Digital Systems for Telecommunications

Digital System design flow

Programmable Logic Devices: FPGA

DSP ?

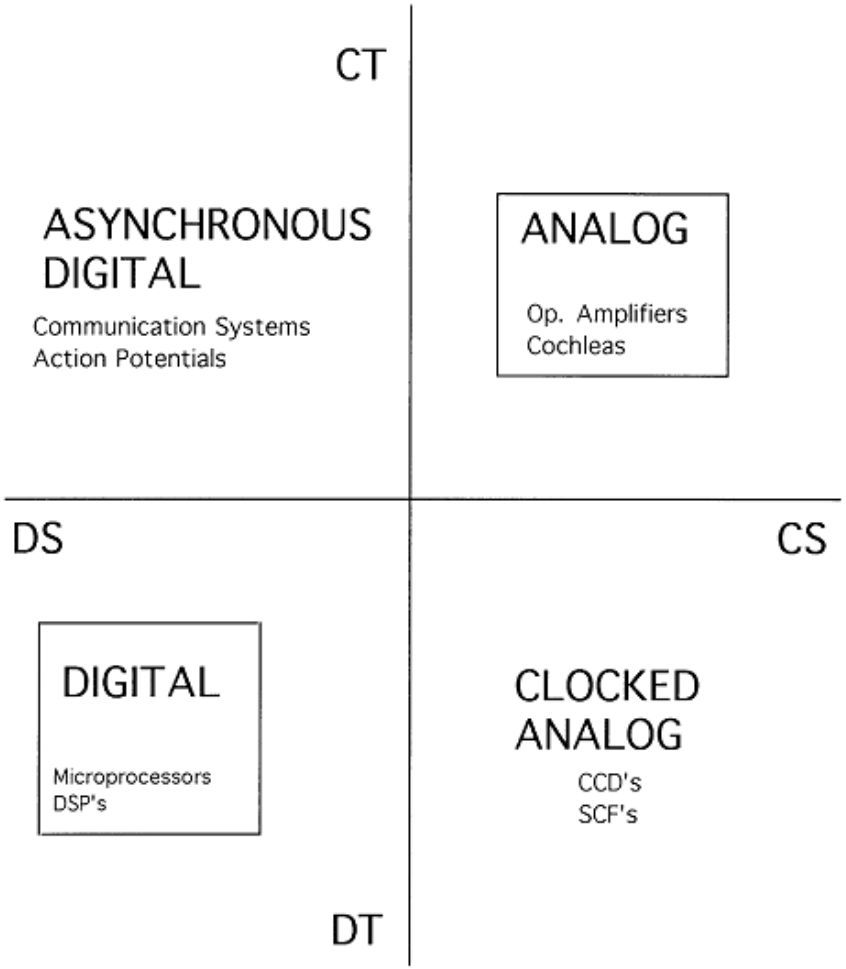
- ***Signal*** – a ***detectable physical quantity*** or impulse (as a voltage, current, or magnetic field strength) by which messages or ***information*** can be transmitted (Webster Dictionary)

- Signals generated via physical phenomenon are analog in that
  - Their amplitudes are defined over the range of real/complex numbers
  - Their domains are continuous in time or space.
  - Processing analog signal requires dedicated, special hardware.

- Definition: ***DSP*** – Digital Signal Processing/Processor
  - It refers to:
    - Theoretical signal processing by digital means
    - Specialized hardware (processor) that can process signals in real-time

Digital Systems

- **The four types of systems**
  - Systems that operate with **continuous** or **discrete signals** (CS or DS) and in **continuous** or **discrete time** (CT or DT).
  - Systems that are continuous in both the signal and time domains (CSCT) and systems that are discrete in both the signal and time domains (DSDT) are usually referred as “analog” or “digital” systems, respectively



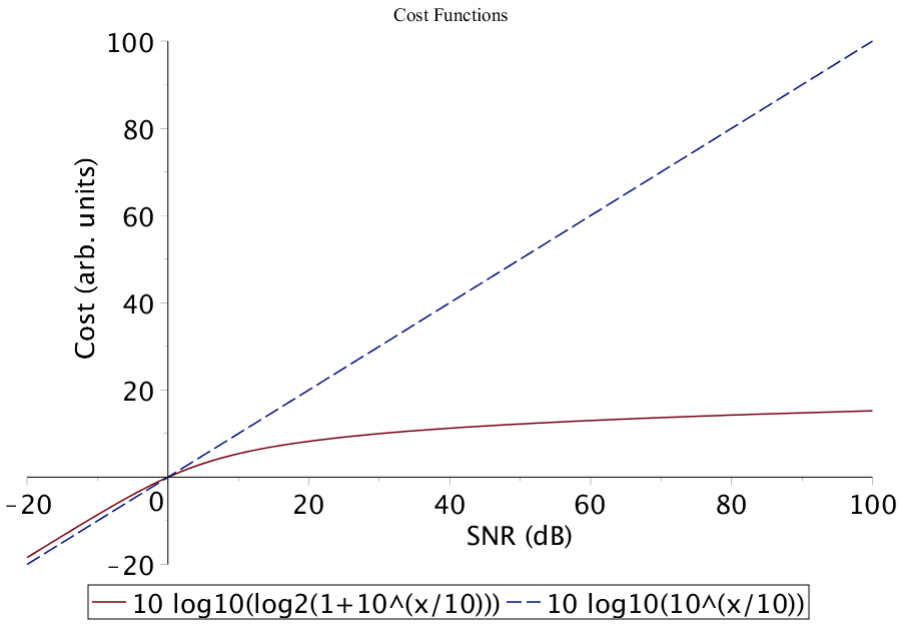
SCF stands for switched capacitor filter; CCD stands for charge coupled device.

Digital vs Analog

- **Digital vs Analog**
  - Everything is performed in the digital way today!
  - SNR VS Cost (Area, Power)

Analog
Cost(A) ∝ SNR
Cost(P) ∝ SNR

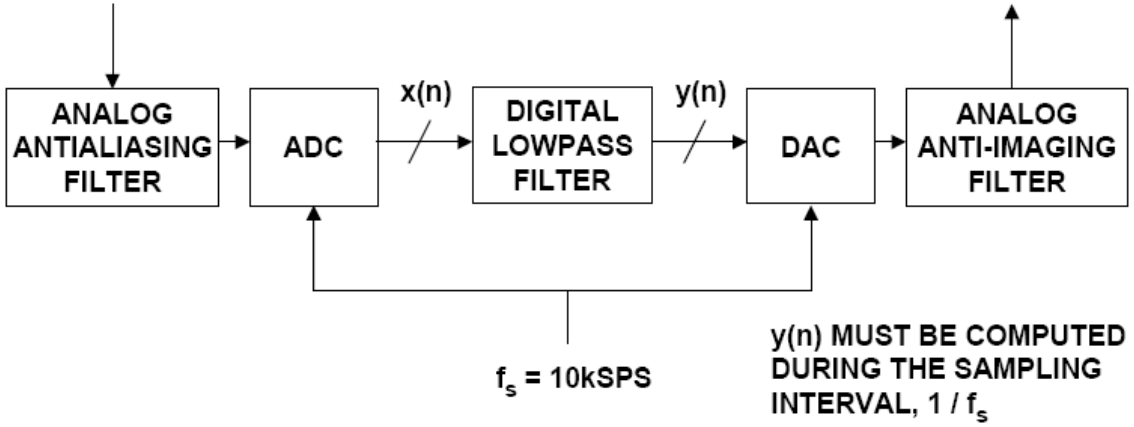
Digital
Cost(A) ∝ Bits ∝ Log2(1+SNR)
Cost(P) ∝ Bits ∝ Log2(1+SNR)



- **System Design**
  - We deal with (very) complex systems, not a single device!
- Usually in the form of **Embedded Systems**

Filter implementation example: digital

- It is assumed that an ADC/DAC combination is available with sufficient sampling frequency, resolution, and dynamic range to accurately process the signal.
- The DSP is fast enough to complete all its calculations within the sampling interval, 1/fs.
- Analog filters are still required at the ADC input and DAC output for antialiasing and anti-imaging, but the performance demands are not as great.



Filter: Analog vs. Digital Implementations

Analog

- Cons:
  - Approximate Filter Coefficients
    - Only standard components available
  - Environment Temperature dependent
  - Less accurate (components tolerance…)
  - Suffer from aging
  - Can be used only for designed purpose
- Pros:
  - **Operate in real-time**

Digital (DSP)

- Cons:
  - Real-time operation is dependent on the speed of processor and the complexity of problem at hand.
- Pros:
  - Accurate Filter implementation to desired precision
  - Operation independent on the environment.
  - Linear phase response is possible
  - Flexible (DSP processors can be reprogrammed).

Three key system technologies

- Technology
  - A manner of accomplishing a task, especially using technical processes, methods, or knowledge

- Three key technologies for embedded systems
  - **Processor technology**: The architecture of the computation engine used to implement a system's desired functionality
  - **IC technology**: The manner in which an analog/digital implementation is mapped onto an IC
  - **Design technology**: The manner in which we convert our concept of desired system functionality into an implementation

Processor technology

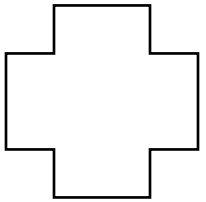
- Processors vary in their customization for the problem at hand

Desired functionality

```
total = 0
for i = 1 to N  loop
   total += M[i]
end loop
```

General-purpose processor

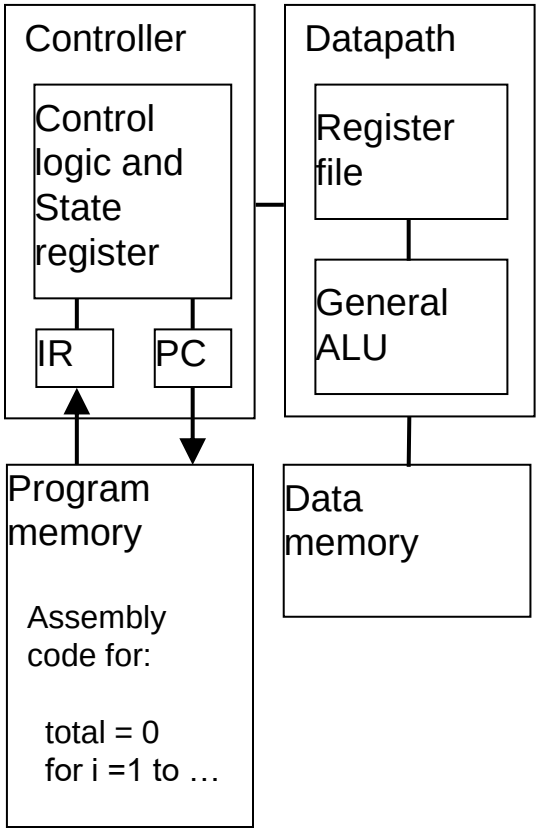
Application-specific processor



Single-purpose processor

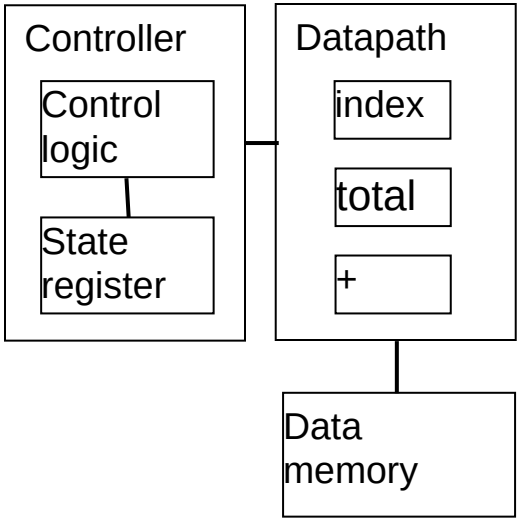
General-purpose processors

- Programmable device used in a variety of applications
  - Also known as “microprocessor”
- Features
  - Program memory
  - General datapath with large register file and general ALU
- User benefits
  - Low time-to-market and NRE costs
  - High flexibility
- “Intel processors” the most well-known, but there are hundreds of others



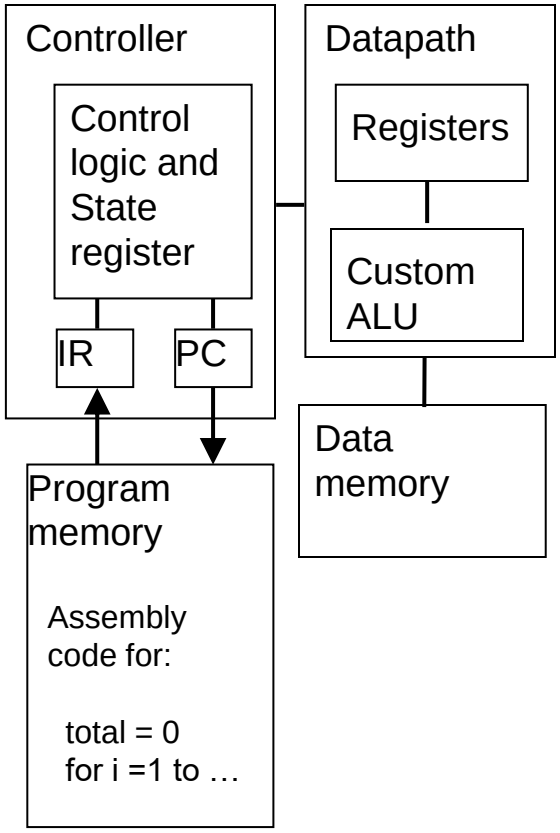
Single-purpose processors

- Digital circuit designed to execute exactly one program
  - a.k.a. coprocessor, accelerator or peripheral
- Features
  - Contains only the components needed to execute a single program
  - No program memory
- Benefits
  - Fast
  - Low power
  - Small size



Application-specific processors

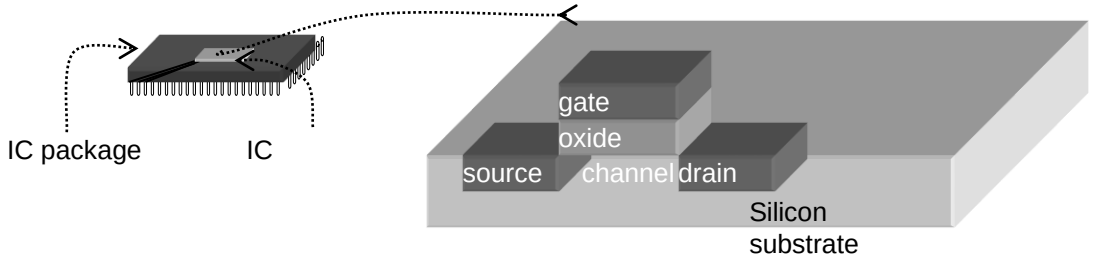
- Programmable processor optimized for a particular class of applications having common characteristics
  - Compromise between general-purpose and single-purpose processors
- Features
  - Program bs Data memory
  - Optimized datapath
  - Special functional units
- Benefits
  - Some flexibility, good performance, size and power



IC technology

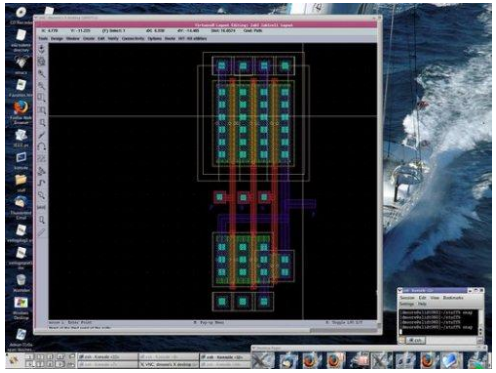
- IC technologies:
  - IC: Integrated circuit, or “chip”
  - IC technologies differ in their customization to a design
  - IC’s consist of numerous layers (perhaps 10 or more); IC technologies differ with respect to who builds each layer and when

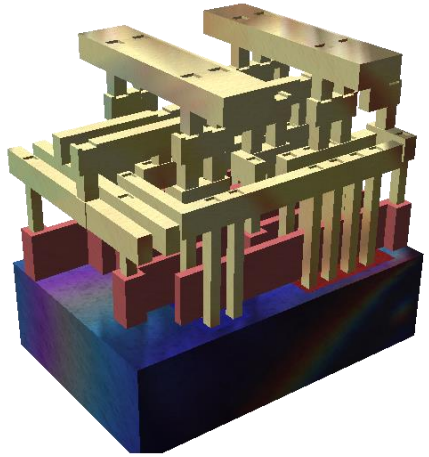
- Three types of IC technologies
  - Full-custom/VLSI
  - Semi-custom ASIC (gate array and standard cell)
  - PLD (Programmable Logic Device)



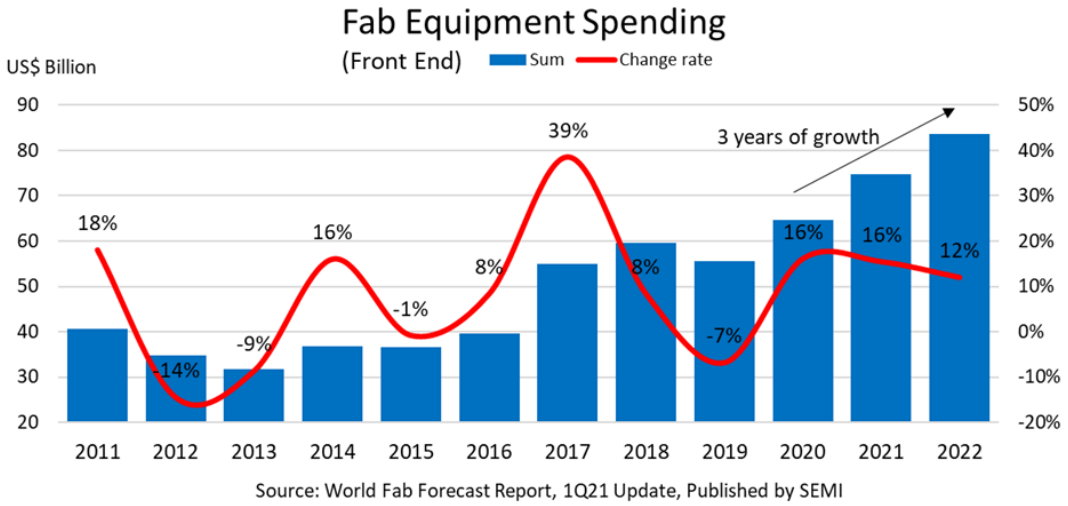
Full-custom/VLSI

- All layers are optimized for an embedded system's particular digital implementation
  - Placing transistors
  - Sizing transistors
  - Routing wires



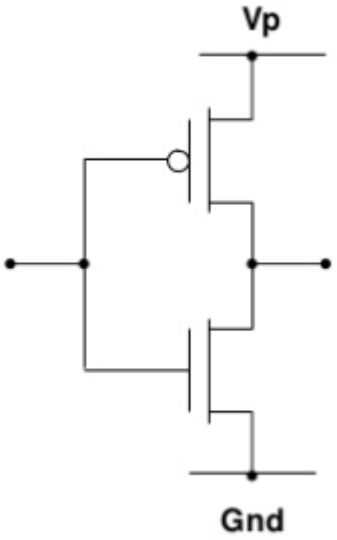


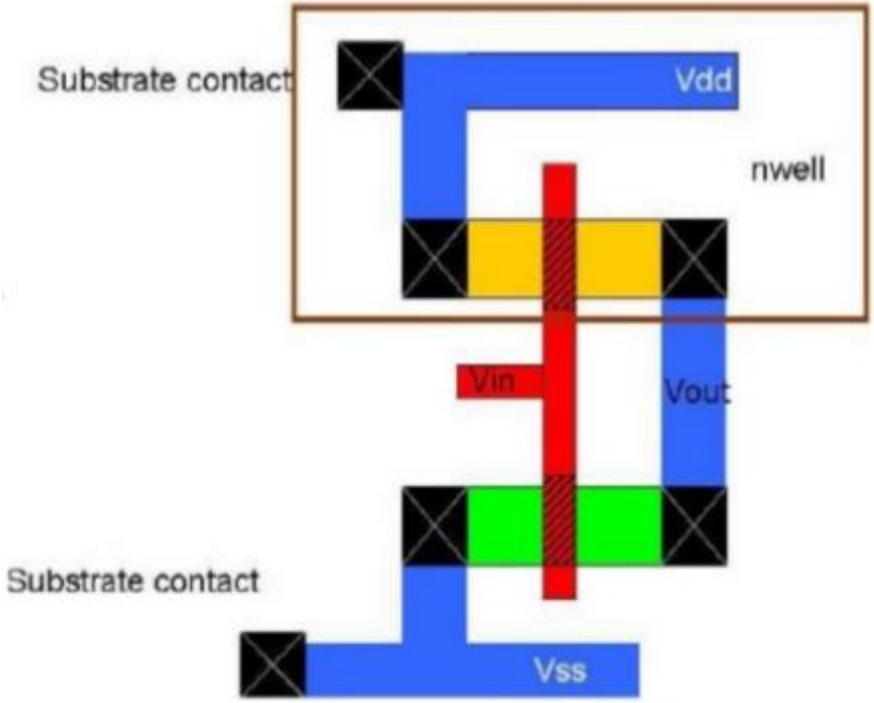
- Benefits
  - Excellent performance, small size, low power
- Drawbacks
  - High NRE cost (e.g., $300k), long time-to-market



Standard cells

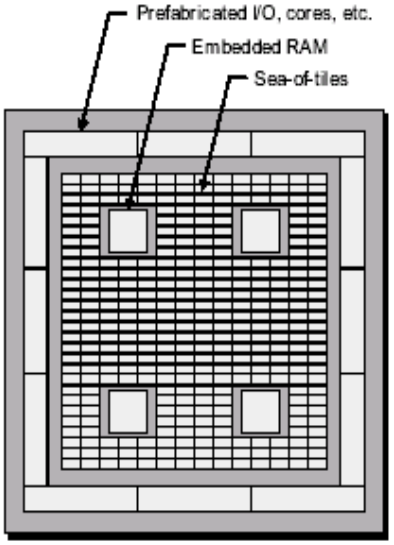
- Standard cell: pre-designed collection of logic functions
- Contains both schematic and layout view

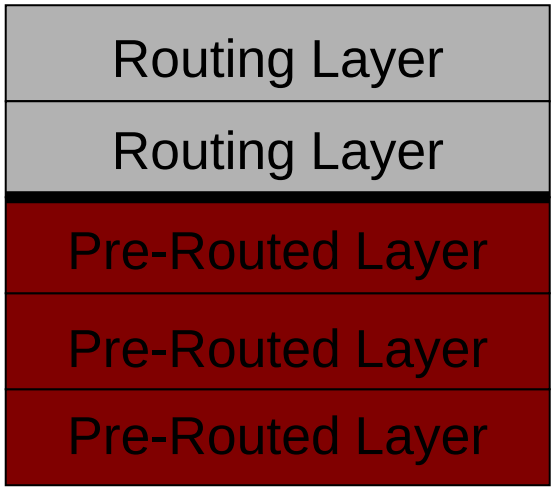




Semi-custom

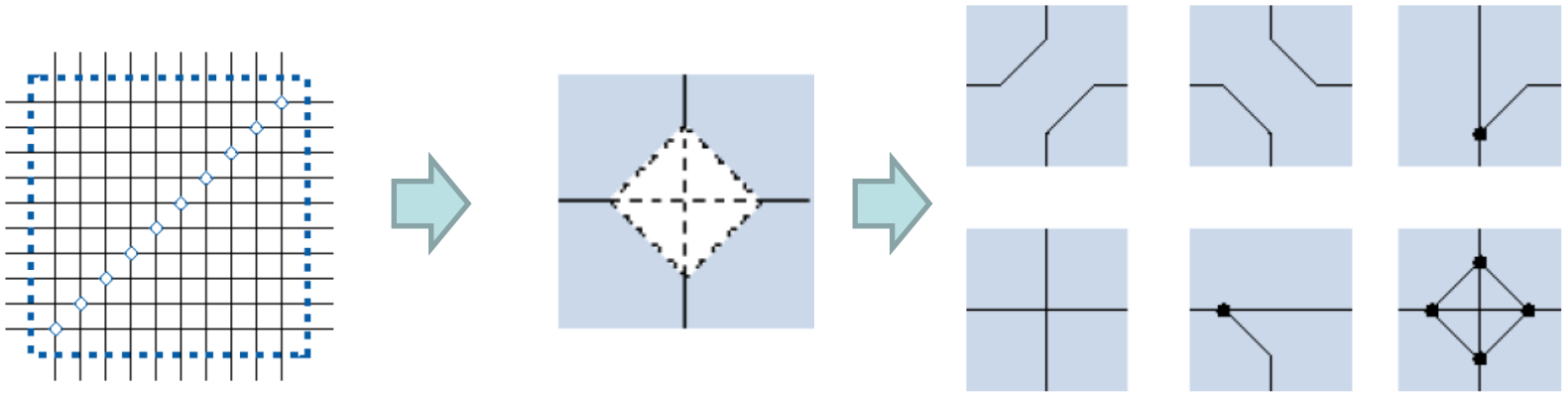
- Lower layers are fully or partially built
  - Designers are left with routing of wires and maybe placing some blocks
- Benefits
  - Good performance, good size, less NRE cost than a full-custom implementation (perhaps $10k to $100k)
  - Largely Prefabricated: components are “almost” connected in a variety of predefined configurations; only a few metal layers are needed for fabrication.
- Drawbacks
  - Still require weeks to months to develop





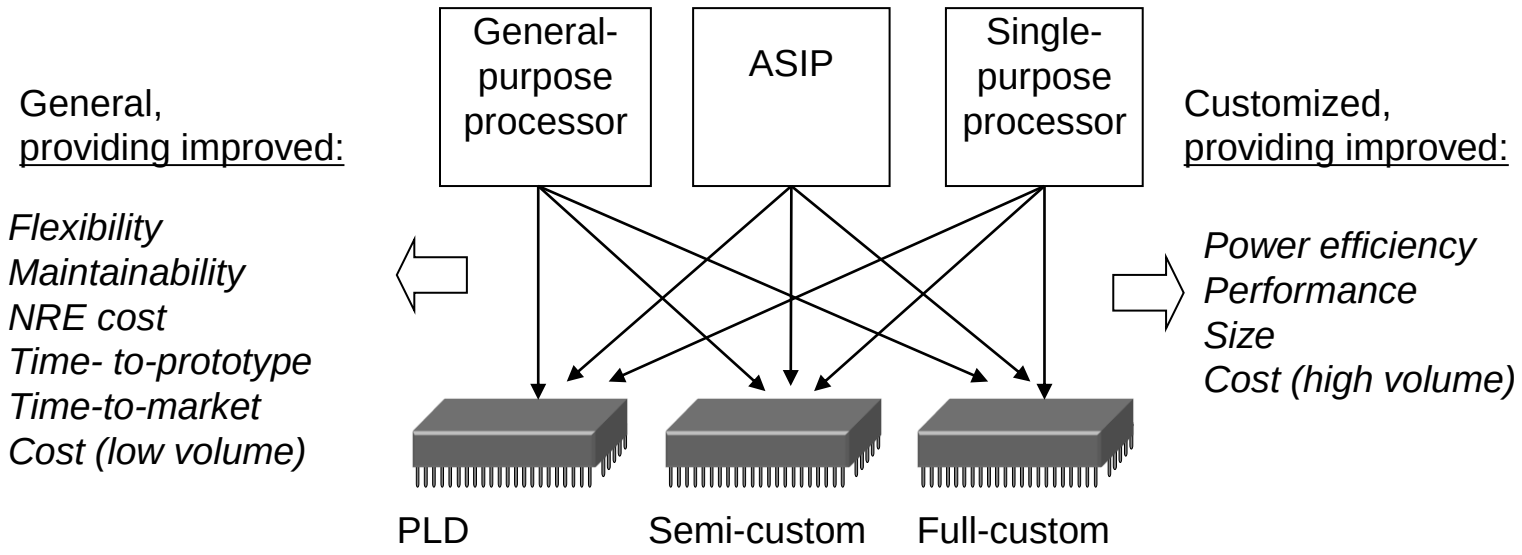
PLD (Programmable Logic Device)

- All layers already exist
  - Designers can purchase an IC
  - Connections on the IC are either created or destroyed to implement desired functionality
  - Field-Programmable Gate Array (FPGA) very popular
- Benefits
  - Low NRE costs, almost instant IC availability
- Drawbacks
  - Bigger, expensive (perhaps $30 per unit), power hungry, slower



Independence of processor and IC technologies

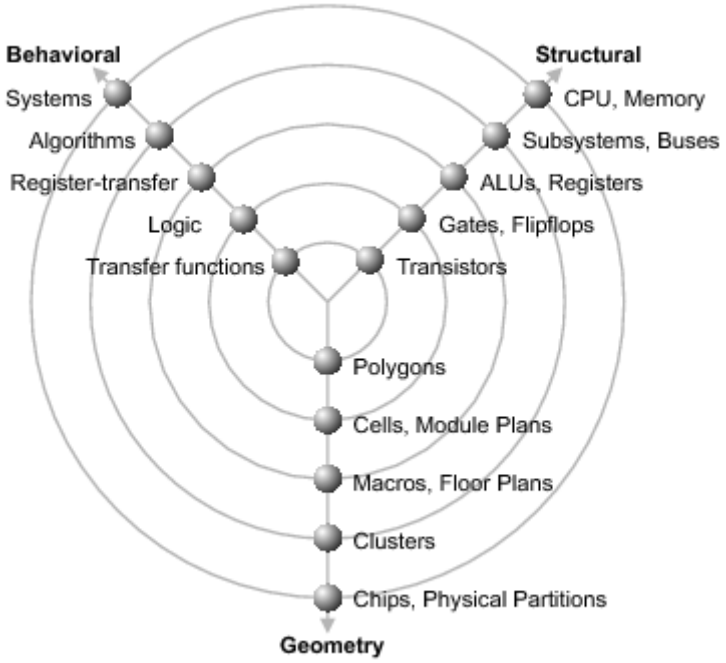
- Basic tradeoff
  - General vs. custom
  - With respect to processor technology or IC technology
  - The two technologies are independent



ASIP: Application Specific Instruction set Processor

Abstraction levels: not only for SW

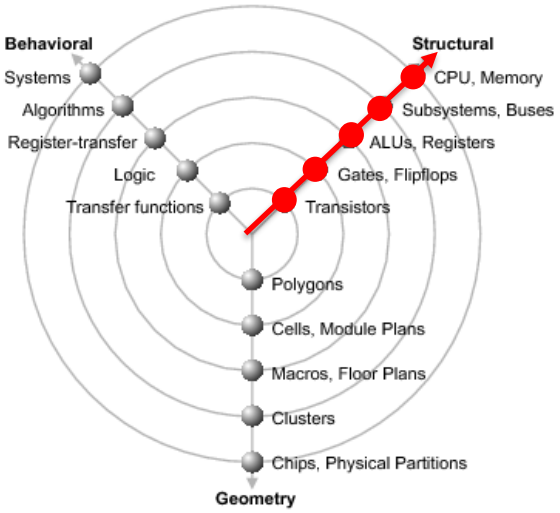
- **Designs** can be expressed / viewed in one of three possible **domains**:
  - Behavioral Domain (e.g. using boolean expressions)
  - Structural/Component Domain (e.g. connection of modules).
  - Physical Domain (e.g. layout).
- **Abstraction**: the amount the information an entity is hiding within it.
- D. Gajski and R. Kuhn developed a model called the "Y" model in 1983 (refined by D. Thomas in 1985); along the tree axis of this model are the 3 description levels with the design views in each level



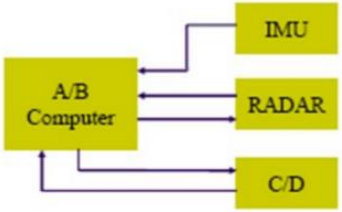
Abstraction Levels

- It is important to work on the right level of abstraction:
  - The higher the level, the shorter the design time
  - The lower the level, the more details can be fined-tuned

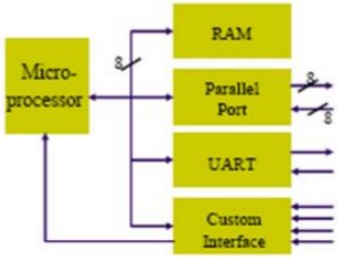
- E.g. structural design



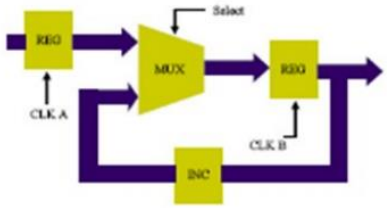
Functional blocks



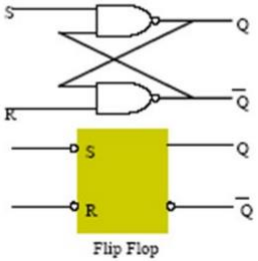
Subsystems, buses



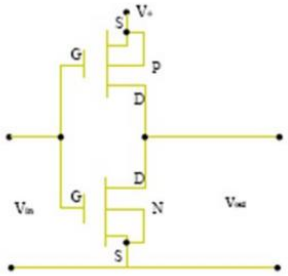
ALUs, registers…



Gates, FFs

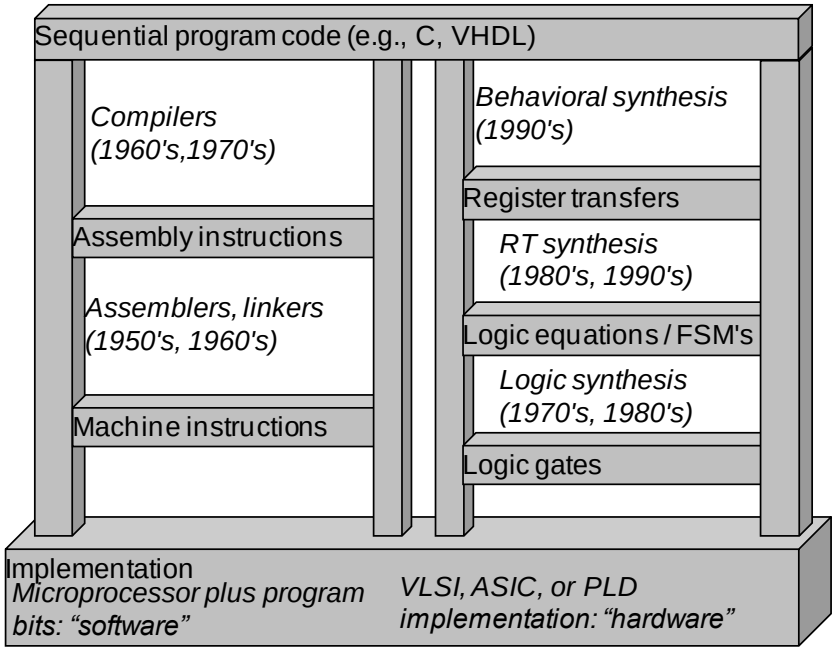


Transistors



The co-design ladder

- In the past:
  - Hardware and software design technologies were very different
  - Recent maturation of synthesis enables a unified view of hardware and software
- Now: hardware/software "codesign"
  - The choice of hardware versus software for a particular function is simply a tradeoff among various design metrics, like performance, power, size, NRE cost, and especially flexibility; there is no fundamental difference between what hardware or software can implement.

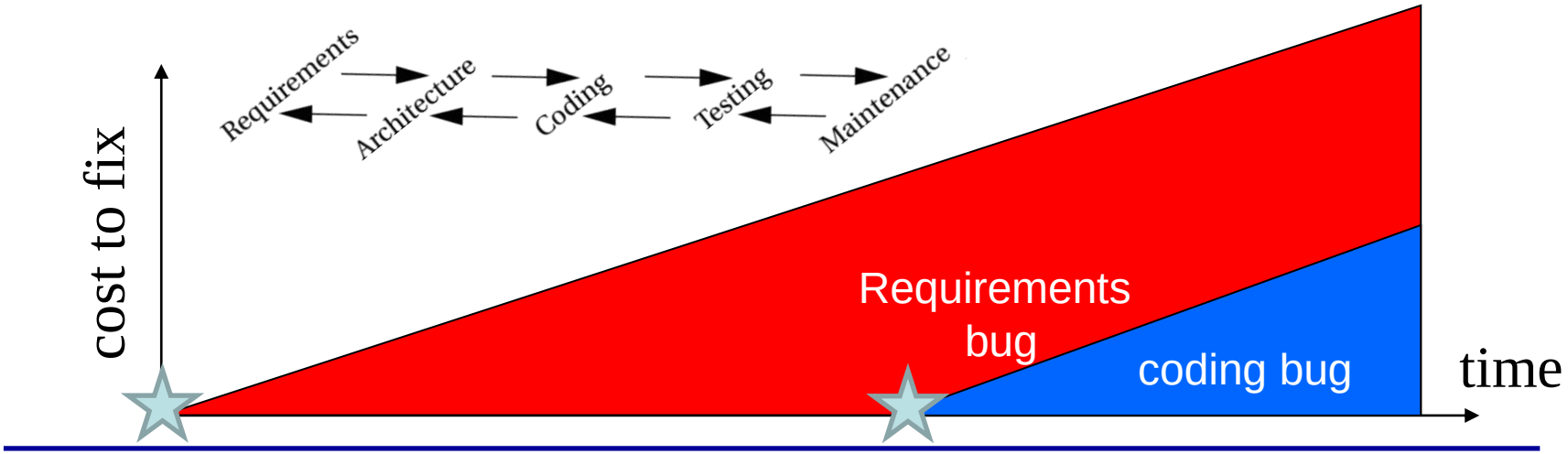


Verifying requirements and specification

- Requirements:
  - prototypes;
  - pre-existing systems.
- Specifications:
  - usage scenarios;
  - formal techniques.

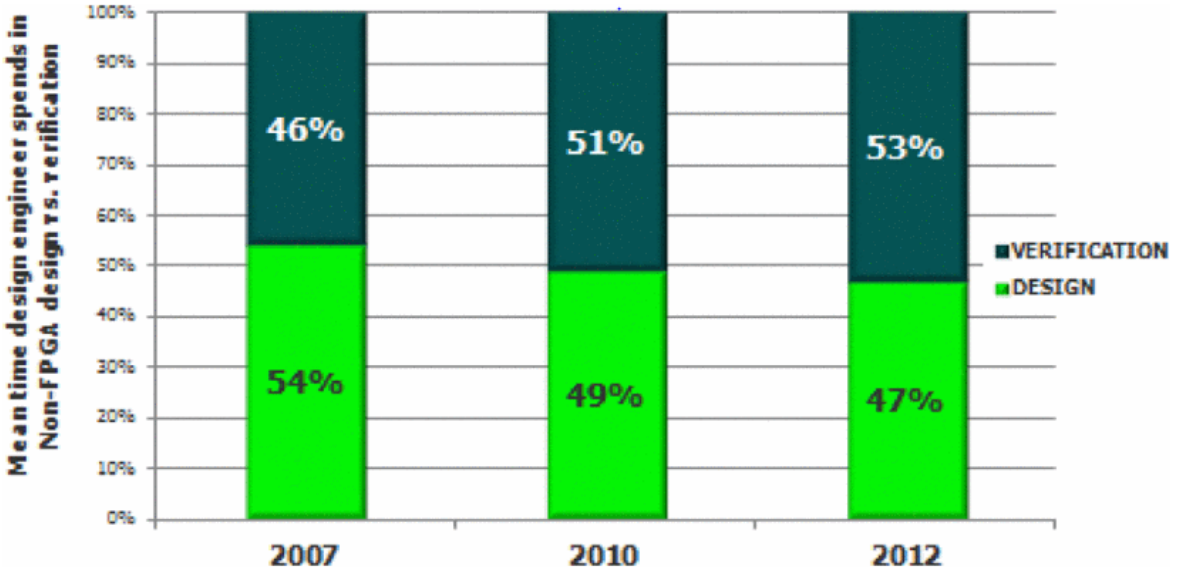
- Designers must start integrating hardware and software before they have silicon.
  - The ideal solution would allow engineers to use existing tools and models with minimal modifications. This would enable developers to work in familiar development and debug environments.

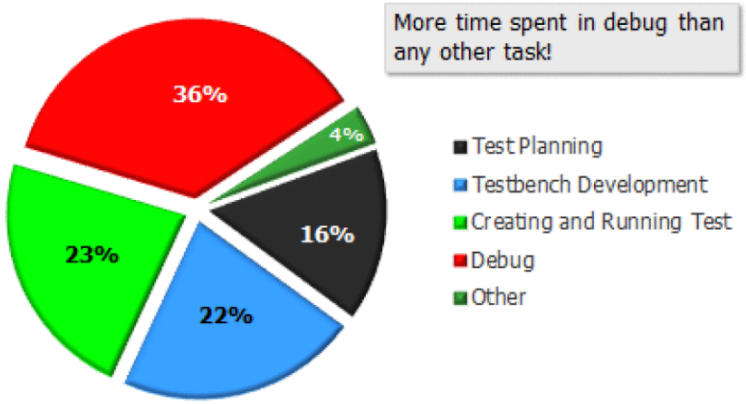
The sooner the bug is discovered, the better is!



Where Engineers Spend Their Time

- Design engineers do not spend all of their time only doing design: the mean time a design engineer spends on verification has increased in the recent past
- Verification engineers tend to spend most of their time involved in debugging

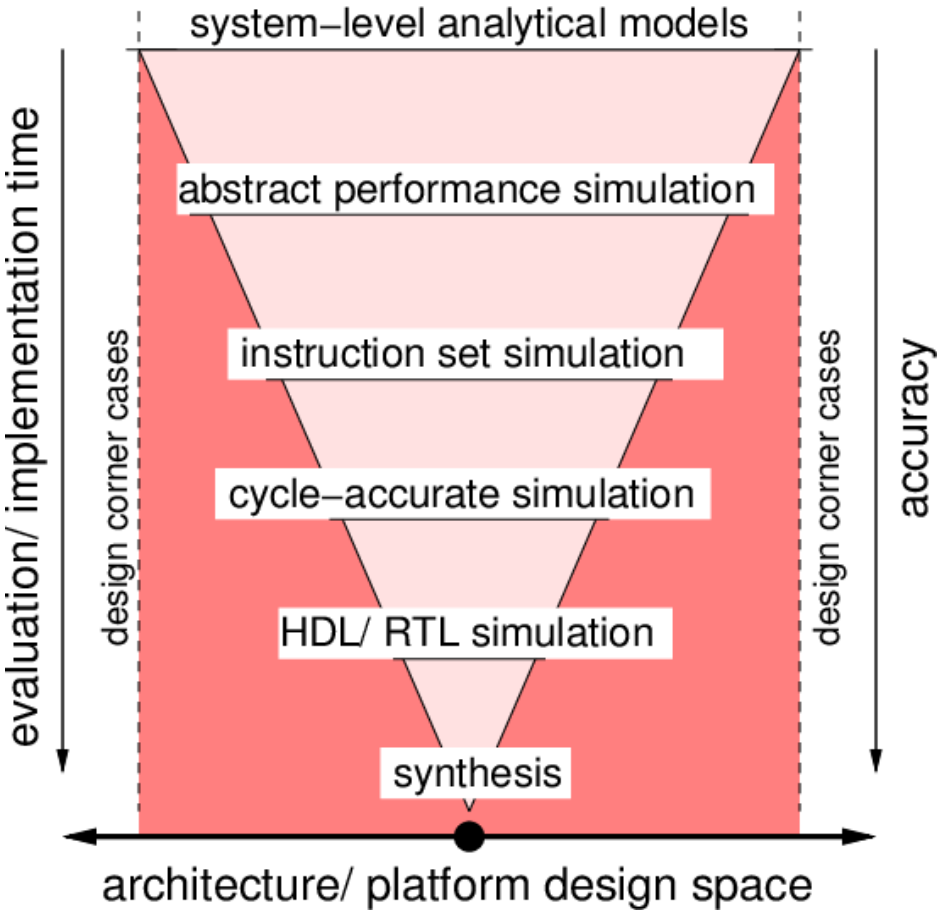




H. D. Foster, "Why the design productivity gap never happened," *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, 2013, pp. 581-584.

The cost of simulation

- Simulation pros: Controllability, Observability, Debugging
- Simulation cons: time consuming, accurateness



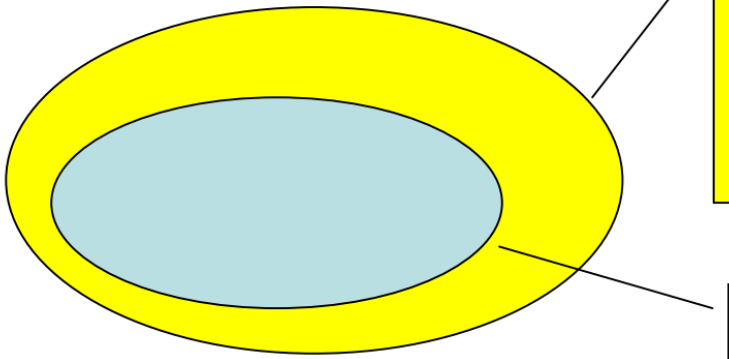
Characteristics of embedded systems

- Embedded systems perform **computations** (software) that are subject to **physical constraints** (hardware)
  - Reaction to a physical environment: deadline, throughput, jitter
  - Execution on a physical platform: processor speed, power, reliability

- Embedded systems are:
- Single-functioned
  - Dedicated to perform a single function
- Complex functionality
  - Often have to run sophisticated algorithms or multiple algorithms (cell phone, laser printer…)
- Tightly-constrained
  - Low cost, low power, small, fast, etc.
- Reactive and real-time
  - Continually reacts to changes in the system's environment
  - Must compute certain results in real-time without delay
- Safety-critical
  - Must not endanger human life and the environment

Reactive vs Real-time systems

- Crucial concern: correct and safe operations
- Logical correctness is always required
- Temporal requirements: further requirements for RT systems

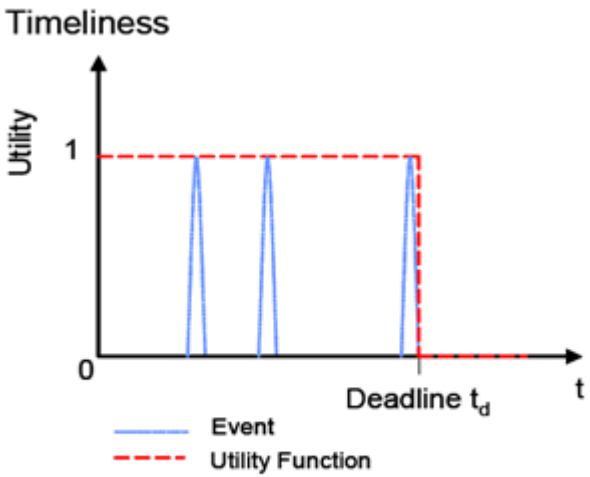


A Reactive System= a system that maintains a permanent interaction with its environment

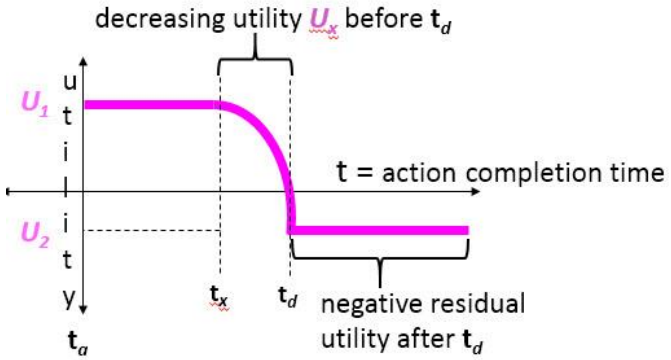
A Real-Time Reactive System = Reactive system subject to externally defined timing constraints

Determinism

- A system is said to be real-time if the total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed.
- The classical conception is that in a hard real-time or immediate real-time system, the completion of an operation after its deadline is considered useless - ultimately, this may cause a critical failure of the complete system.
- A soft real-time system on the other hand will tolerate such lateness, and may respond with decreased service quality (e.g., omitting frames while displaying a video).



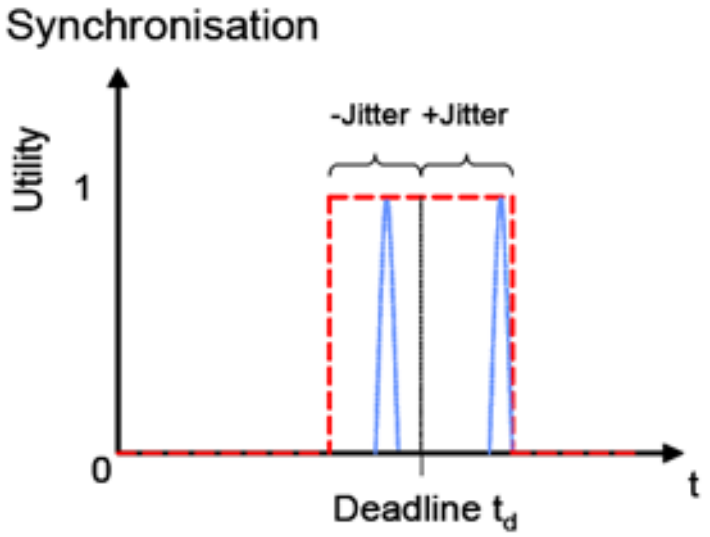
Deadline = *deadline* for execution



In general, an action has a decreasing utility value

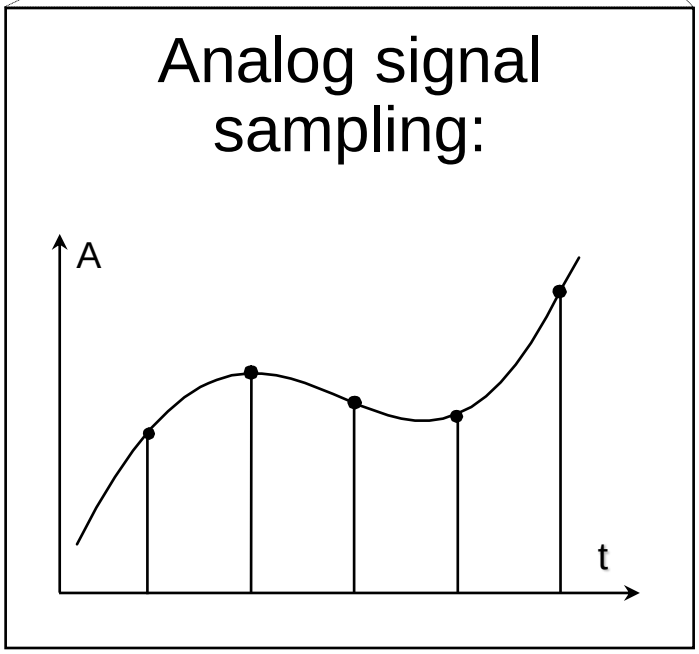
Isochronous systems

- Isochronous: it literally means to occur at the same time or at equal time intervals. In general English language, it refers to something that occurs at a regular interval, of the same duration; as opposed to synchronous which refers to more than one thing happening at the same time



Deadline = *point* of execution

Determinism & Isochronicity: an example

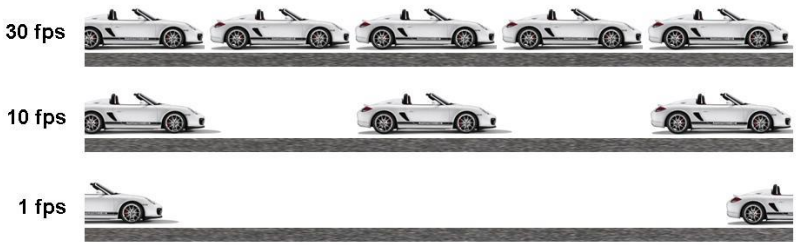


DETERMINISM

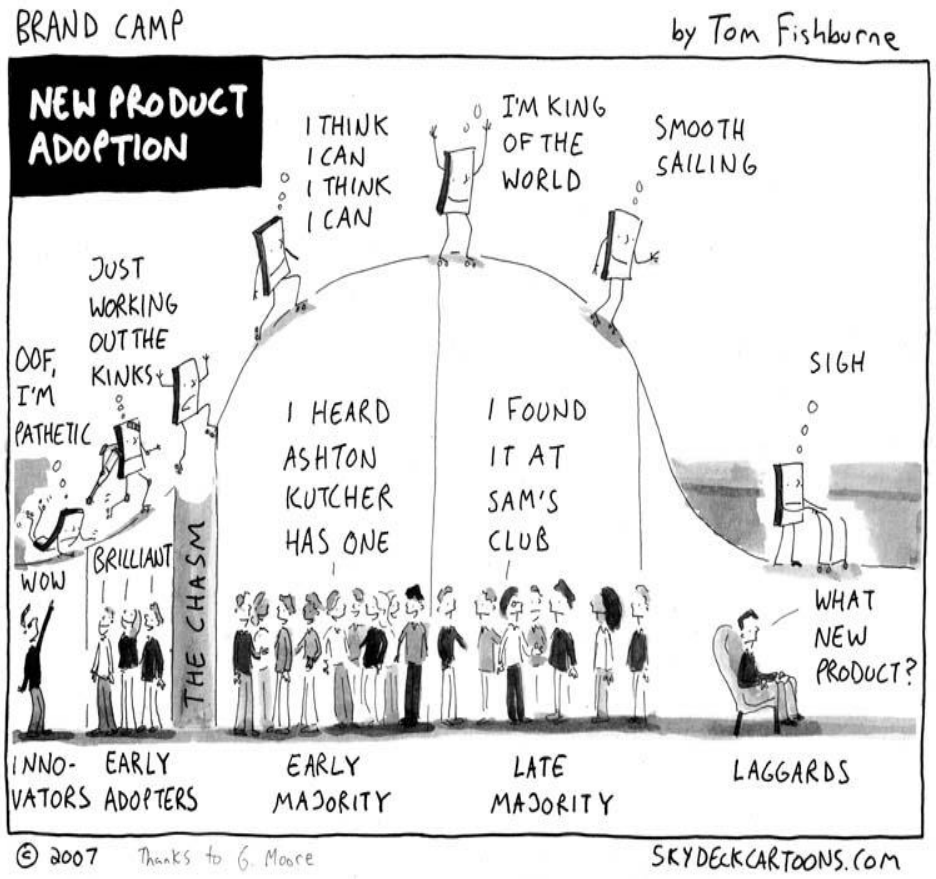
- Maximum algorithm instructin number:
  $$\frac{\text{sampling time}}{\text{instruction cycle time}}$$
- Fs = 44.1kHz → Ts = 22.6us
  F_{CLK}=200 MHz → T_{CLK}=5ns
  Ninstruction ~ 4500

The performance design metric

- Widely-used measure of system, widely-abused
  - Clock frequency, instructions per second – not good measures
  - Digital camera example – a user cares about how fast it processes images, not clock speed or instructions per second
- Latency (response time)
  - Time between task start and end
  - e.g., Camera's A and B process images in 0.25 seconds
- Throughput
  - Tasks per second, e.g. Camera A processes 4 images per second
  - Throughput can be more than latency seems to imply due to **concurrency**, e.g. Camera B may process 8 images per second (by capturing a new image while previous image is being stored).
- *Speedup* of B over S:
  - B's performance / A's performance
  - Throughput speedup = 8/4 = 2



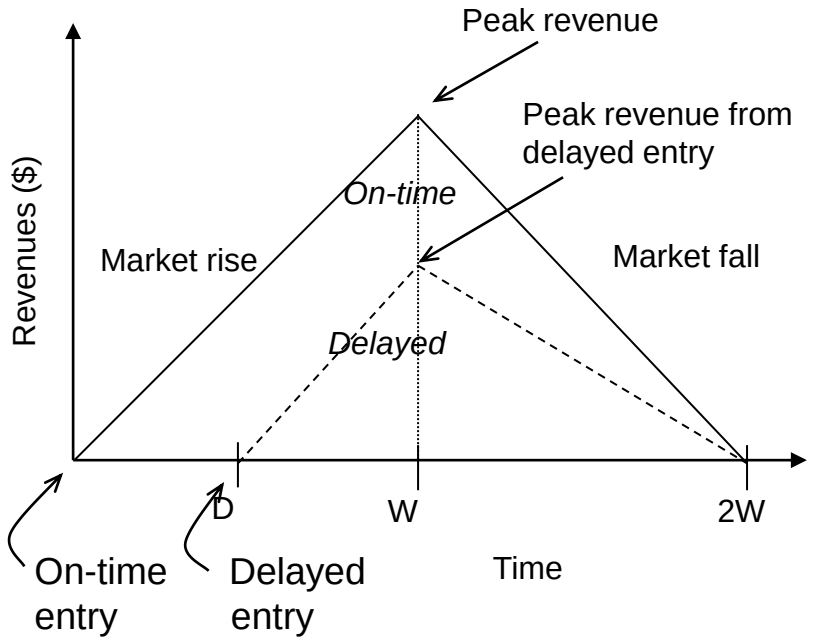
Time-to-market: a demanding design metric



- Time required to develop a product to the point it can be sold to customers
- Market window
  - Period during which the product would have highest sales
- Average time-to-market constraint is about 8 months
- Delays can be costly

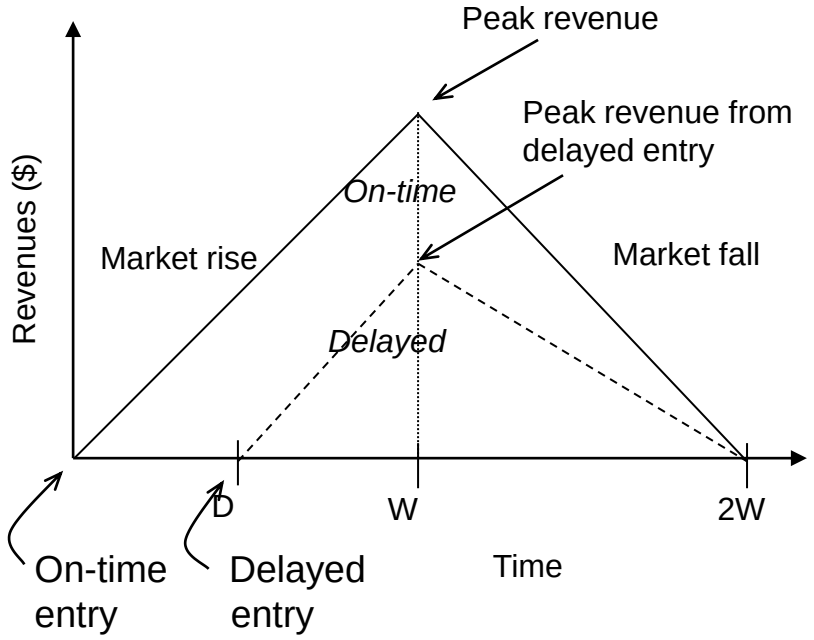
Everett Rogers, “Diffusion of Innovations”,

Losses due to delayed market entry



- Simplified revenue model
  - Product life = 2W, peak at W
  - Time of market entry defines a triangle, representing market penetration
  - Triangle area equals revenue
- Loss
  - The difference between the on-time and delayed triangle areas

Losses due to delayed market entry (cont.)



- Area = 1/2 * base * height
  - On-time = 1/2 * 2W * W
  - Delayed = 1/2 * (W-D+W)*(W-D)
- Percentage revenue loss = $(D(3W-D)/2W^2)*100\%$
- Try some examples
  - Lifetime 2W=52 wks, delay D=4 wks
  - (4*(3*26 –4)/2*26^2) = 22%
  - Lifetime 2W=52 wks, delay D=10 wks
  - (10*(3*26 –10)/2*26^2) = 50%
  - Delays are costly!

NRE and unit cost metrics

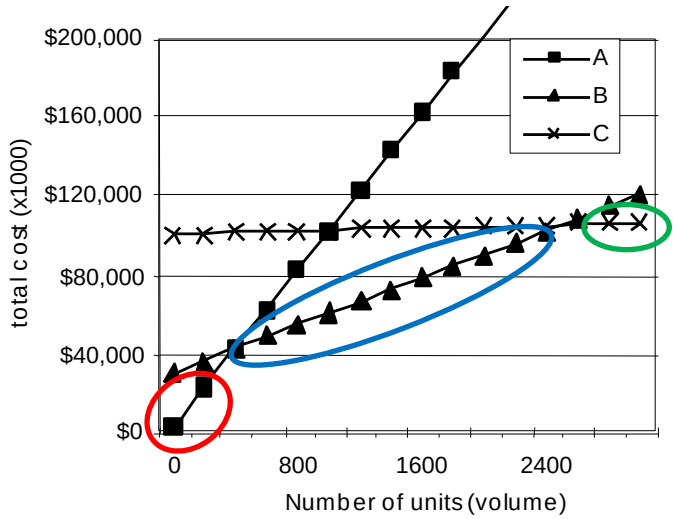
- Costs:
  - Unit cost: the monetary cost of manufacturing each copy of the system, excluding NRE cost
  - NRE cost (Non-Recurring Engineering cost): The one-time monetary cost of designing the system
  - *total cost = NRE cost + unit cost * # of units*
  - *per-product cost = total cost / # of units*
    = (NRE cost / # of units) + unit cost
- Example
  - NRE=$2000, unit=$100
  - For 10 units
    - total cost = $2000 + 10*$100 = $3000
    - per-product cost = $2000/10 + $100 = $300

Amortizing NRE cost over the units results in an additional $200 per unit

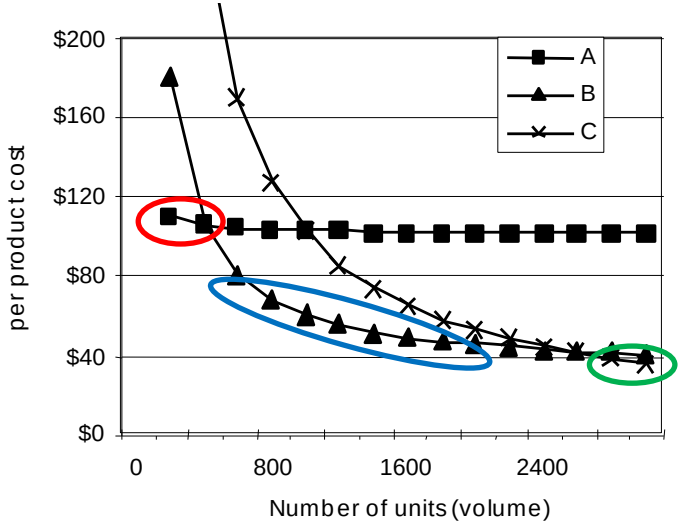
NRE and unit cost metrics

- Compare technologies by costs -- best depends on quantity
  - Technology A: NRE=$2,000, unit=$100
  - Technology B: NRE=$30,000, unit=$30
  - Technology C: NRE=$100,000, unit=$2

*NRE cost + unit cost * # of units*



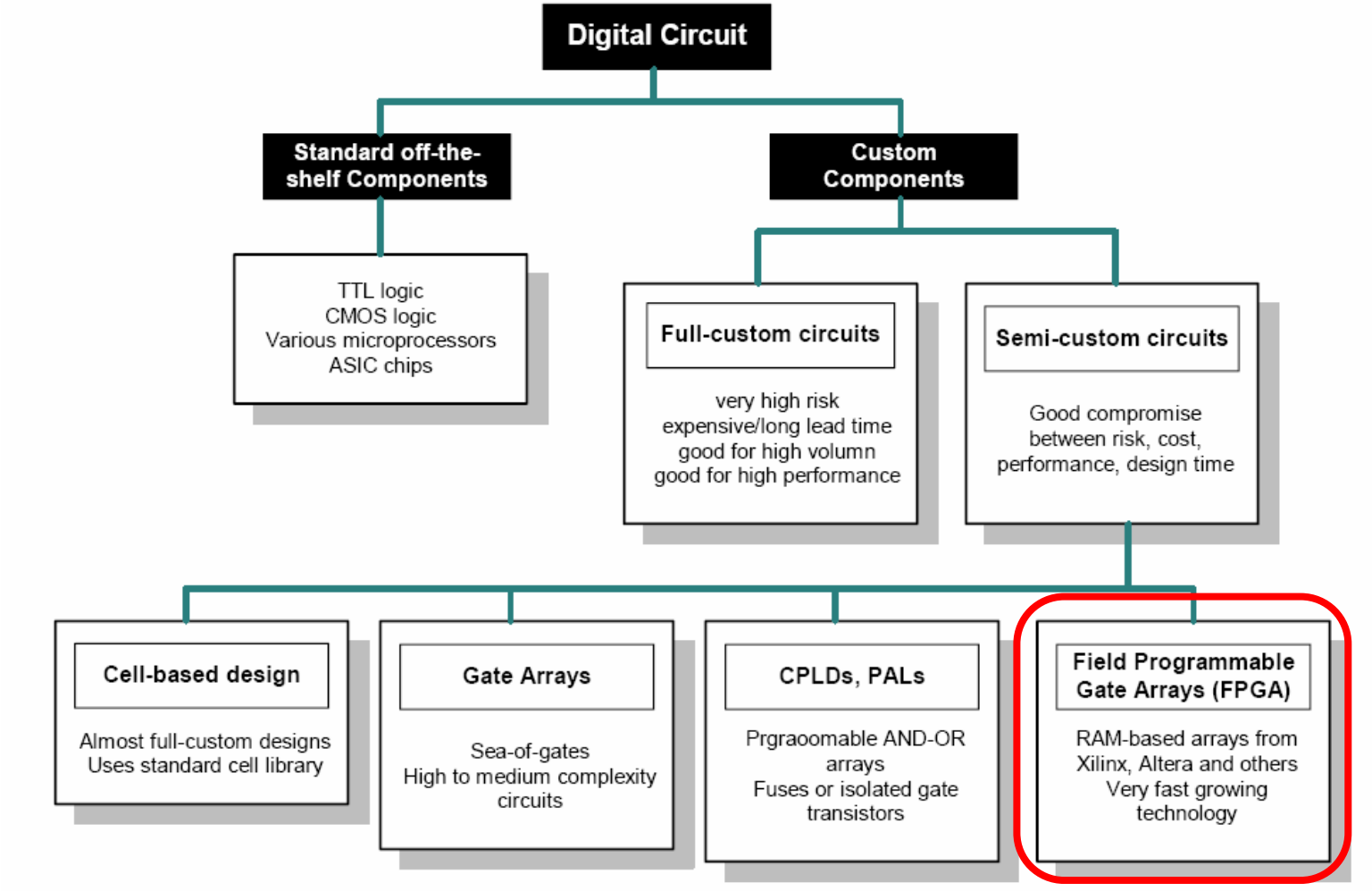
(NRE cost / # of units) + unit cost



- But, must also consider time-to-market

PLDs

Classification of digital ICs



PLD

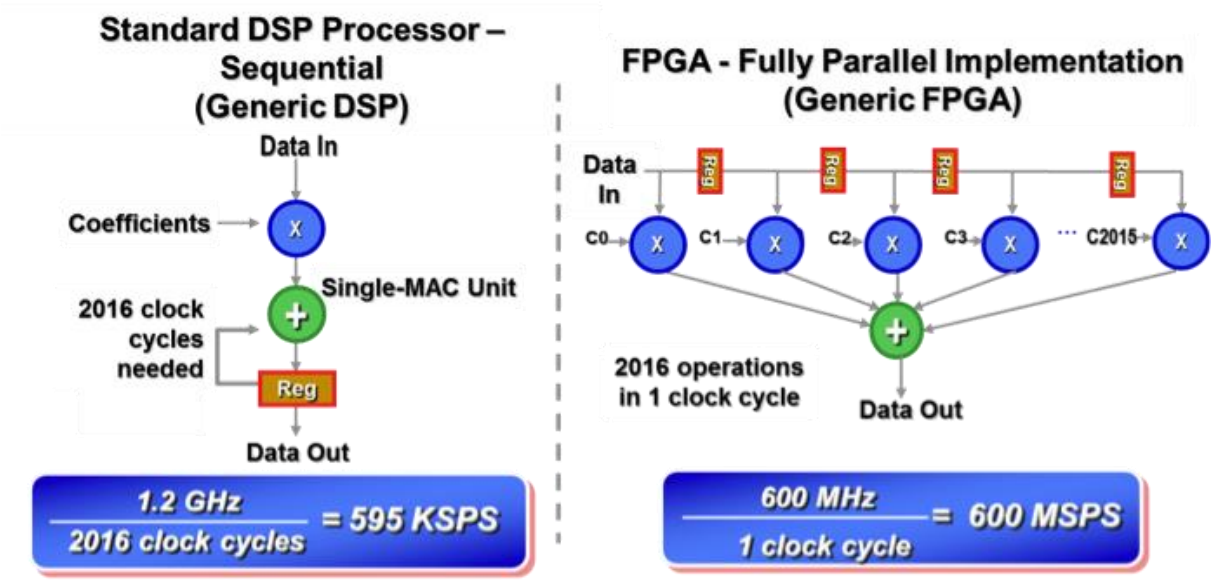
- **Programmable Logic Device** is a generic term that covers all subfamilies of programmable logic.

- Today, PLDs can be programmed to incorporate several **CPU (soft) cores**. We can inexpensively offload some of the special-purpose tasks to the peripheral processor(s), in a real multiprocessing context.
- PLD-based projects are usually developed using Verilog, **VHDL**, or some other hardware design language (HDL). This approach significantly simplify the development process in comparison to a hardware-based breadboard approach.

- Possible advantages from parallel implementation and pipelining

Serial vs Parallel Implementation

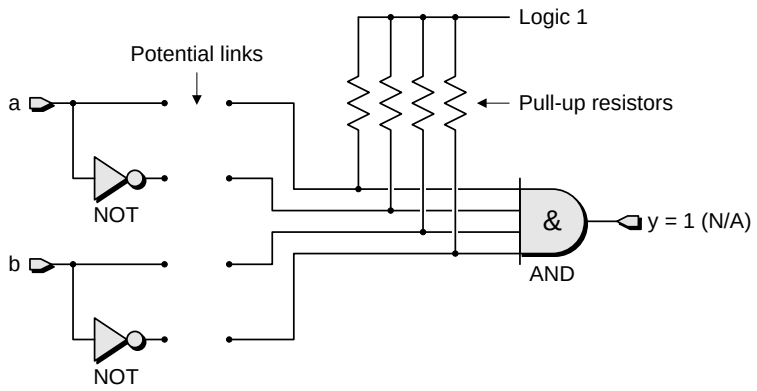
- Consider the implementation of a 2016-tap FIR filter
  - DSPs use a serial solution (supposing a single-MAC unit)
  - PLDs can exploit the parallelism inherent in hardware (FIR filter have symmetrical coeffs… semiparallel implementation using half the multipliers…)

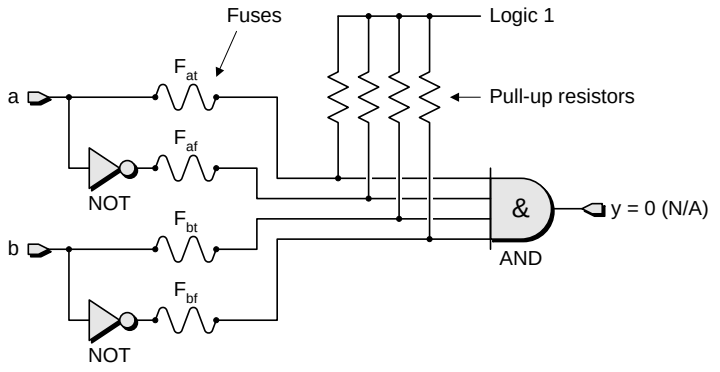


PLD Technologies

Fuse technologies

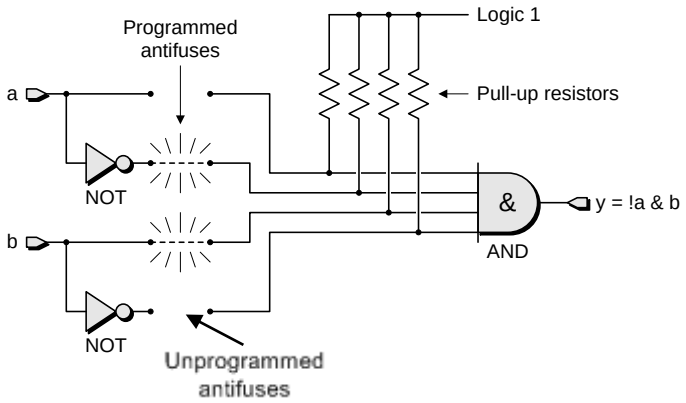
- Consider a very simple programmable function with two inputs called a and b and a single output y
  - The inverting (NOT) gates associated with the inputs mean that each input is available in both its *true (unmodified)* and *complemented (inverted) form.*
  - One of the first techniques to program devices was the fusible-link technology. In this case, the device is manufactured with all of the links in place, where each link is referred to as a fuse
  - Design engineers can selectively remove undesired fuses by applying pulses of relatively high voltage and current to the device's inputs.

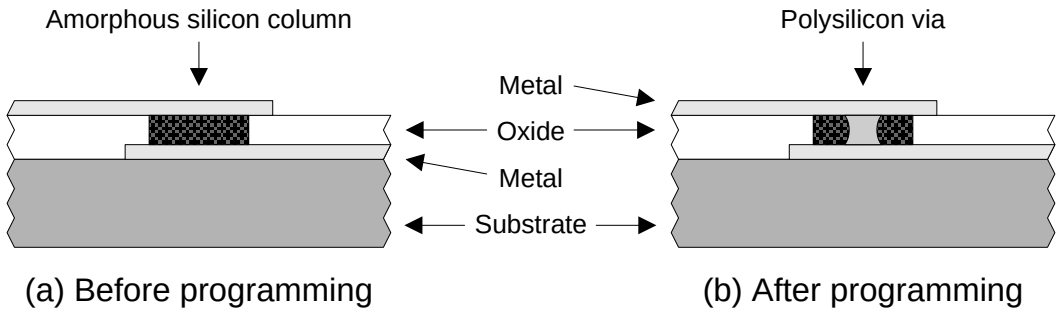




Antifuse technologies

- In antifuse technologies configurable path has an associated link called an antifuse.
  - In its unprogrammed state, an antifuse has such a high resistance that it may be considered an open circuit; they can be selectively programmed by applying pulses of relatively high voltage and current to the device's inputs.
  - An antifuse commences life as a microscopic column of amorphous (noncrystalline) silicon linking two metal tracks: very high resistance.
  - A via is created by converting the insulating amorphous silicon into conducting polysilicon



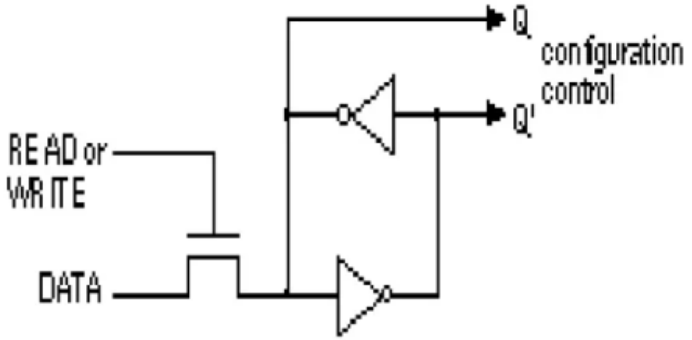


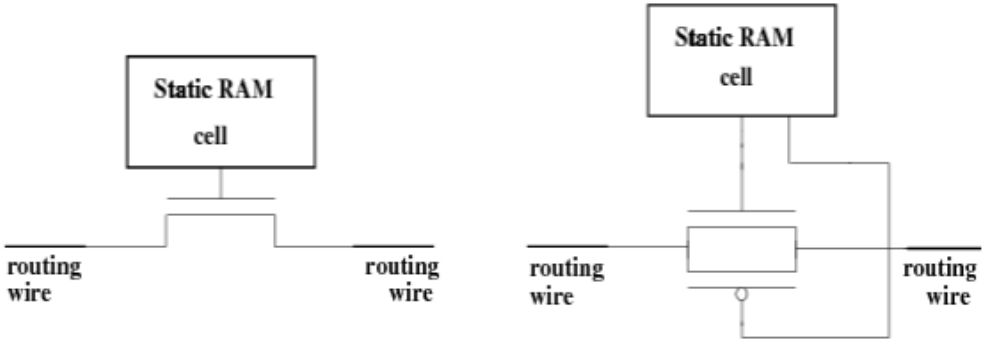
(a) Before programming (b) After programming

SRAM configuration cell

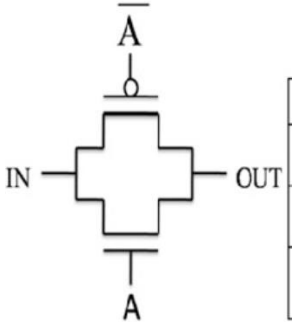
- Consists of 2 cross-coupled inverters
- Uses standard (CMOS) process

- The output of the configuration cell is used to drive the gates of other transistors (pass transistors, transmission gate) to make or break the connection





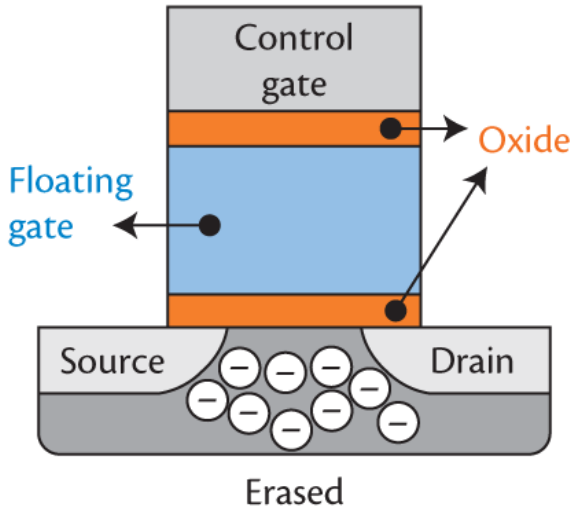
Transmission gate:



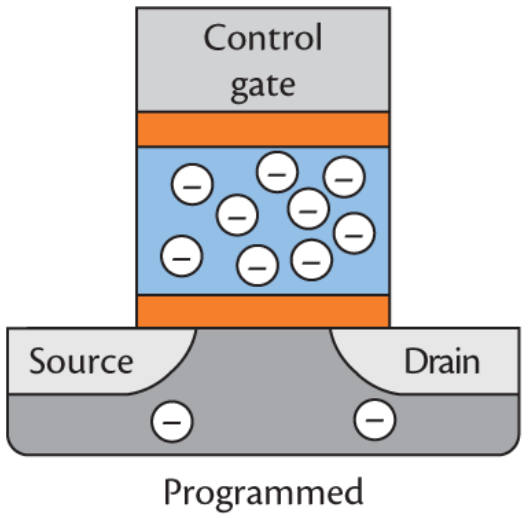
A	IN	OUT
H	H	H
H	L	L
L	X (don't care)	Z(high impedance)

Flash technology

- A floating gate transistor constitutes a flash memory cell.
- Encoded data are represented by the level of charge stored inside the transistor's floating gate.
- The transistor traps charge within its floating gate, which dictates the threshold voltage level at which the transistor turns on.
- The threshold voltage level of the floating gate is used to determine the value of the digital data stored inside the transistor.

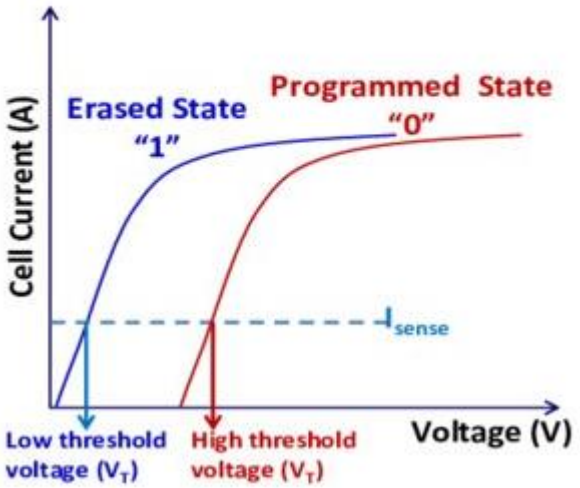


Erased



Programmed

(b) I-V characteristics of memory cell



Technologies: summary

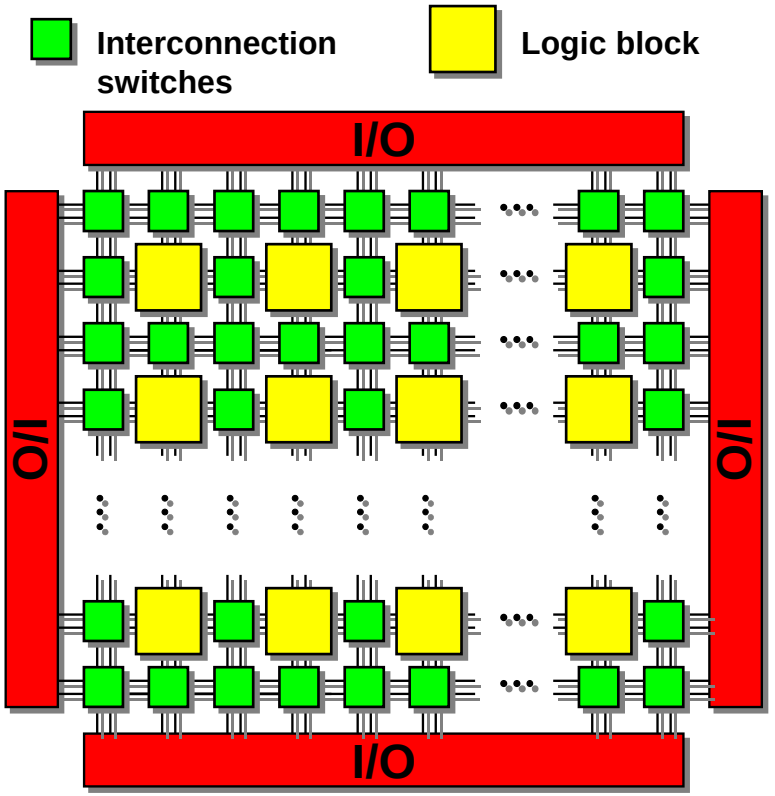
- Antifuse prevents reverse engineering.
  - Once the device has been programmed, it is possible to set a special security antifuse that subsequently prevents any programming data from being read out of the device.
  - Even if the device is decapped (its top is removed), programmed and unprogrammed antifuses appear to be identical (they are buried in the internal metallization layers)

Feature	SRAM	Antifuse	E2PROM / FLASH
Technology node	State-of-the-art	One or more generations behind	One or more generations behind
Reprogrammable	Yes (in system)	No	Yes (in-system or offline)
Reprogramming speed (inc. erasing)	Fast	----	3x slower than SRAM
Volatile (must be programmed on power-up)	Yes	No	No (but can be if required)
Requires external configuration file	Yes	No	No
Good for prototyping	Yes (very good)	No	Yes (reasonable)
Instant-on	No	Yes	Yes
IP Security	Acceptable (especially when using bitstream encryption)	Very Good	Very Good
Size of configuration cell	Large (six transistors)	Very small	Medium-small (two transistors)
Power consumption	Medium	Low	Medium
Rad Hard	No	Yes	Not really

FPGAs

FPGA

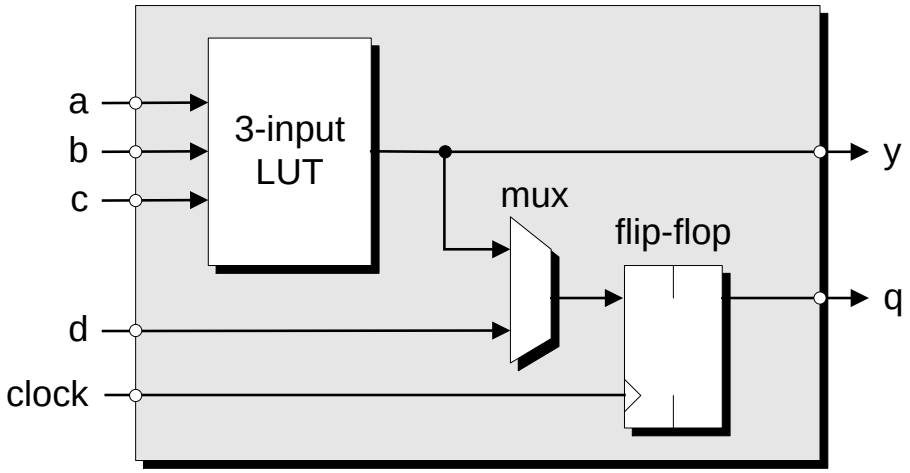
- FPGA consists of an array of programmable basic logic cells (Configurable or Programmable Logic Blocks CLBs - Xilinx, Logic Elements - Altera) surrounded by programmable interconnect.
- Usually it is a SRAM device. At power ON, the configuration is downloaded to the device as a serial bit stream from an external non volatile memory. Thus, the configuration of the device can be dynamically changed at runtime



LUT based FPGA

- The early devices comprised a 3-input lookup table (LUT), a register that could act as a flip-flop or a latch, and a multiplexer
- For example, assume that a LUT was required to perform the function:

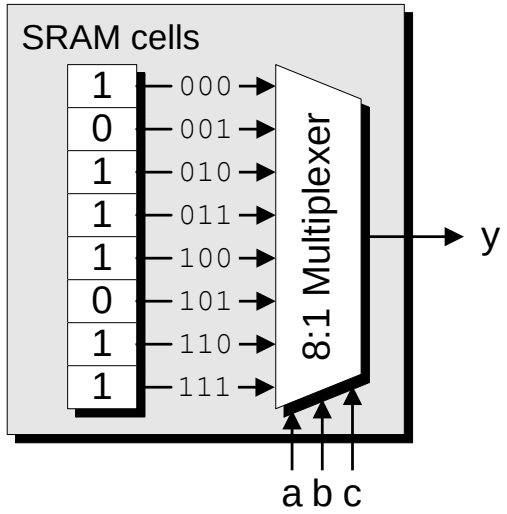
  y = (a & b) | !c



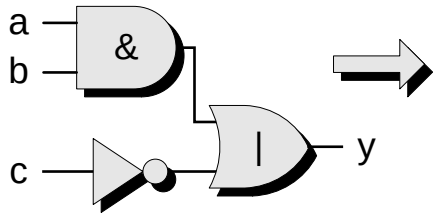
Truth table

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

Programmed LUT



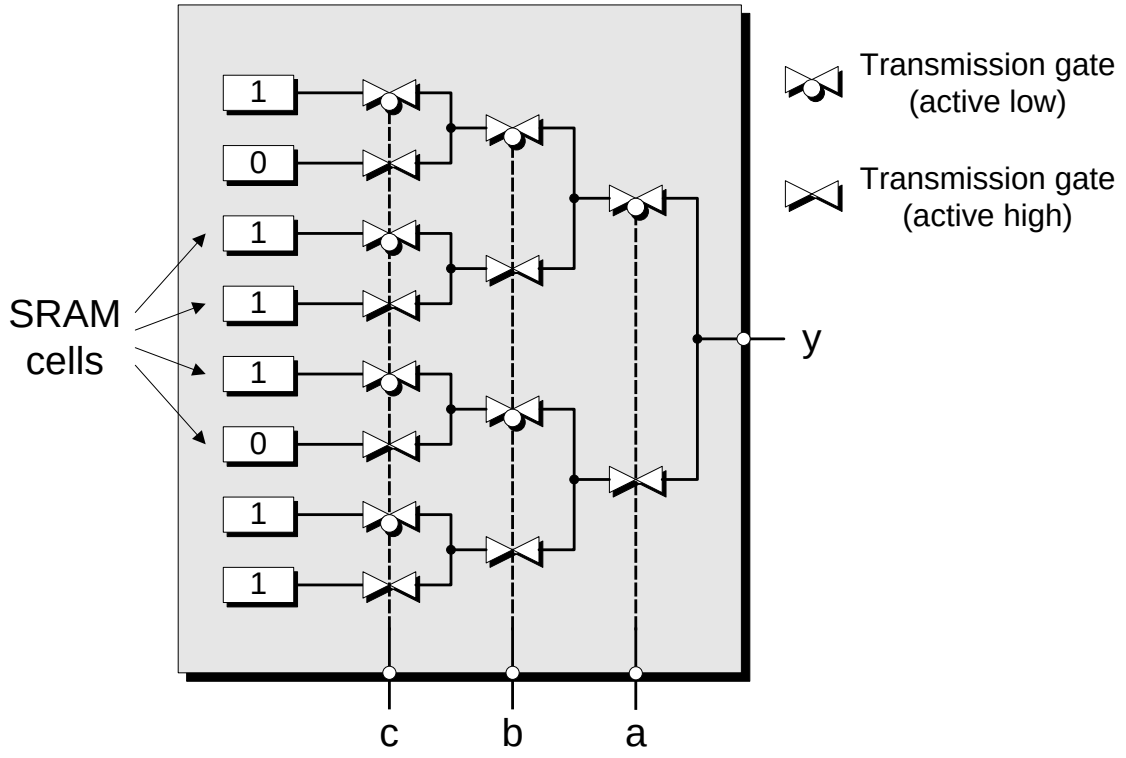
Required function



y = (a & b) | !c

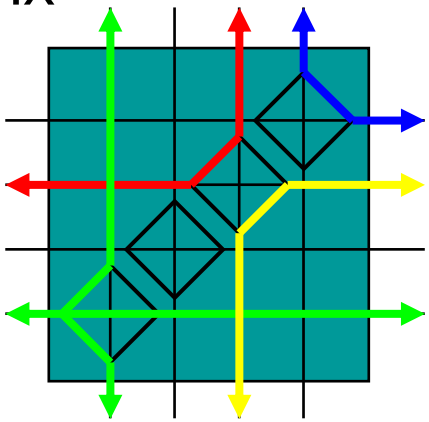
LUT based FPGA

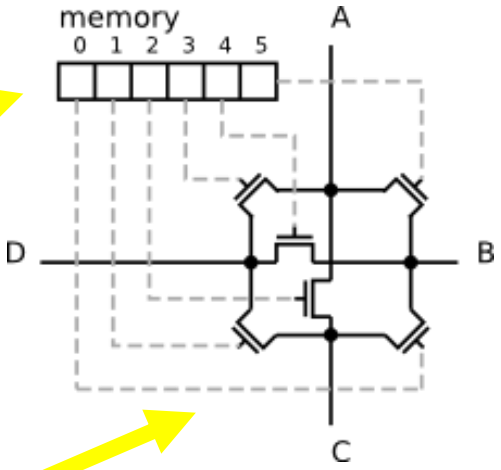
- A commonly used technique to implement the LUT is to use the inputs to select the desired SRAM cell using a cascade of transmission gates

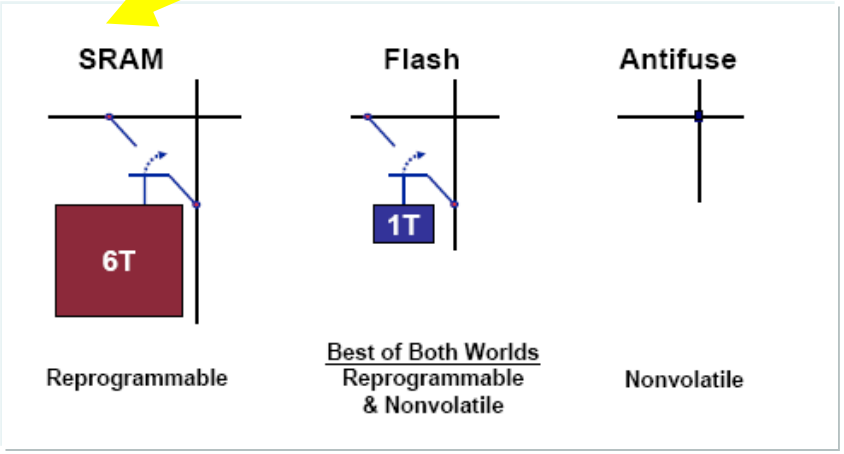


Switch Matrix Operation

Programmed Switch Matrix







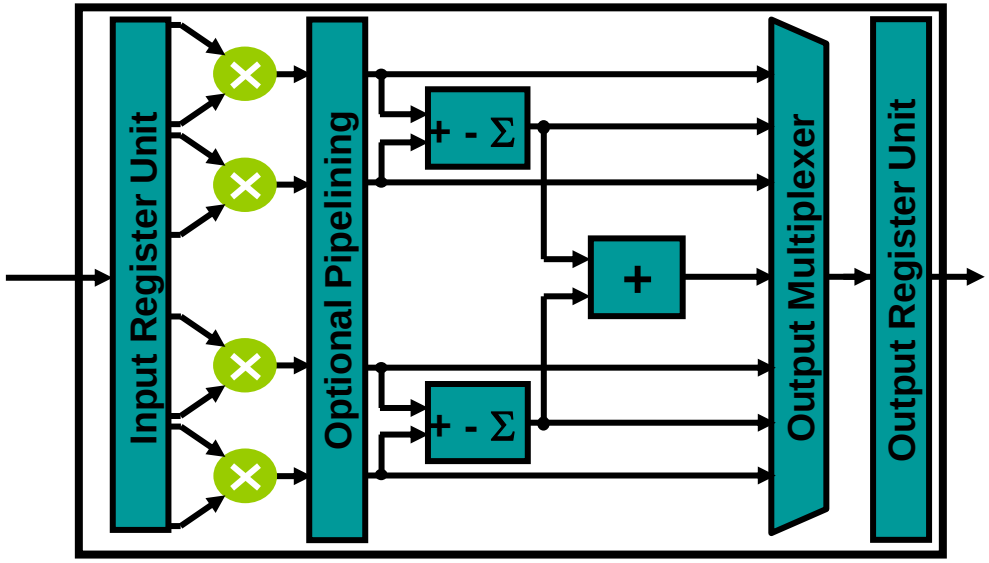
- 6 pass transistors per switch matrix interconnect point
- Pass transistors act as programmable switches
- Pass transistor gates are driven by configuration memory cells

Embedded multipliers, adders, MACs

- Some functions, like multipliers, are inherently slow if they are implemented by connecting a large number of programmable logic blocks together.
- Since these functions are required by a lot of applications, many FPGAs incorporate special hardwired multiplier blocks.

This **DSP block** consists of:

- A multiplier block
- An adder (subtractor/accumulator) block
- A summation block
- An output interface
- Output registers
- Routing and control signals



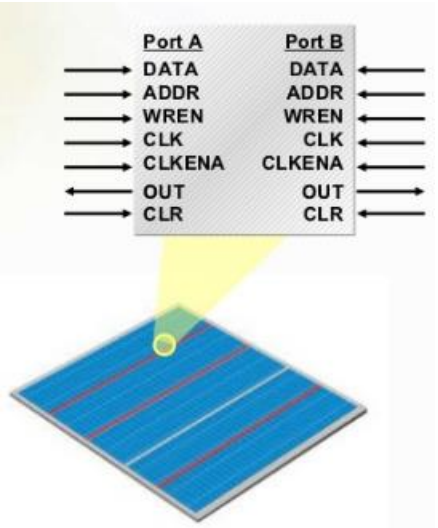
Embedded Memory

- Using LUTs as registers does not provide enough space or versatility.
- Time-dependent applications, performing many computations, need an entire built-in memory.
- The main advantages of embedded (built-in) memory are:

- short access time
- high bandwidth
- great versatility

It can behave like:

- RAM (single and dual port)
- ROM
- Buffer (FIFO, LIFO, etc.)
- Cache
- Shift registers
- etc…



HDL

Traditional PL vs HDL

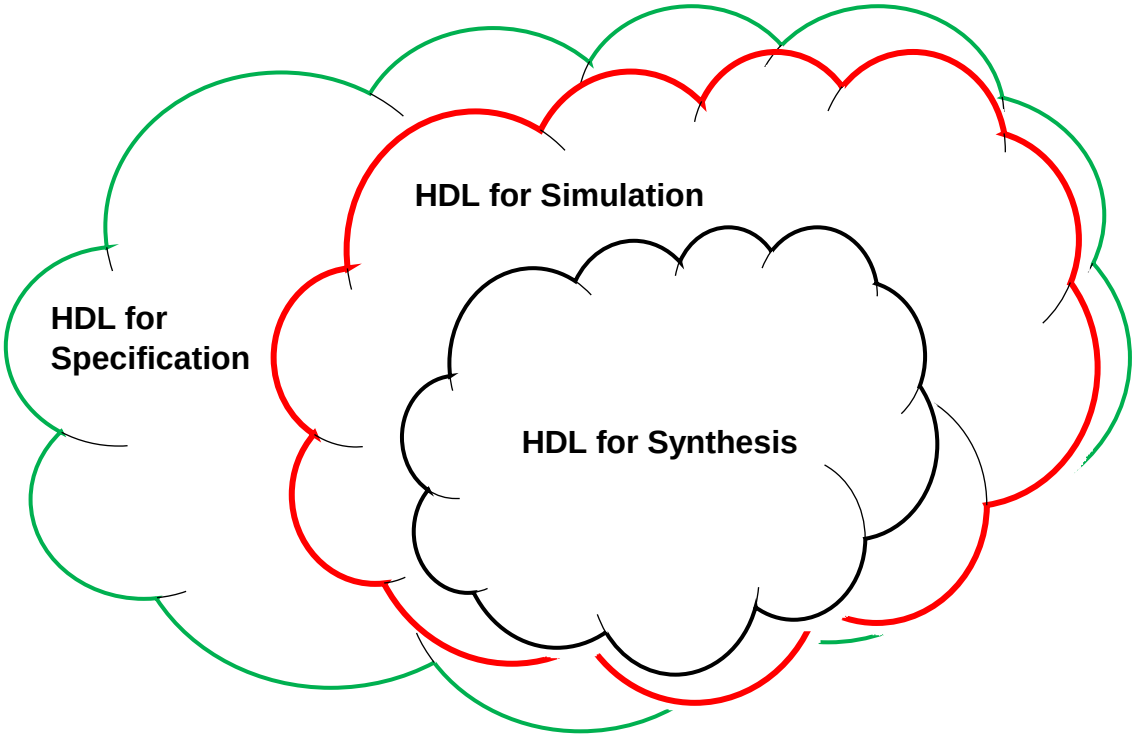
- Traditional PL: Modeled after a sequential process
  - Operations performed in a sequential order
  - Help human's thinking process to develop an algorithm step by step
  - Resemble the operation of a basic computer model

- Characteristics of digital hardware
  - Connections of parts
  - Concurrent operations
  - Concept of propagation delay and timing
- These Characteristics cannot be captured by traditional PLs

Require new languages: HDL

Use of an HDL program

- Formal documentation
- Input to a simulator
- Input to a synthesizer

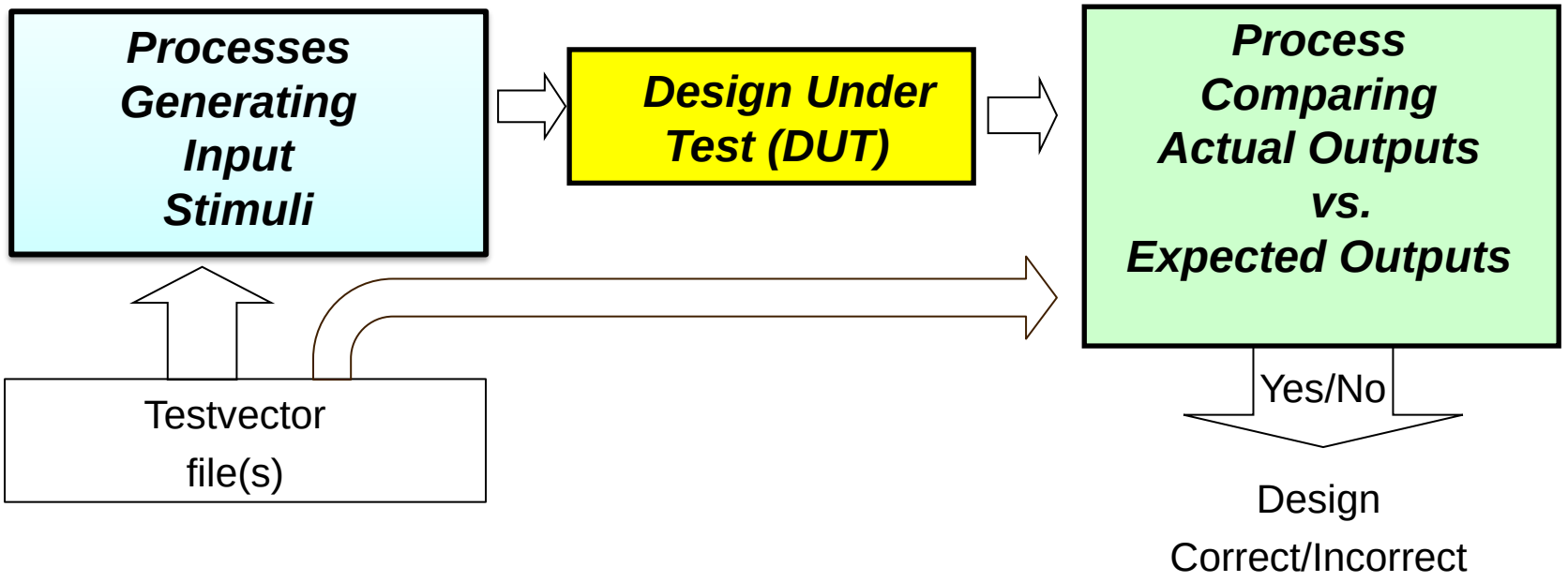


HDL highlights

- Highlights of modern HDL:
  - Encapsulate the concepts of entity, connectivity, concurrency, and timing
  - Incorporate propagation delay and timing information
  - Consist of constructs for structural implementation
  - Incorporate constructs for behavioral description (sequential execution of traditional PL)
  - Describe the operations and structures in gate level and RT level.
  - Consist of constructs to support hierarchical design process
  - Technology/vendor independent; Portable and Reusable
- Two solutions mainly used: VHDL and Verilog
  - Syntax and “appearance” of the two languages are very different
  - Capabilities and scopes are quite similar

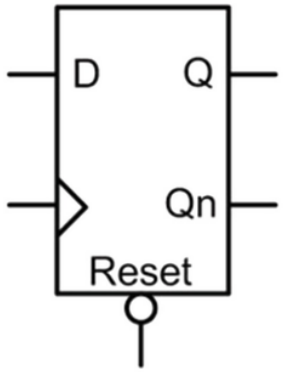
Testbench

- A “virtual” experiment table
  - Circuit to be tested
  - Input stimuli (e.g., function generator)
  - Output monitor (e.g., logic analyzer)



Sample code

Example: Behavioral Model of a D-Flip-Flop with Asynchronous Reset in VHDL



$\overline{R}$	Clk	D	Q	Qn	
0	X	X	0	1	Reset
1	0	X	Last Q	Last Qn	Store
1	1	X	Last Q	Last Qn	Store
1	⌞‾	0	0	1	Update
1	⌞‾	1	1	0	Update

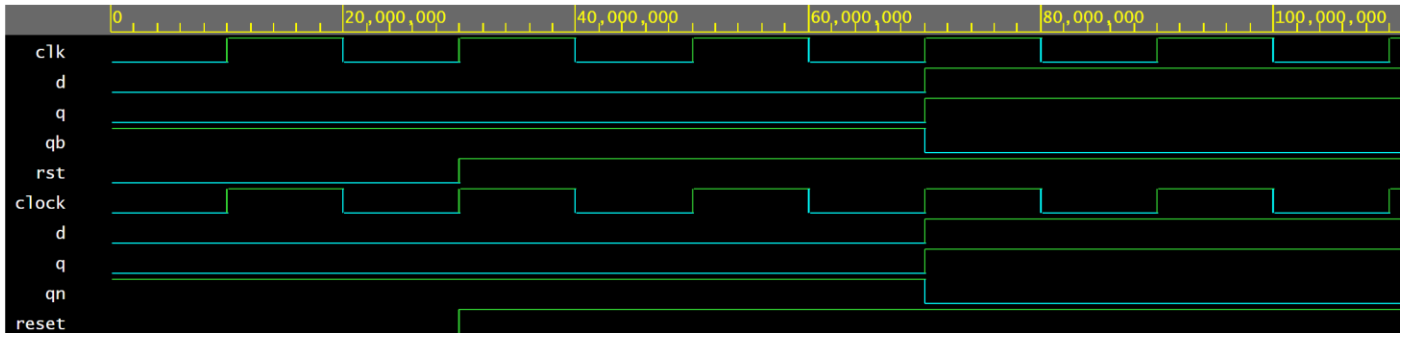
```
1  Library IEEE;
2  Use IEEE.Std_logic_1164.all;
3
4  entity DFlipFlop is
5     port(
6        Clock, Reset :in std_logic;
7        Q,Qn : out std_logic;
8        D :in  std_logic
9     );
10 end DFlipFlop;
11
12 architecture DFlipFlop_arch of DFlipFlop is
13 begin
14  process(Clock, Reset)
15  begin
16     if Reset = '0' then
17         Q <= '0'; Qn <= '1';
18     elsif rising_edge(Clock) then
19         Q <= D; Qn <= not D;
20     end if;
21  end process;
22 end DFlipFlop_arch;
```

https://www.edaplayground.com/x/Dd8Q

Sample code

- Testbench for a D-type FF

```
1  Library IEEE;
2  USE IEEE.Std_logic_1164.all;
3
4  entity DFF_tb is
5  end entity DFF_tb;
6
7  architecture tb of DFF_tb is
8    signal CLK, RST, D, Q, Qb : STD_LOGIC;
9
10 begin
11
12   ClockGen: process
13   begin
14     CLK <= '0';
15     wait for 10 NS;
16     CLK <= '1';
17     wait for 10 NS;
18   end process;
19
20   stim: process
21   begin
22     RST <= '0';
23     D <= '0';
24     wait for 30 ns;
25     RST <= '1';
26     wait for 40 ns;
27     D <= '1';
28   wait;
29   end process;
30
31 uut: entity work.DFlipFlop(DFlipFlop_arch)
32 port map(D => D, Clock => CLK, Reset => RST, Q => Q, Qn => Qb);
33
34 end architecture tb;
```



https://www.edaplayground.com/x/Dd8Q

PLD Design Flow