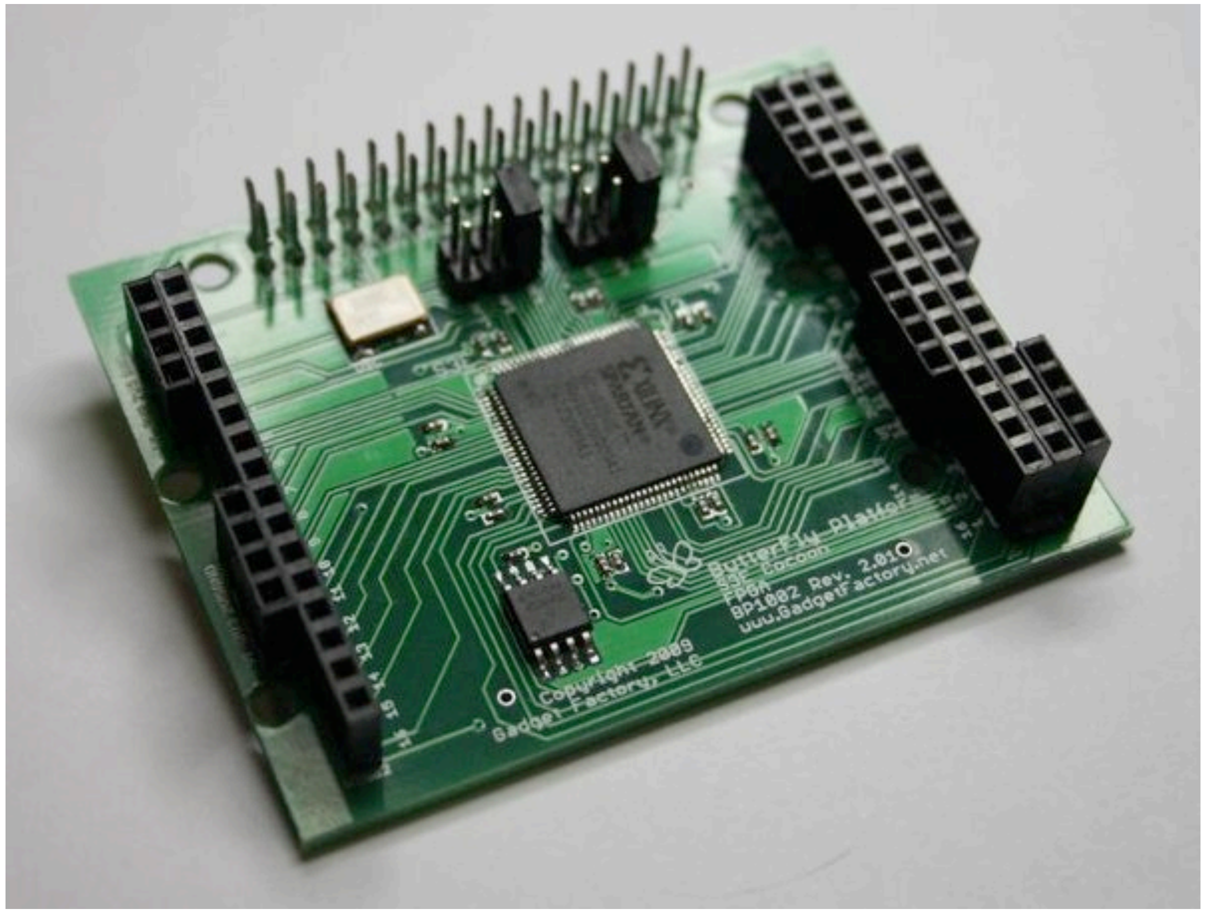


		Bank 0															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A	GND	TDI	CPU_A2	CPU_A5	CPU_A8	VCCAUX	CPU_D1	CPU_D7	CPU_D11	CPU_D14	VCCAUX	DTACK	INIT_B	CPU_RST	TCK	GND	
B	IOB3_0	IOB3_1	CPU_A1	CPU_A4	VCCO_0	CPU_A12	CPU_D0	CPU_D6	GND	CPU_UDS	CPU_IPL2	VCCO_0	CPU_CLK	SPI_DOUT	TMS	15KHz	Bank 1
C	IOB3_2	IOB3_3	CPU_A0	CPU_A3	CPU_A7	CPU_A11	CPU_A15	CPU_D5	CPU_AS	CPU_LDS	CPU_IPL1	SPI_CLK	SPI_DIN	TDO	RAM_D0	RAM_D1	
D	IOB3_4	USER_I10	PROG_B	VCCINT	CPU_A6	CPU_A10	CPU_A14	CPU_D4	CPU_D10	CPU_D13	CPU_IPL0	CPU_R/W	VCCINT	RAM_D2	RAM_D3	CTS	
E	IOB3_5	VCCO_3	IOB3_6	IOB3_7	VCCINT	CPU_A9	CPU_A13	CPU_D3	CPU_D9	CPU_D12	CPU_D15	VCCINT	RAM_D4	RAM_D5	VCCO_2	RAM_D6	
F	VCCAUX	USER_I8	IOB3_8	IOB3_9	USER_I9	GND	VCCO_0	CPU_D2	CPU_D8	VCCO_0	GND	RAM_D7	RAM_D8	RAM_D9	RAM_D10	VCCAUX	
G	USER_I7	IOB3_10	IOB3_11	IOB3_12	IOB3_13	VCCO_3	GND	GND	GND	GND	VCCO_1	RXD	RAM_D11	RAM_D12	RAM_D13	RAM_D14	
H	USER_I6	GND	IOB3_14	IOB3_15	IOB3_16	IOB3_17	GND	GND	GND	GND	RAM_D15	RAM_A0	JOYB0	RAM_A1	RAM_A2	JOYB1	
J	IOB3_18	IOB3_19	IOB3_20	IOB3_21	IOB3_22	USER_I5	GND	GND	GND	GND	JOYB2	JOYB3	RAM_A3	RAM_A4	GND	RAM_A5	
K	IOB3_23	IOB3_24	IOB3_25	USER_I4	IOB3_26	VCCO_3	GND	GND	GND	GND	VCCO_1	RAM_A6	RAM_A7	RAM_A8	RAM_A9	RAM_A10	
L	VCCAUX	IOB3_27	IOB3_28	IOB3_29	IOB3_30	GND	VCCO_2	MCLK	GREEN2	VCCO_2	GND	RAM_A11	RAM_A12	RAM_A13	RAM_A14	VCCAUX	
M	IOB3_31	VCCO_3	USER_I3	IOB3_32	VCCINT	IOB2_10	JOYA2	KBDDAT	GREEN3	RED1	JOYA4	VCCINT	JOYB4	JOYB5	VCCO_2	RAM_A15	
N	IOB3_33	USER_I1	USER_I2	VCCINT	IOB2_6	IOB2_9	IOB2_12	KBDCLK	BLUE0	RED2	JOYA3	PWRLED	VCCINT	RAMSEL0	RAMSEL1	RAMWE	
P	IOB3_34	IOB3_35	IOB2_0	IOB2_3	IOB2_5	IOB2_8	IOB2_11	MSDAT	BLUE1	RED3	HSYNC	VSYNC	TXD	RTS	RAMOE	RAMUB	
R	IOB3_36	IOB3_37	FPGASL2	IOB2_2	VCCO_2	IOB2_7	JOYA1	GND	BLUE2	GREEN0	RED0	VCCO_2	AUDIOL	CCLK	RAMLB	JOYB5	
T	GND	FPGASL0	FPGASL1	IOB2_1	IOB2_4	VCCAUX	JOYA0	MSCLK	BLUE3	GREEN1	VCCAUX	JOYA5	AUDIOR	DIN		GND	
		Bank 2															

Each solder point in a BGA has a well defined I/O functionality

What circuits can you implement on reconfigurable hardware?

... Besides size constraints you can deploy pretty much any circuit that has been done with standard logic gates.

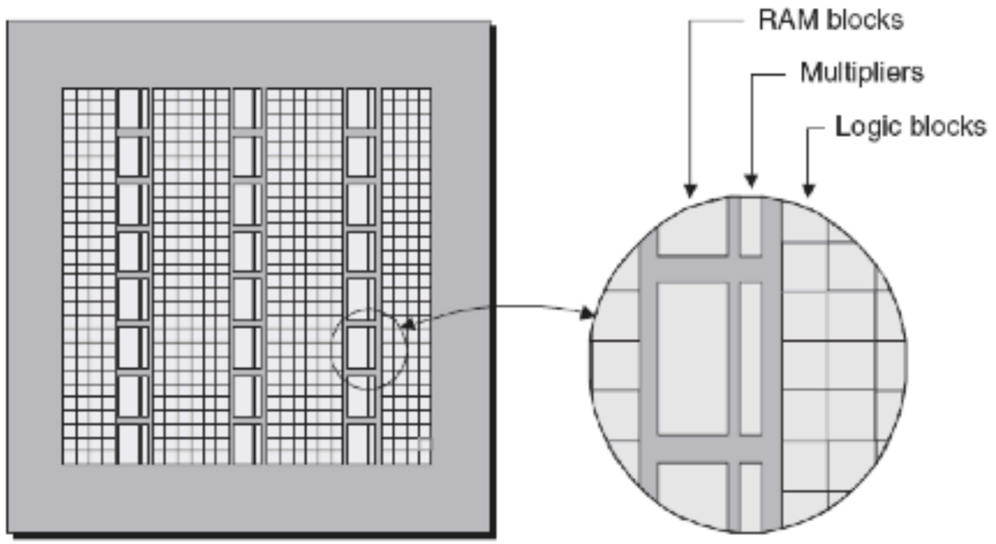


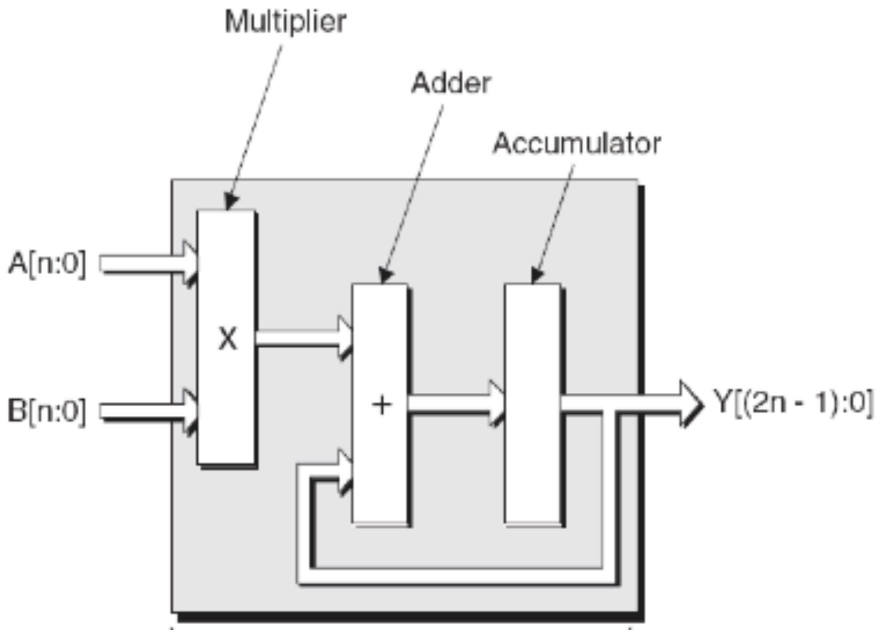
- These folks implemented an arduino compatible microprocessor in an FPGA
- Why? You can add extra-functionality (such as extra I/O pins) to an existing micro-processor.

http://gadgetforge.gadgetfactory.net/gf/project/wiringide/

Embedded RAM and multipliers

- Problem: "Expensive" to implement memory with configurable logic blocks
- Solution: Add hard chunks of RAM blocks.
    - Position/size vary depending on the FPGA device.
- Problem: Multipliers are inherently slow if cascaded
- Solution: Add hard-wired multiplier blocks





Higher performance than software

... is obtained in many ways. For example, with spatial based computing.

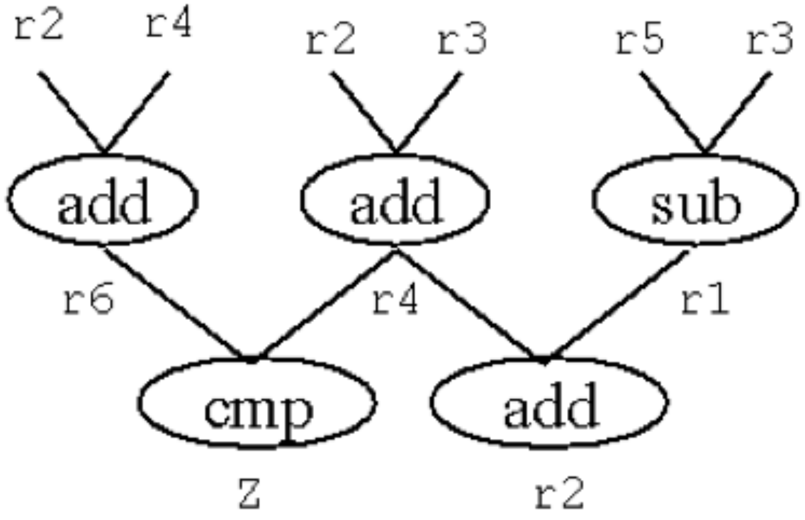
Goal: I have an algorithm/program I want to run really fast.

1) I try to extract parallelism (or concurrency) from the instructions as much as possible

2) Then, I implement the algorithm as hardware.

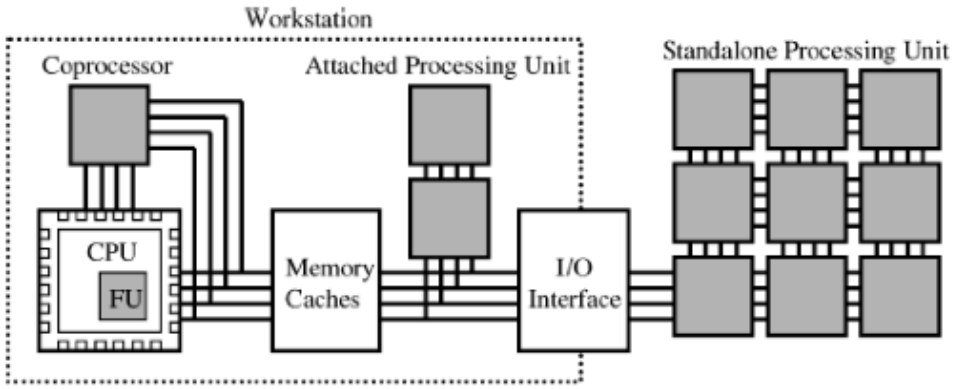
```
add    r2,r4,r6
add    r2,r3,r4
sub    r5,r3,r1
add    r4,r1,r2
cmp    r6,r4
```

Spacial based execution



More flexible than ASIC

- Low/med volume IC production
- Early prototyping and logic emulation
- Accelerating algorithms in reconfigurable computing environments



[Compton'02]

1. Reconfigurable functional units within a host processor (custom instructions)
2. Reconfigurable units used as coprocessors
3. Reconfigurable units that are accessed through external I/O or a network

- Legacy Computing

Why reconfigurable computing is more relevant these days?

- There is a demand for high-performance, data processing, computation. E.g. Gene sequencing and financial market analysis
- *Why are general-purpose processors not meeting the demand?*

1. Single thread performance is no longer improving (individual core frequencies do not increase due to thermal problems)
2. Consume large amount of power

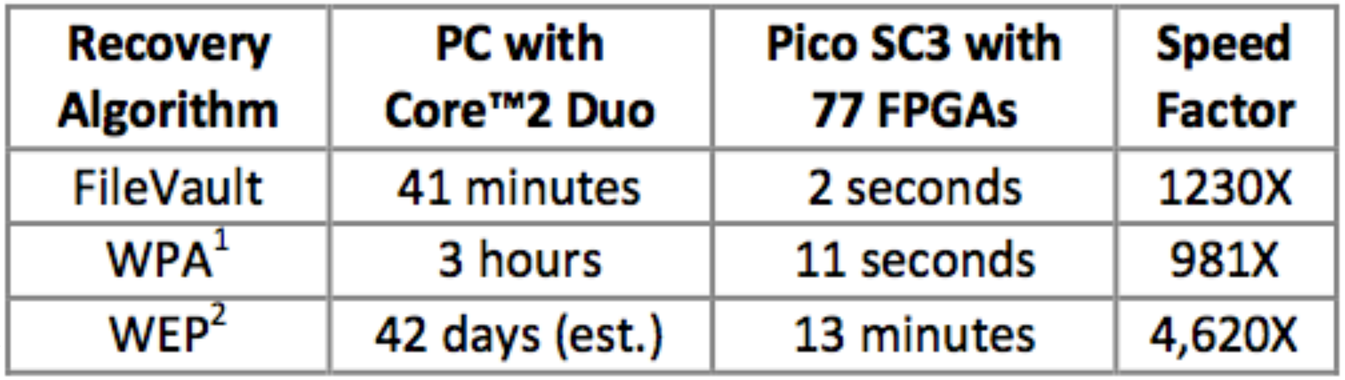
- *Why reconfigurable architectures could meet the computational demand?*
  - Can process large streams of data directly in hardware
  - Inexpensive and consume little power

Tangible examples of reconfigurable computing applications

Fast password recovery

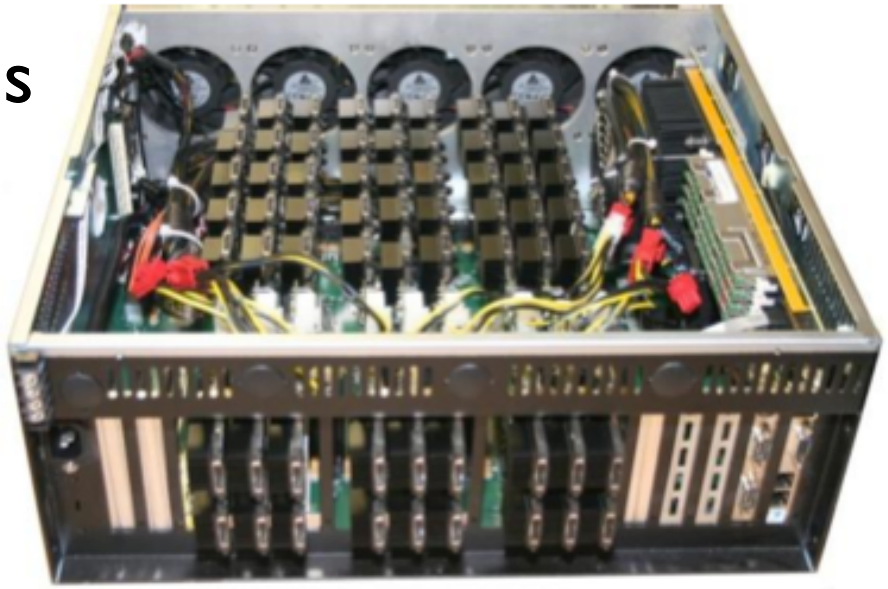
"Using FPGA Clusters for Fast Password Recovery" - Pico Computing, Inc. (www.picocomputing.com) - 2009 white-paper

- They take password cracking algorithms that crack SHA-1,WPA and WEP.
- Convert code into logic gates and optimize it
- Deploy the same code across a grid of reconfigurable hardware
- Load protected data in memory and start process



Recovery Algorithm	PC with Core™2 Duo	Pico SC3 with 77 FPGAs	Speed Factor
FileVault	41 minutes	2 seconds	1230X
WPA[1]	3 hours	11 seconds	981X
WEP[2]	42 days (est.)	13 minutes	4,620X

40 bit →

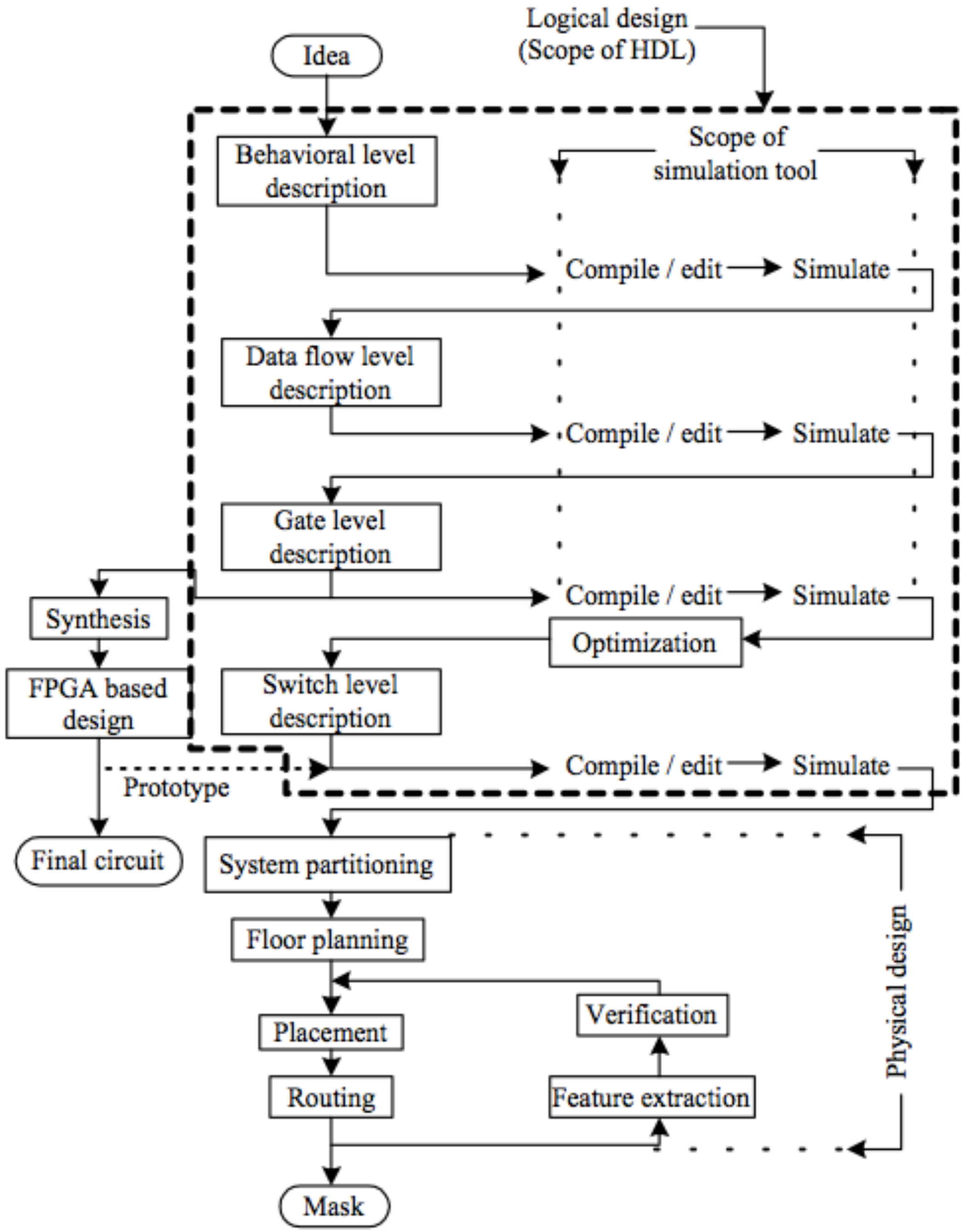


Accurate debug and simulation of ASIC designs

- Behavioral: Entire circuit in terms of functionality (algorithm)
- Dataflow: Breaks circuits into blocks and modules
- Gate: Modules are decomposed into logic gates.

There is a lot of overlap between the design flow for an ASIC and an FPGA.

To some extent all that can be done in ASIC can be done on an FPGA.



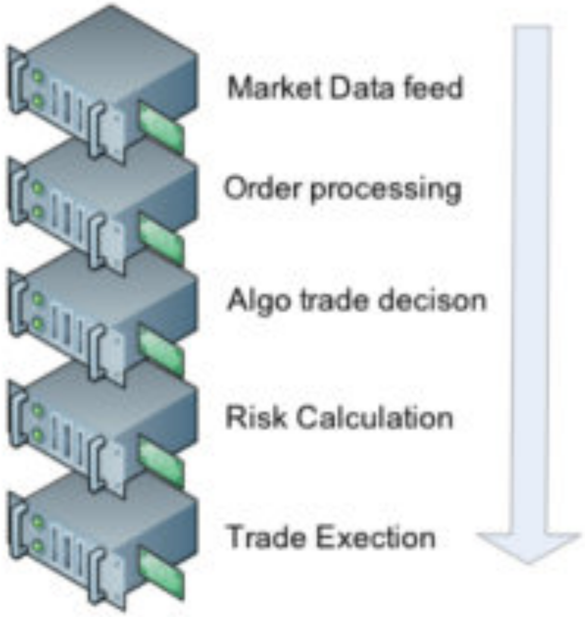
High frequency trading (HFT)

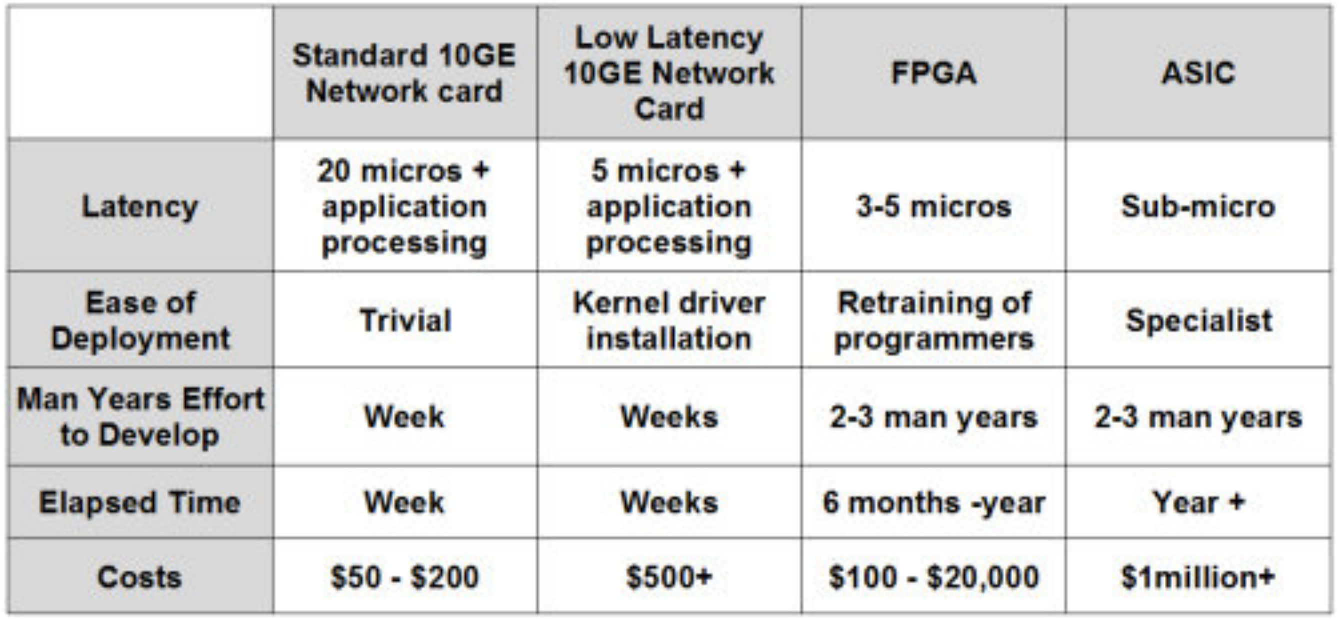
- HFT is an investment technique that gets in and out of positions very quickly, while making tiny amounts on each transaction (≈ few cents)

- If you trade a lot, and faster than your competitors, all those cents add up

- For example, we can reduce a 30µs risk calculation on a PC, into a 3µs calculation on a FPGA

- FPGAs are embedded into network cards to reduce any latency



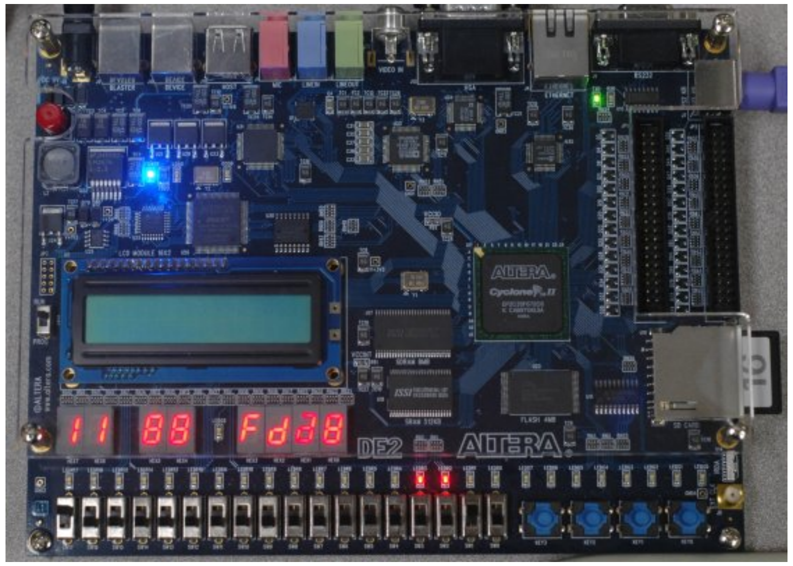


	Standard 10GE Network card	Low Latency 10GE Network Card	FPGA	ASIC
Latency	20 micros + application processing	5 micros + application processing	3-5 micros	Sub-micro
Ease of Deployment	Trivial	Kernel driver installation	Retraining of programmers	Specialist
Man Years Effort to Develop	Week	Weeks	2-3 man years	2-3 man years
Elapsed Time	Week	Weeks	6 months -year	Year +
Costs	$50 - $200	$500+	$100 - $20,000	$1million+

Re-creating an1980's Apple II+

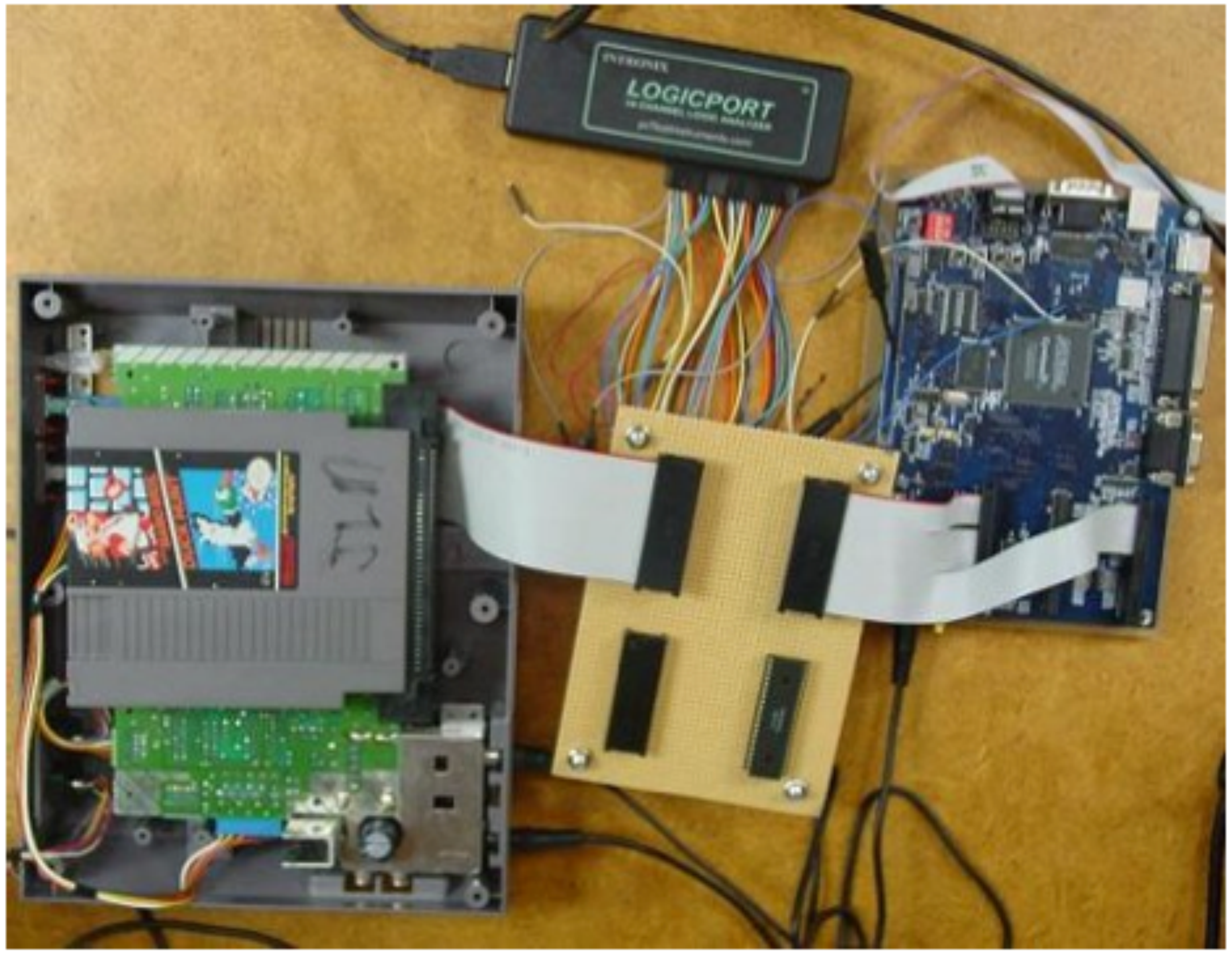
http://www.cs.columbia.edu/~sedwards/apple2fpga/

The point, aside from entertainment, was to illustrate the power (or rather, low power) of modern FPGAs. (...) What made Steve Jobs his first million can now be a class project for my embedded systems class.



Re-create game consoles (NES)

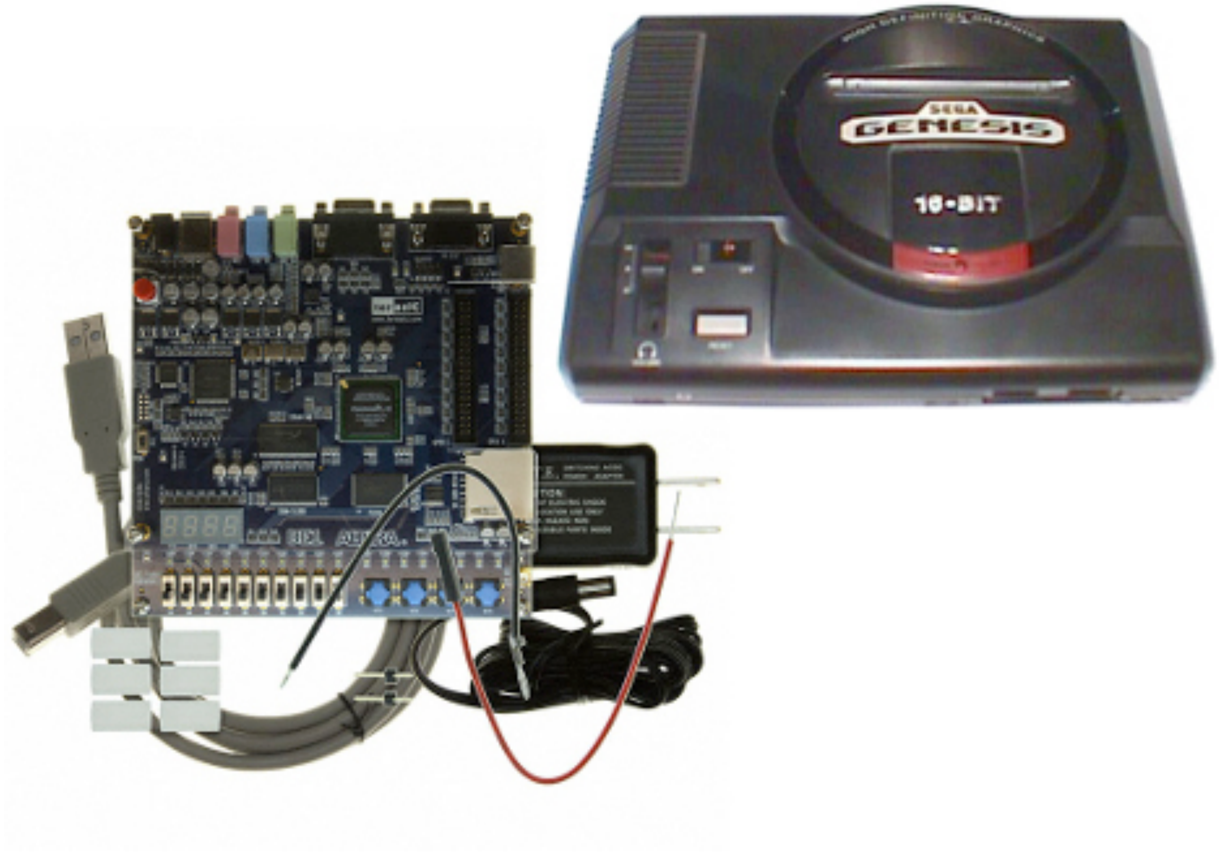
http://cegt201.bradley.edu/projgrad/proj2006/



- Dan Leach's Master's project at Bradley University
- NES is a Legacy System, still used but no longer manufactured: if one piece breaks, you are in trouble
- 1 year project with lots of detective work!
- Source code in VHDL is provided

Re-create game consoles (Genesis)

http://hackaday.com/2010/07/16/sega-genesis-cloned-with-an-fpga/



- This fellow managed to re-implement a SEGA Genesis with an old FGPA board
- The onboard push buttons are used as the controller with VGA for the display
- Unfortunately source code is not provided
- For other game consoles and more info, check: http://www.fpgaarcade.com/

What's the point of resurrecting legacy systems?

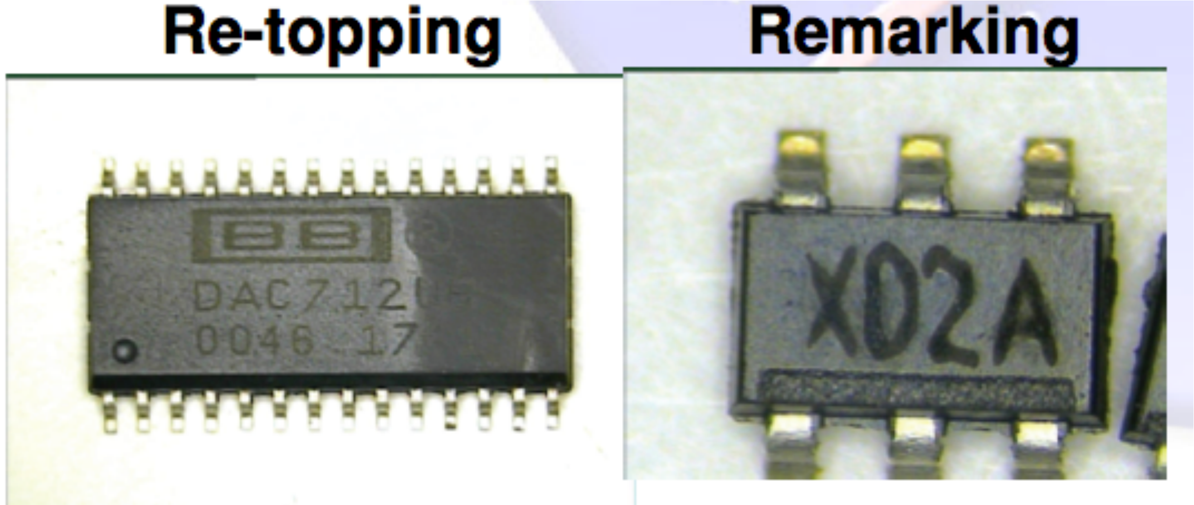
With many NES and Genesis emulators available, what is the advantage of re-creating these systems in hardware?

In other words why would you want an FPGA implementation of a legacy system?

- Replace aging, malfunctioning hardware
- Reduce power consumption by replacing a system
- Reduce the overall cost of a larger, older system
- Reduce component size and thus system size for use in smaller areas
- <u>Safe-guard against counterfeit parts!</u>

Counterfeit parts

- “Counterfeit Electronic Parts”, White paper @ Trilateral Safety and Mission Assurance Conference (2008)
- http://www.hq.nasa.gov/office/codeq/trismac/apr08/day2/hughitt_NASA_HQ.pdf



Shoddy counterfeits

Sophisticated counterfeiting industry

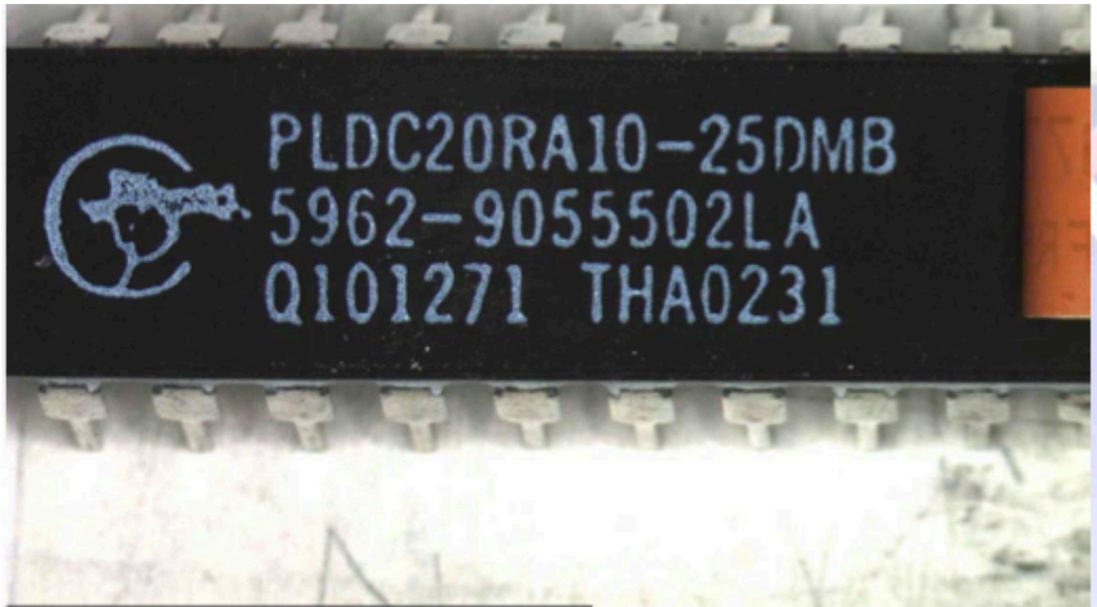


Figure 1: Photo of suspect part.

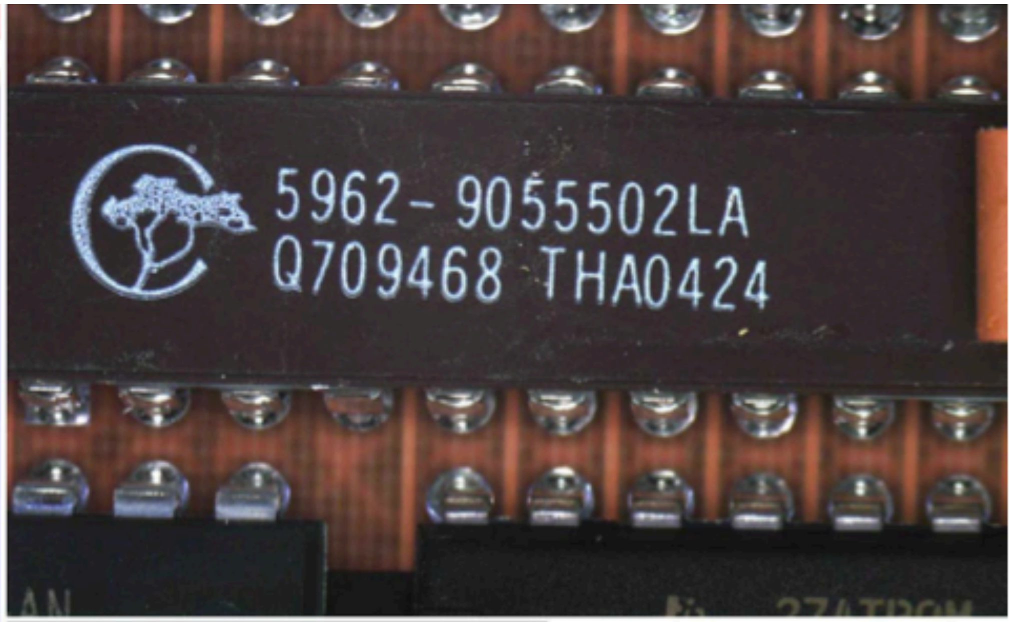
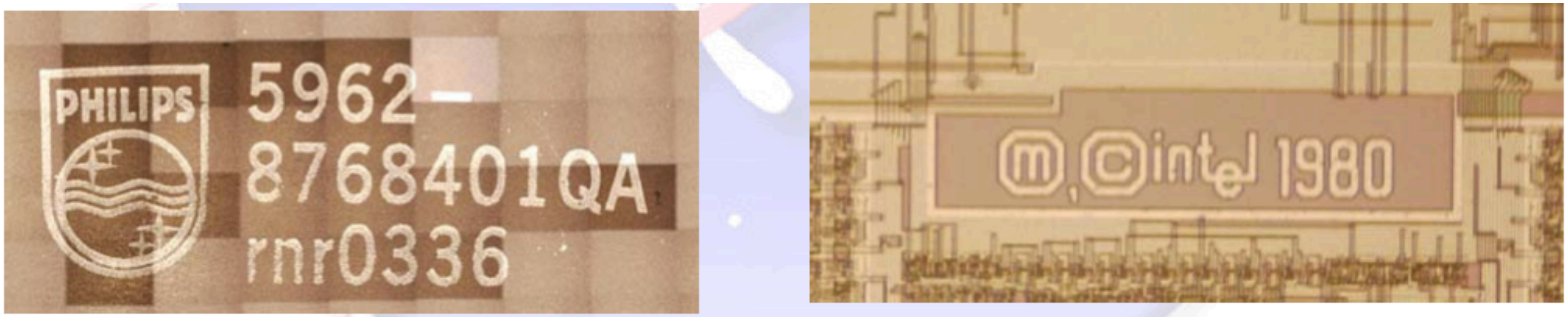


Figure 2: Photo of Known Good Part



The packaging mark (outside label) does not match internal die markings!

Hardware acceleration with GPUs

FPGAs vs GPUs

FPGA

- Custom chip which eliminates inefficiencies of Von Neumann execution models.

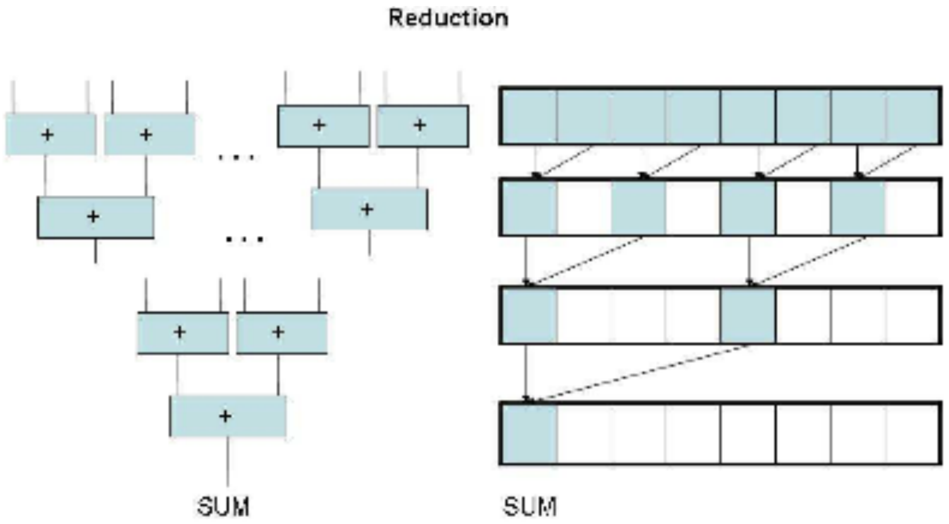
- All programming languages are hardware based.

GPU

- HW board with high memory bandwidth and allows thousands of hardware threads.

- Flexible and "easy" to program with high level languages which abstract hardware.

- NVIDIA's CUDA, AMD's CAL are new language development APIs.



- Both extremely parallel architectures whose algorithmic speedup is based on reduction steps.

- **(Left)** FPGA with a cascade of adders of depth of log(N).

- **(Right)** Number of working threads reduces in half in each iteration of a GPU implementation of a reduce which requires log(N) iterations.

Pro’s and con’s

	FPGA	GPU
Development time	Long	Short
Synchronize tasks	Easy. We can insert hardware barriers.	Hard. API only allows synchronization of all threads.
Logic operations	Bitwise operations: add, shift & permute done in 1 cycle. (DES algorithm)	Native support for floating point operations!
Communication Overhead	Reconfiguration takes considerable amount of time.	Uses off-chip device memory. PCI-Express bus.