CS 110
Computer Architecture

FPGA

Instructor:
Sören Schwertfeger and Chundong Wang

https://robotics.shanghaitech.edu.cn/courses/ca/21s

School of Information Science and Technology SIST

ShanghaiTech University

Based on Prof. Ha Yajun's
Guest Lecture in 2018

Review: Virtual Memory

- Why virtual memory?
  - Limited physical memory space, multi-tasks, programming with ease, protection
- Base and bound registers
- Segment, and fragmentation
- Page, page table, page table entries
  - Linear and hierarchical page table
- TLB, TLB miss and page table walk
  - Address translation in pipeline

Review: Virtual Machine

- Virtual Machine (VM)
  - Emulation: Run a complete virtual CPU & Memory & … - a complete virtual machine in software (e.g., QEMU)
  - Virtual Machine: Run as many instructions as possible directly on CPU, only simulate some parts of the machine) (e.g., VirtualBox)
- Hardware supports for virtualization
  - AMD-V, Intel VT-x
- Translation from guest virtual memory page to host physical memory page
- Kernel samepage merging (KSM)
- Cloud servers based on virtual machines
  - Virtual machine migration

Embedded System Design

An example of embedded system

Embedded System

- An embedded system is nearly any computing system (other than a general-purpose computer) with the following characteristics
  - Specifically-functioned
    - Typically, is designed to perform predefined function
  - Tightly constrained
    - Tuned for low cost
    - Single-to-fewer components based
    - Performs functions fast enough
    - Consumes minimum power
  - Reactive and real-time
    - Must continually monitor the desired environment and react to changes
  - Hardware and software co-existence

Embedded Systems Examples

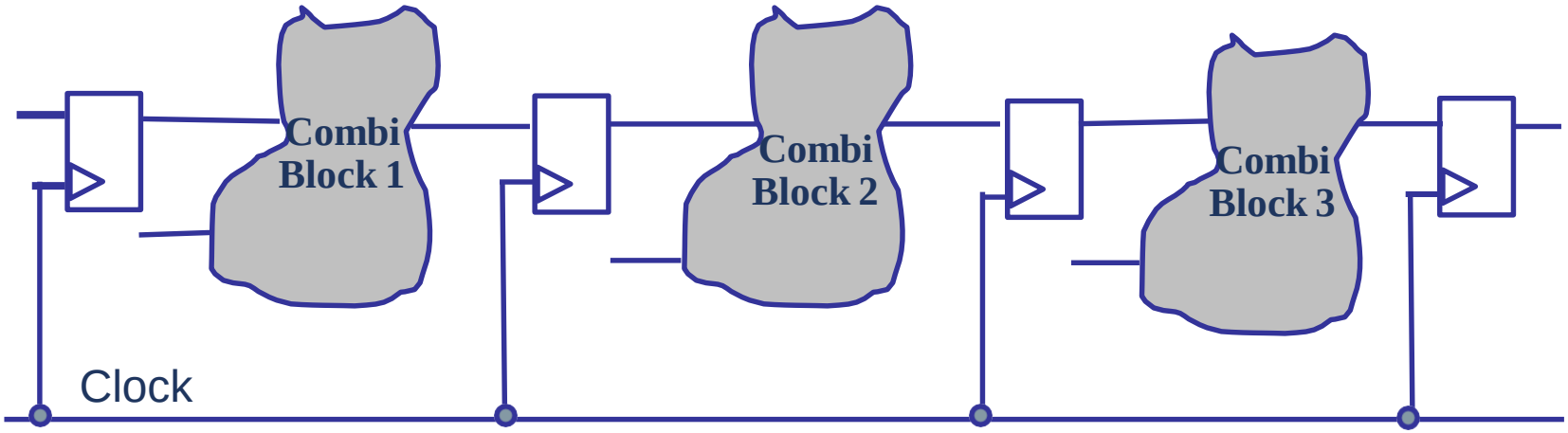
- Examples:
  - Communication devices
    - Wired and wireless routers and switches
  - Automotive applications
    - Braking systems, traction control, airbag release systems, and cruise-control applications
  - Aerospace applications
    - Flight-control systems, engine controllers, auto-pilots and passenger in-flight entertainment systems
  - Defence systems
    - Radar systems, fighter aircraft flight-control systems, radio systems, and missile guidance systems




Major Design Metrics to be Considered

- Timing performance
- Power consumption
- Chip area (Cost)
- Technology
- Reliability
- Testability
- Availability of CAD tools, libraries, IP's
- Time-to-market
- … …

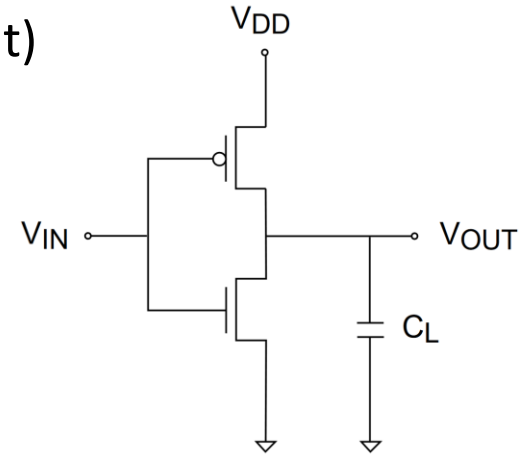
Timing Performance



- Clocks are used to synchronize the start of computations in all combinational blocks.
- Clock period is determined by finding out the longest path delay of all combinational blocks.
- All combinational blocks are supposed to finish the computations of the current clock cycle before the start of the next clock cycle.

Power Consumption

- Why low power?
  - High performance and integrity of VLSI circuits
  - Popularity of portable devices
- Power consumption in CMOS circuits
  - Dynamic power dissipation (used to be dominant)
    - Charging and discharging capacitors
  - Short-circuit power dissipation
  - Leakage power dissipation (increasingly larger)
- Dynamic power dissipation



$$P_{dynamic} = \alpha \cdot C_{phy} \cdot V_{dd}^2 \cdot f_{clk}$$

where α: switching activity, C_{phy}: physical capacitance, V_{dd}: supply voltage, f_{clk}: clock frequency

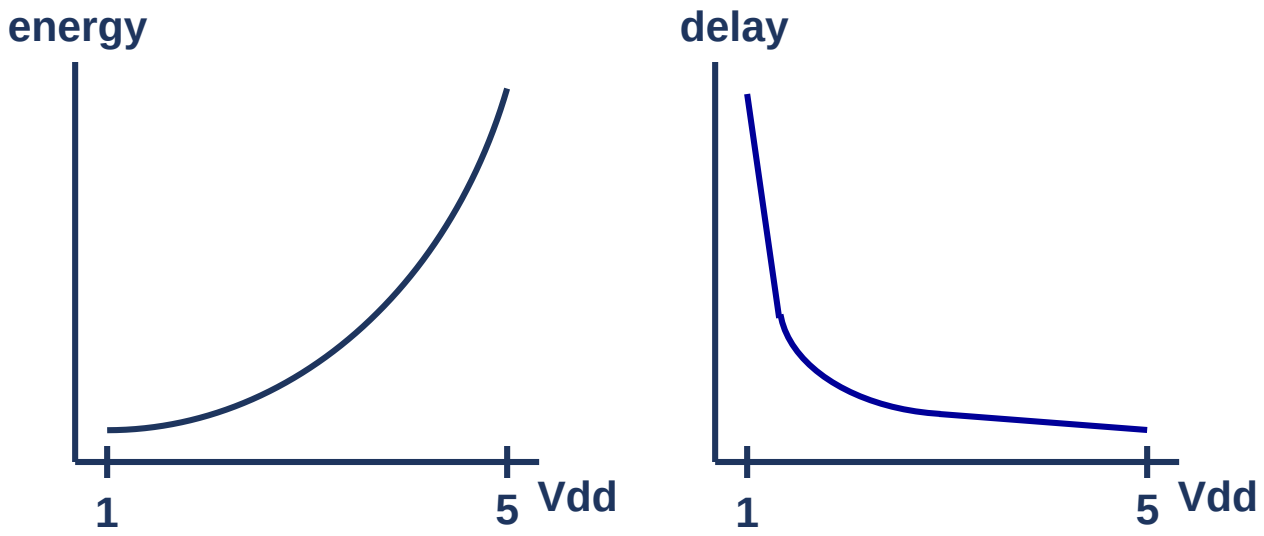
Power Consumption

- Supply voltage reduction
  - Quadratic effect of voltage scaling on power

$$P_{dynamic} = \alpha \cdot C_{phy} \cdot V_{dd}^2 \cdot f_{clk}$$

5V → 3.3V ➔ 60% power reduction

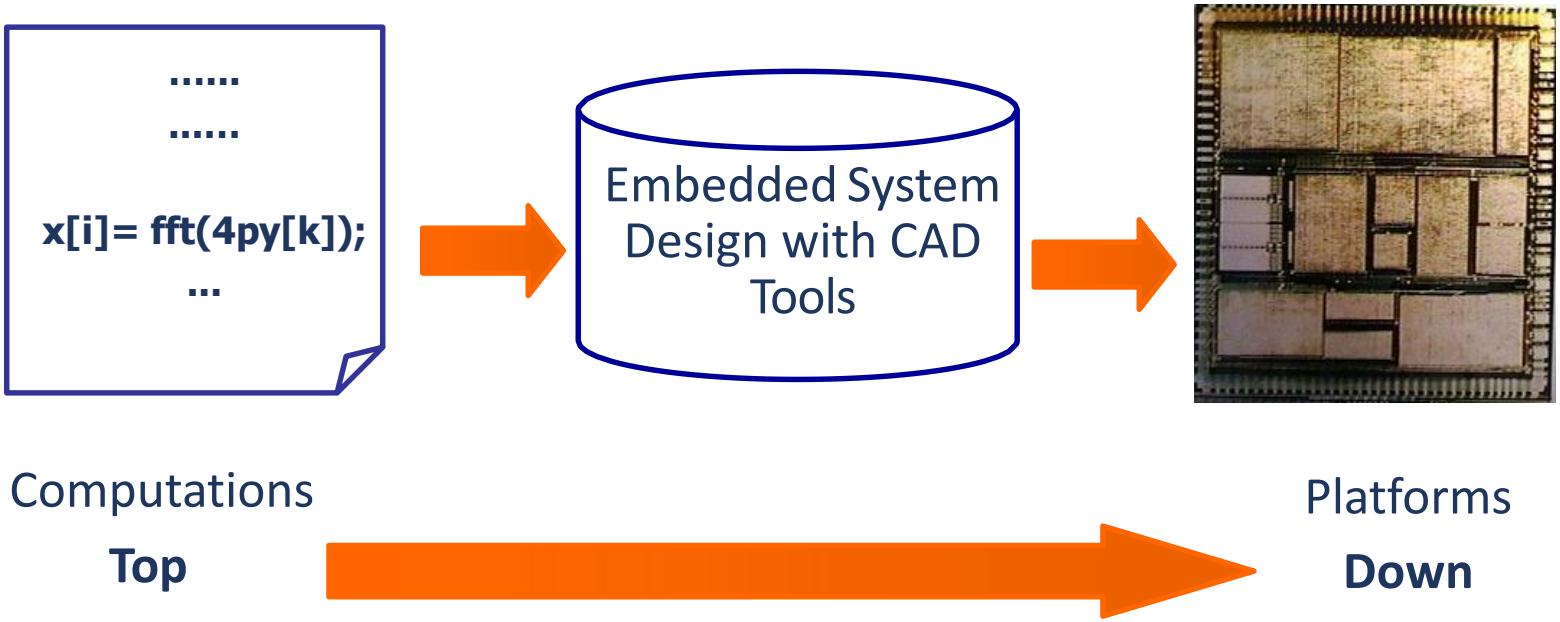
  - Supply voltage reduction ➔ increased latency



Who Contributes to Embedded System Designs

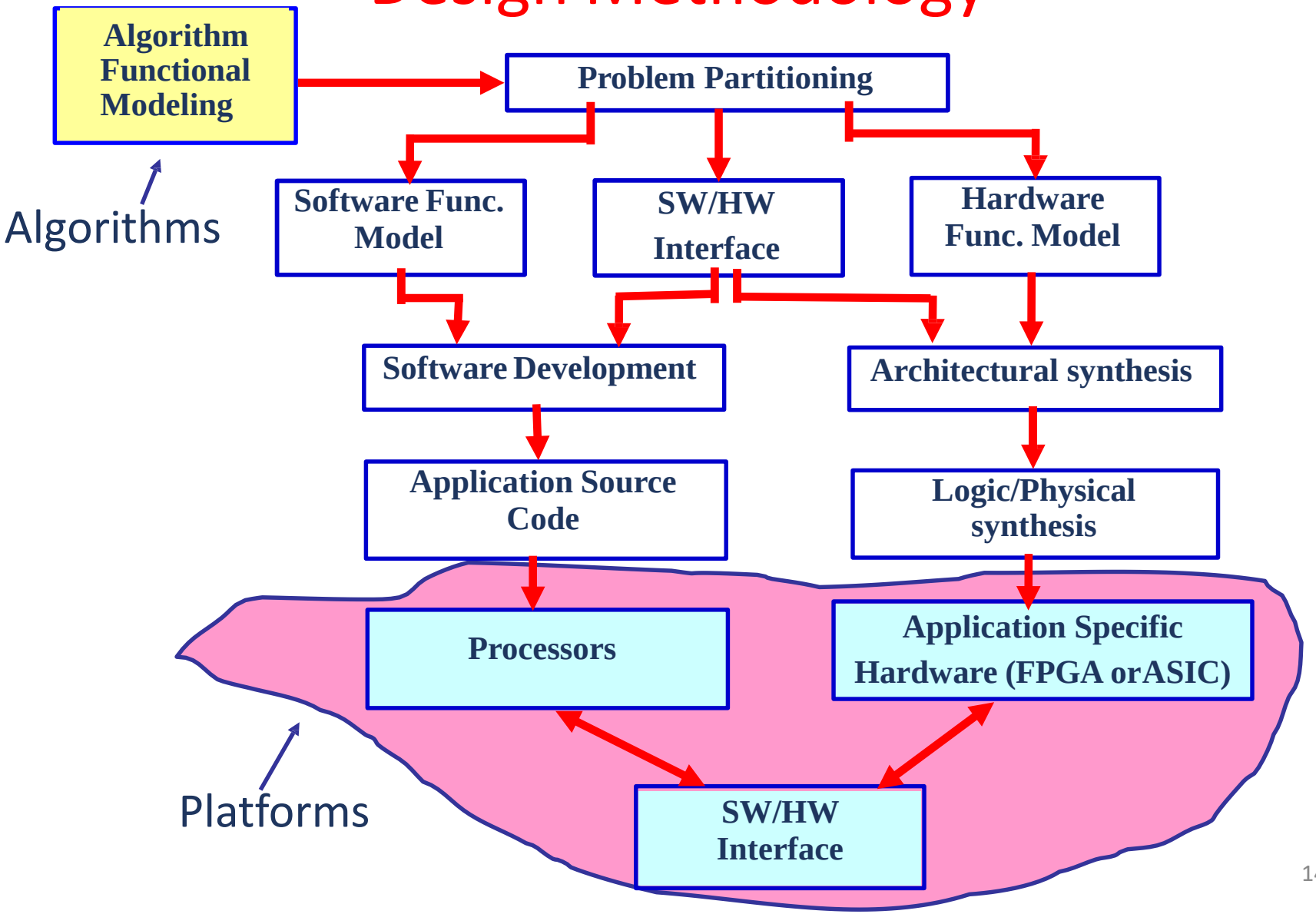
- Application algorithm developer
  - e.g., telecommunication, multi-media researcher
- Computer-Aided Design (CAD) tool developer
  - e.g., Synopsys, Cadence companies
  - Potential research field
- IC designers working at different levels
  - e.g., IC design group in Infineon, Broadcom and HP
  - Industry field
- Test engineer
  - Both research and Industry field

Embedded Systems Implement Computations on Platforms



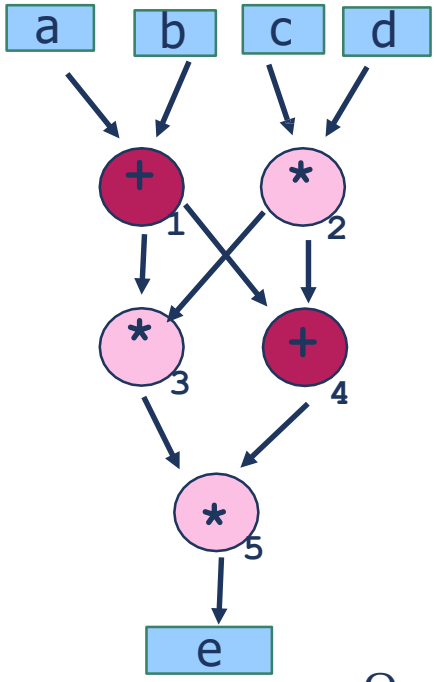
We will examine the ***design methodologies*** to implement computations (algorithms) on platforms. We temporarily forget analog design for a moment.

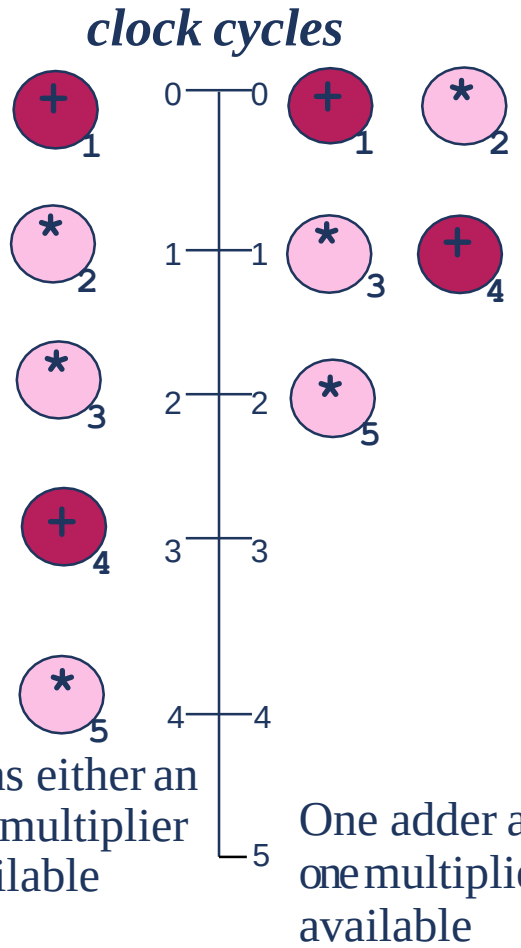
Simplified and General Embedded System Design Methodology



An Example to Start

```
tmp0 = a + b;
tmp1 = c * d;
tmp2 = tmp0 * tmp1;
tmp3= tmp0 + tmp1;
e = tmp2 * tmp3;
```



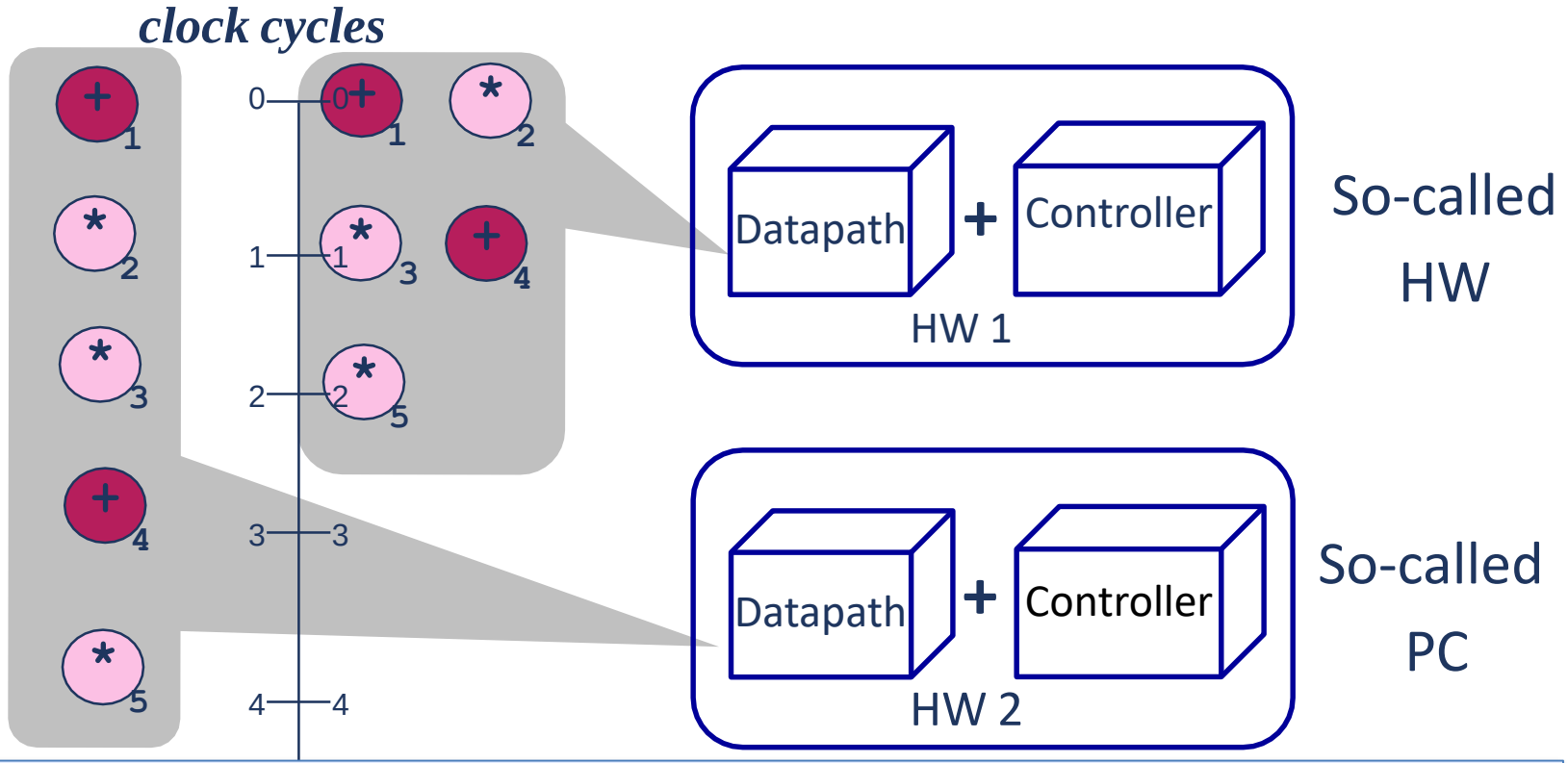


One ALU as either an adder or a multiplier is available

One adder and one multiplier are available

Algorithm Code | Data Flow Graph | schedule 1 | schedule 2

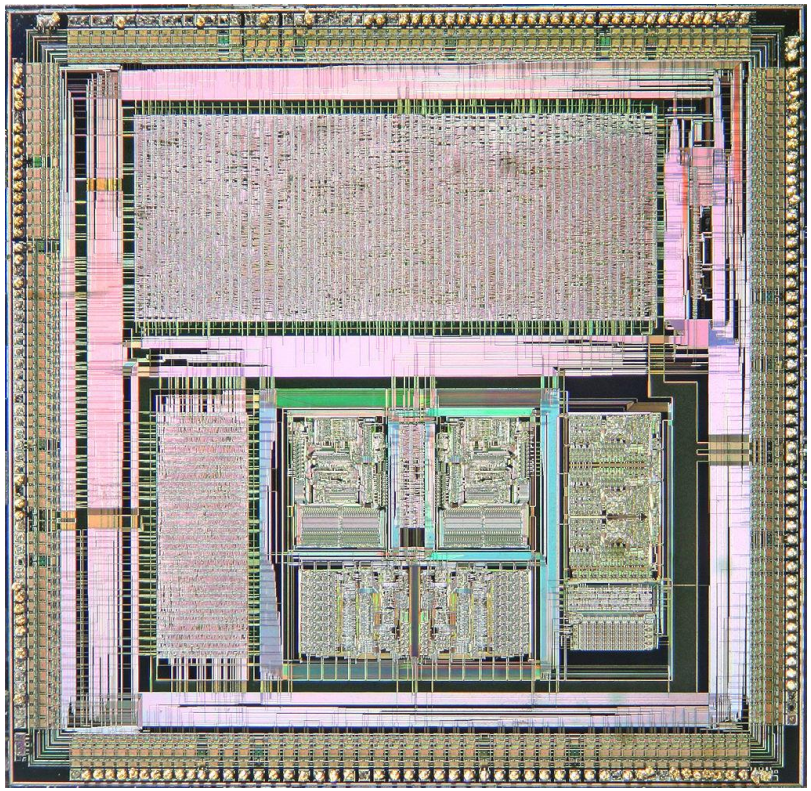
Datapath and Controller



- Datapath implements operators, decides the area and speed that a design can achieve.
- Controller decides which operator of a datapath should work at specific cycle according to schedules.
- Embedded system design is actually to design the datapath and controller.

ASIC

- Application-specific integrated circuits



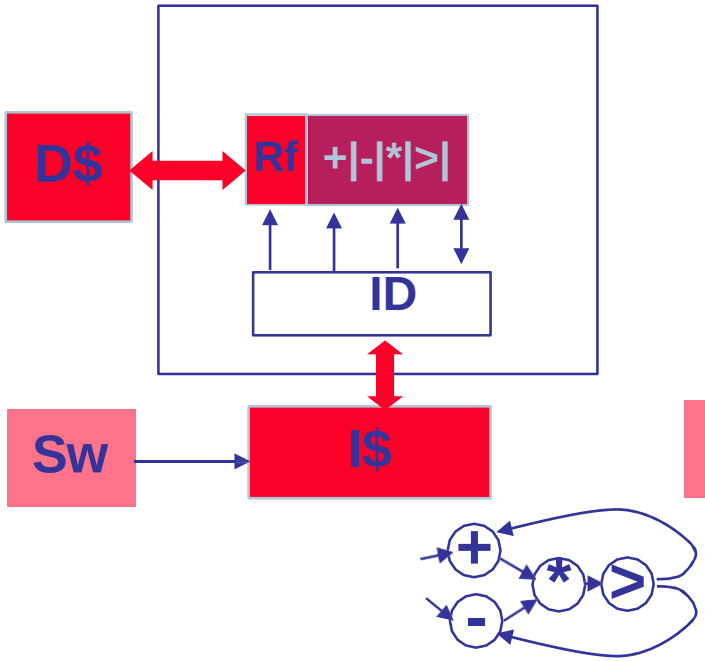
A custom ASIC (486 chipset) showing gate-based design on top and custom circuitry on bottom (Source: Wikipedia)

ASIC

- Application-specific integrated circuits
- De Morgan's theorem
  - Theoretically we only need 2-input NAND or NOR gates to build anything
- ASIC is good, but
  - High risky and expensive to design and manufacture
    - Suitable for very high-volume mass production
  - Permanent circuitry
    - Once designed, not changeable

Three Kinds of Embedded System Implementation Choices

Processor



Programmable

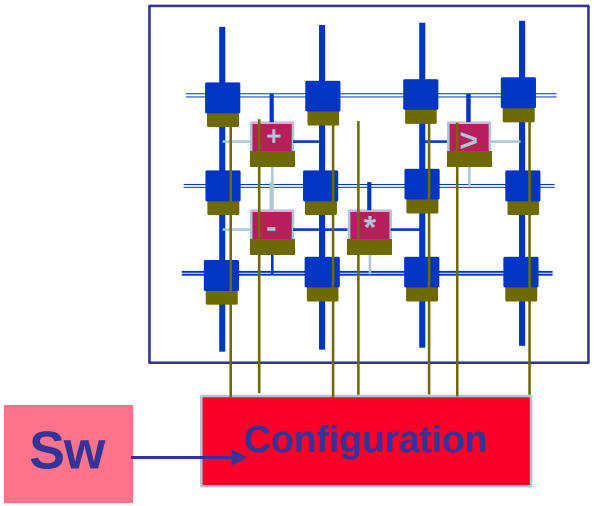
Sequential

Instruction flow (cycle)

Transfer bottleneck

Power: 100

Reconfigurable FPGA



Configurable

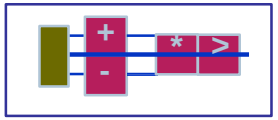
Parallel wired algorithm

"Program" flow (occasionally)

Distributed data

10

ASIC



No wiring

No configuration

overhead

1

Why Use Reconfigurable Hardware?

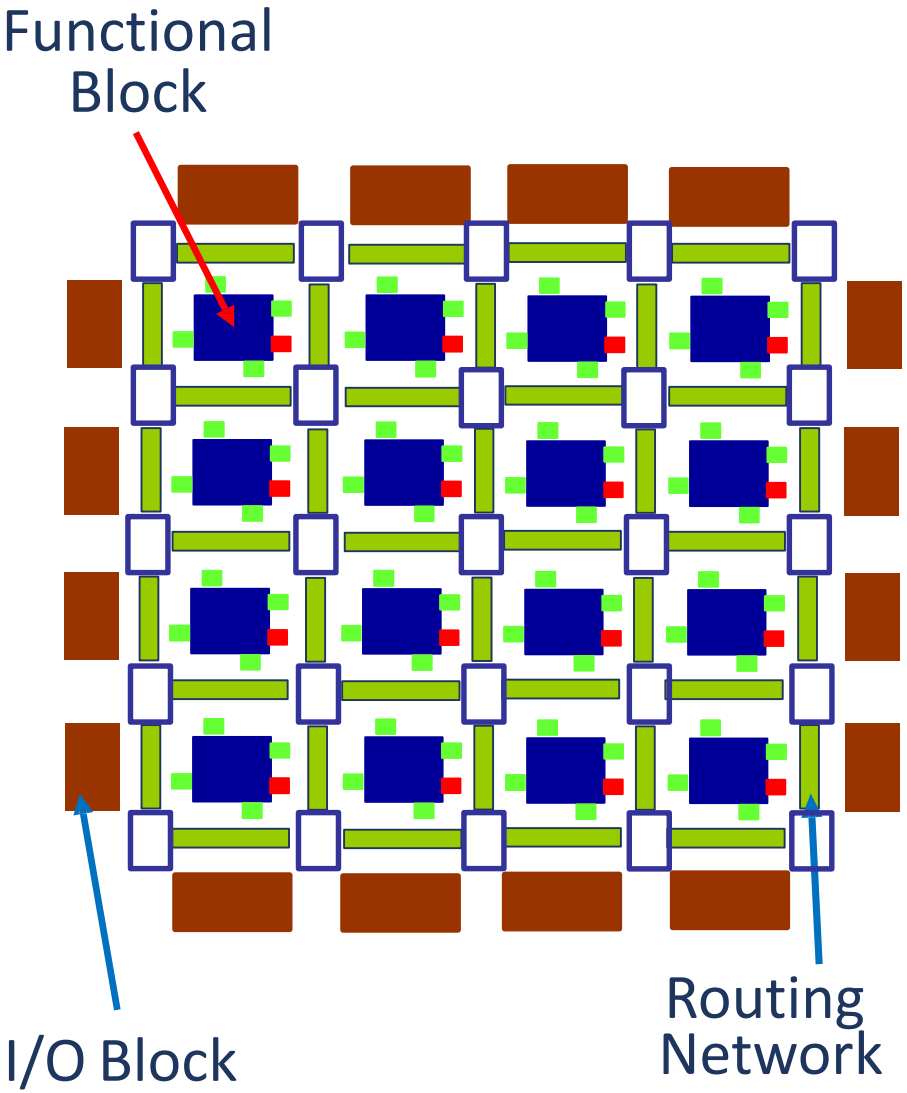
	Processor	FPGA	ASIC
Performance	Low	Medium	High
Flexibility	High	High	Low
Power	High	Medium	Low

Why FPGAs?

- Combine flexibility with performance.
- Shorter time-to-market and longer time-in-market.

Architecture of FPGA

Simplified FPGA Architecture

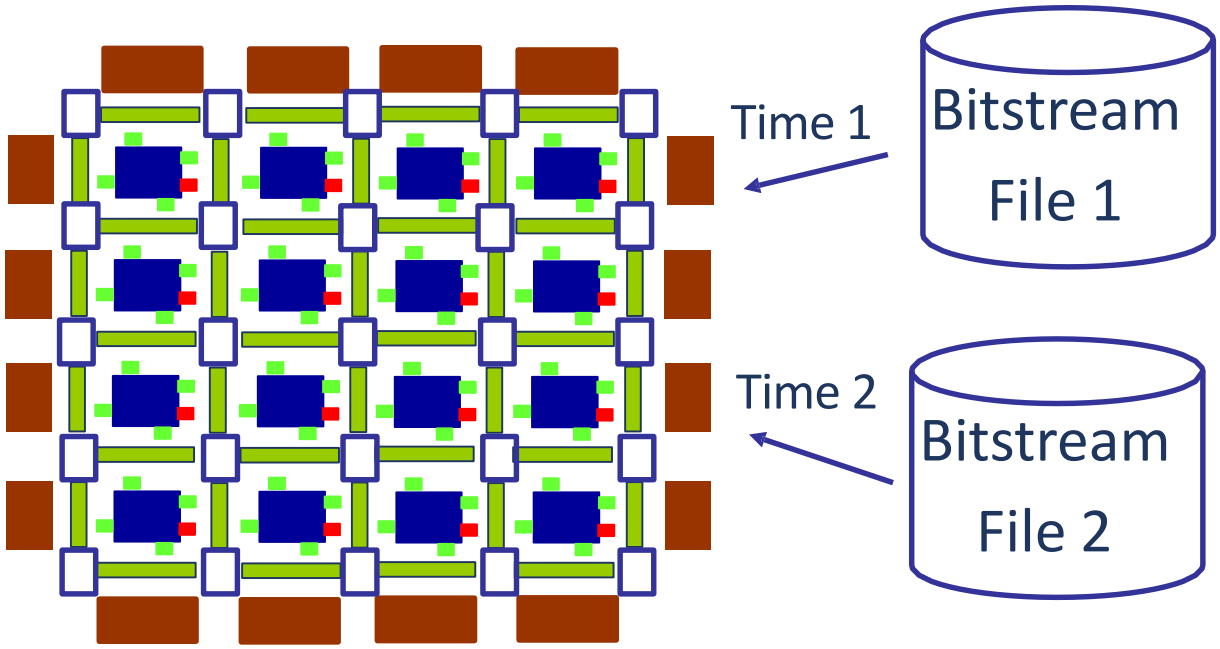


All the three FPGA components can be re-programmed with configurations to implement application-specific digital circuits.

For example,

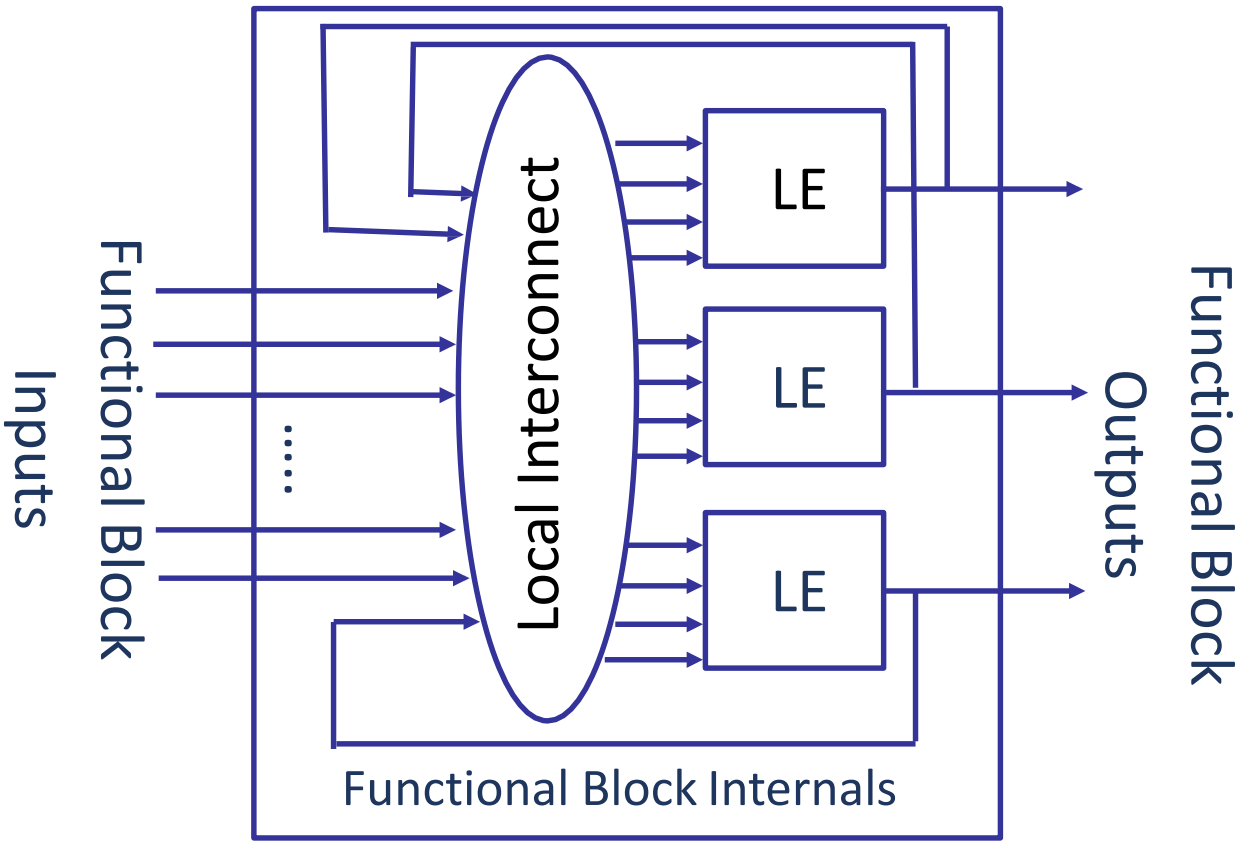
- each functional block can be programmed to implement a small amount of digital logic of a design;
- the routing network can be programmed to implement the design specific interconnection pattern;
- I/O blocks can be programmed to implement the input and output ports according to design requirements.

FPGA Reconfiguration



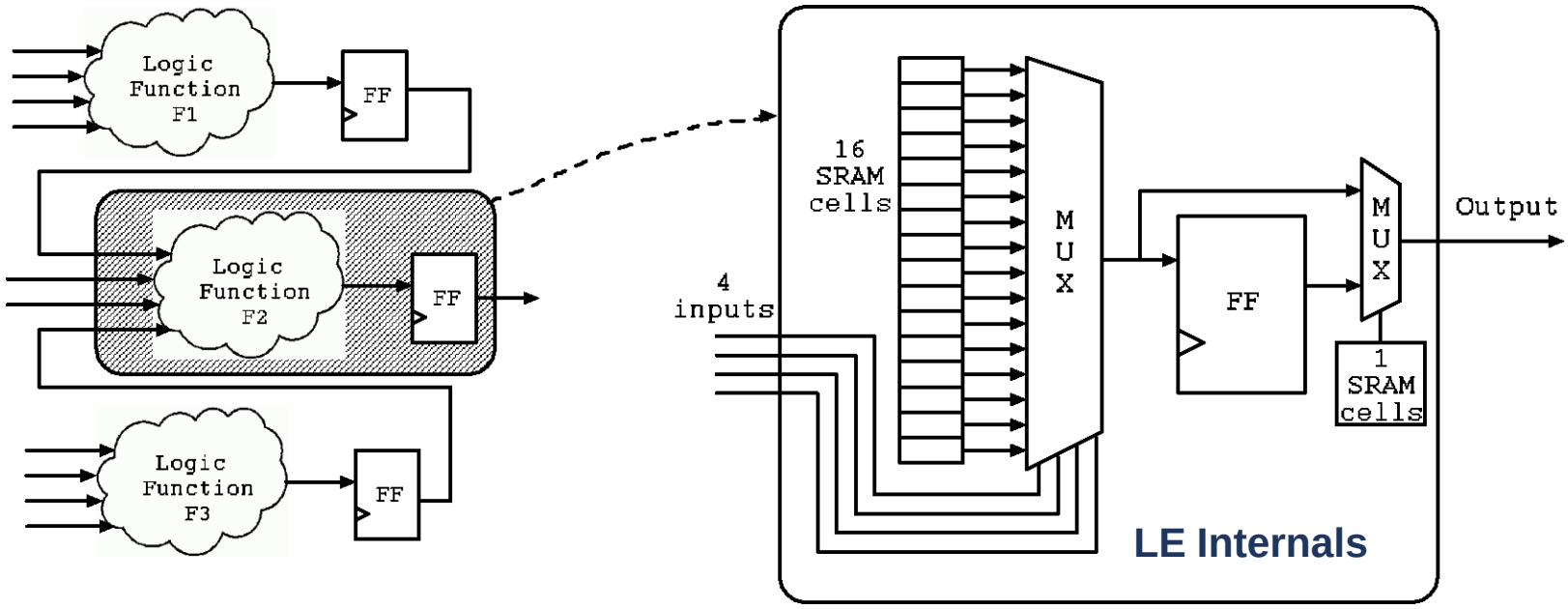
- All the programming information for the three FPGA components is stored in a configuration file. The configuration file for a FPGA is often called a *bitstream* compared to a binary executable for a processor.
- Once a bitstream for a digital logic design is downloaded to a FPGA, the FPGA is programmed to implement the design.
- By providing different bitstreams, a single FPGA can be re-programmed to implement different designs at different times.

FPGA Functional Block



- FPGAs use the Look-Up Table (LUT) type of functional block.
- A functional block is normally made of one or several logic elements (LE).
- Functional blocks differentiate from each other mainly in terms of the input size of an LE and the number of LEs in a functional block.
- State-of-the-art FPGAs normally use 4-input LEs.

FPGA Logic Element

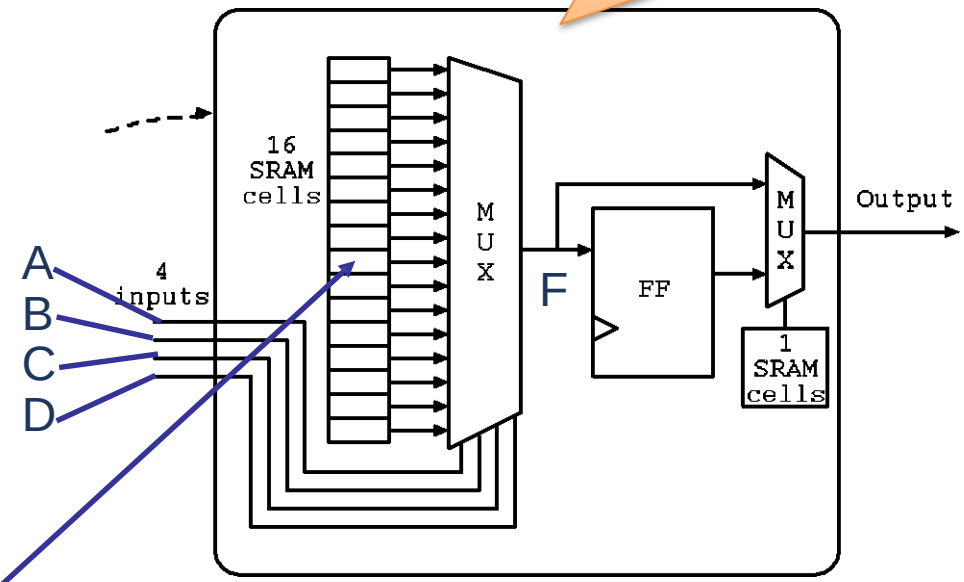


- One LE consists of a 16 SRAM cell **Look-Up Table (LUT),** and a flip flop (FF).
- The 16 SRAM cells LUT stores the truth table of any 4-input logic function. Thus it can implement any 4-input logic function.
- The FF implements the storage element in a sequential circuit.

LUT Content

How many functions can a 4-input LUT represent?

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

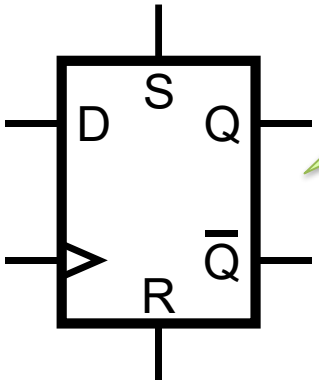


$$F = (A \,\&\, B) \mid (C \,\&\, D)$$

- The 16 SRAM cell LUT stores the output column of the truth table of the F function.
- The 4 inputs A, B, C and D will determine which bit the F value is for the current values of A, B, C and D.

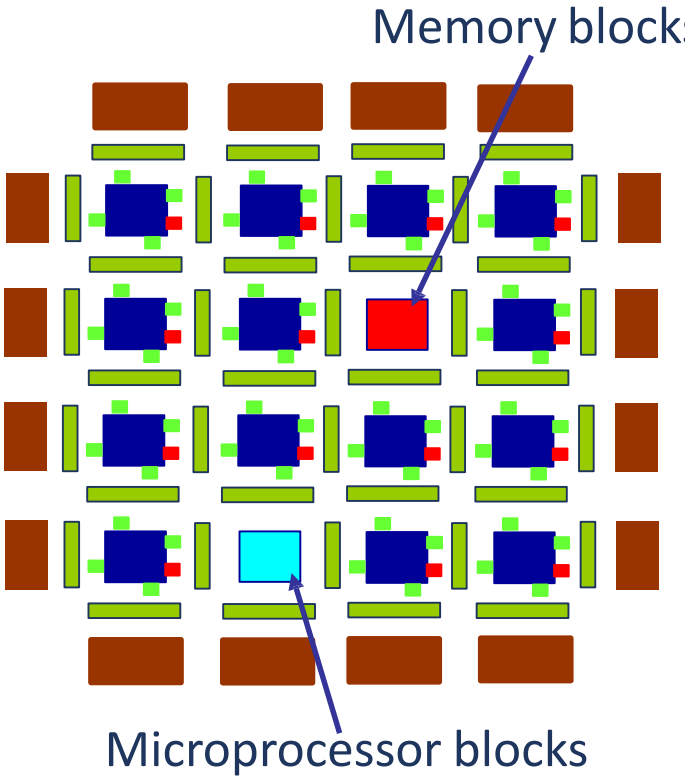
FF

- D Flip Flop
  - A data storage element
  - When FF sees a rising edge of the clock, it registers the data from the Input D to the Output Q.



How long can a D Flip Flop hold current data?

Additional Computational Resources

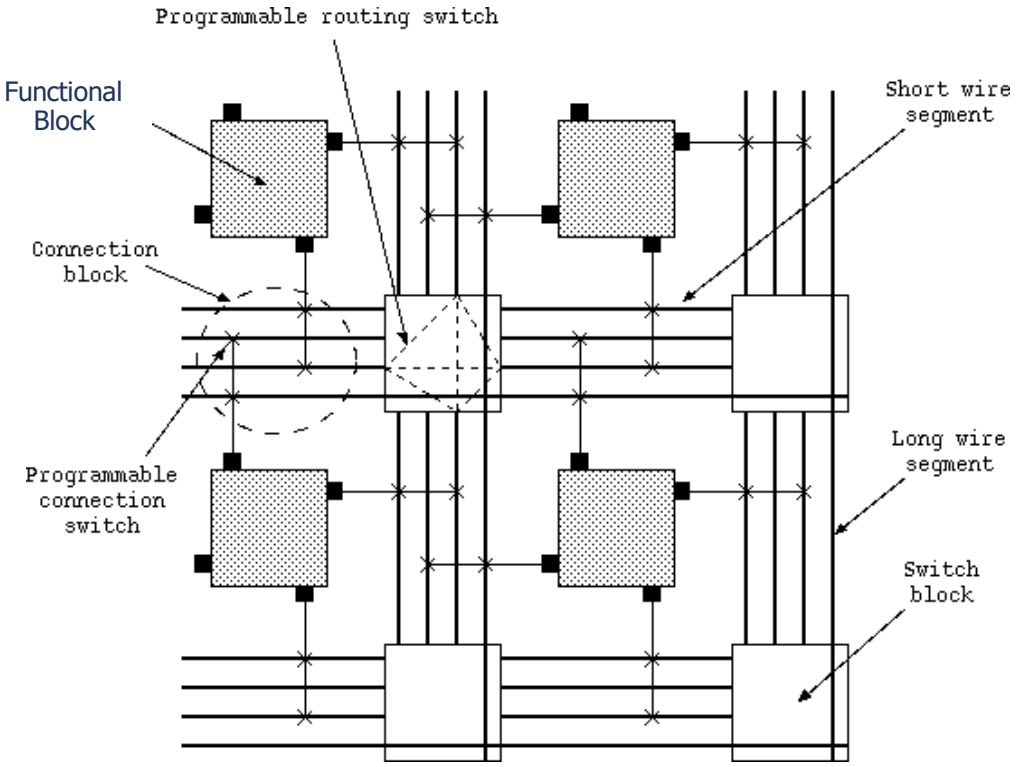


- Besides the LEs, some functional blocks in different FPGAs have architecture specific features to improve the performance when implementing arithmetic functions.
- These architecture specific features include carry logic, embedded memory blocks, multiplier and other hard cores.
- Hard cores generally implement functions efficiently compared to FPGA functional blocks.

Routing

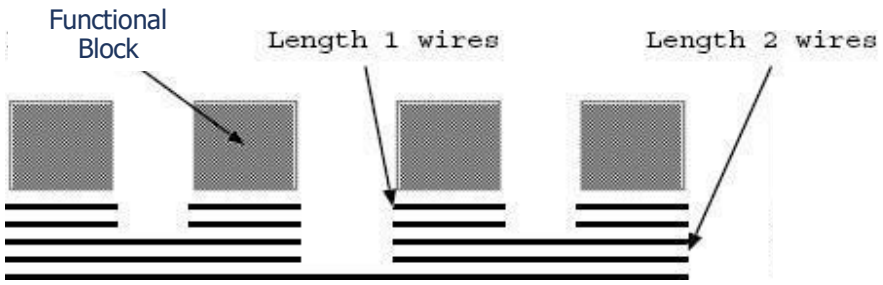
- Routing means interconnecting
  - Through programmable wires and switches
  - Between functional blocks, and between I/O blocks and functional blocks
- Routing is a challenging problem
  - Routing technique used in an FPGA largely decides the amount of area used by wire segments and programmable switches, as compared to area consumed by functional blocks.
  - Inferior routing may lead to congestion or failure of signals.

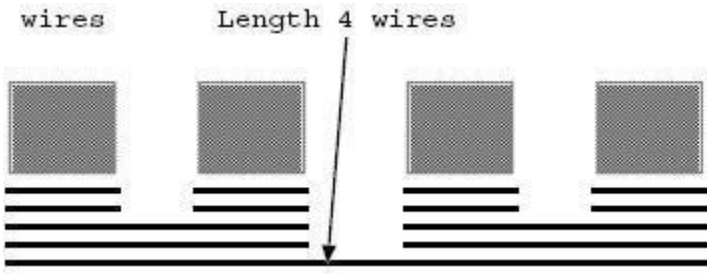
FPGA Routing Architecture



- A functional block input or output pin can connect to some or all of the wiring segments in the channel adjacent to it via a *connection block* of programmable switches.
- At every intersection of a horizontal channel and a vertical channel, there is a *switch block*. It is a set of programmable switches that allow some of the wire segments incident to the switch block to be connected to others.
- By turning on the appropriate switches, short wire segments can be connected together to form longer connections.

FPGA Routing Wires



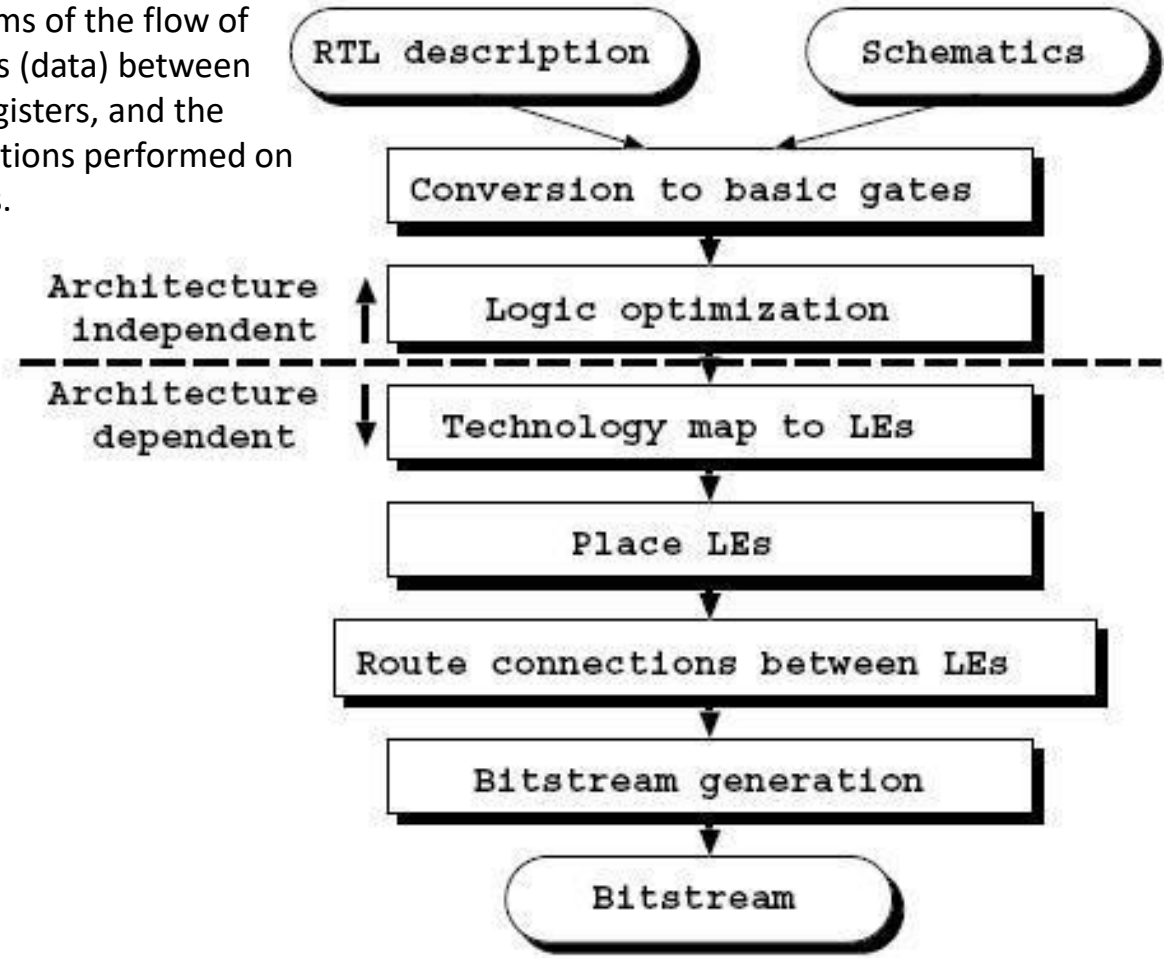


- Some FPGAs contain routing architectures that include different lengths of wires.
- The length of a wire is the number of functional blocks it spans.
- Left figures show wires of length 1, 2 and 4.
- Long wires introduce shorter delays for long interconnections since fewer switch blocks will be passed.

Design Tools for FPGA

FPGA Design Flow

Register-transfer level (RTL) models a synchronous **digital circuit** in terms of the flow of digital signals (data) between hardware registers, and the logical operations performed on those signals.



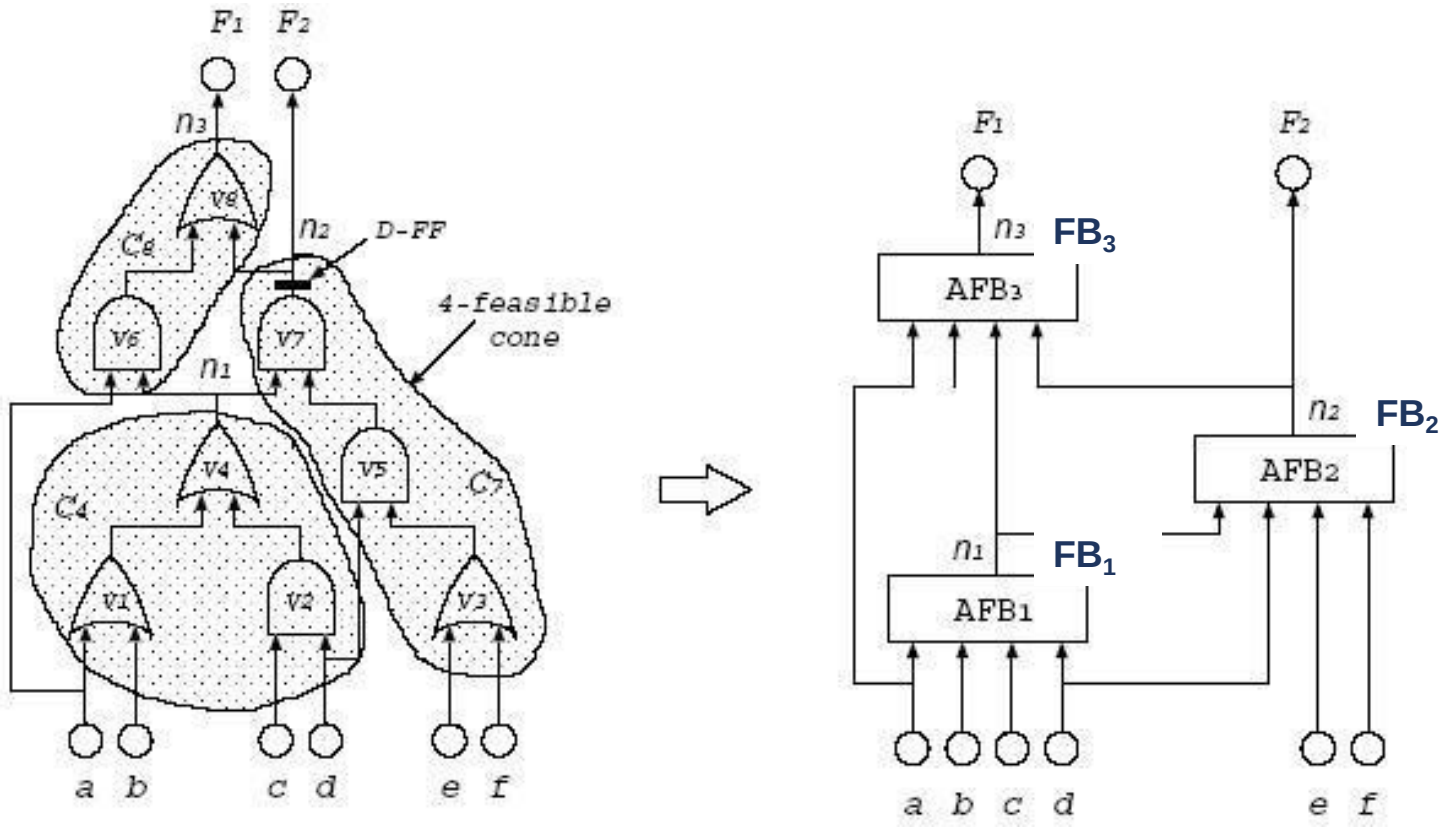
A visual representation of design

Time Profile for Design Flow Steps

CAD Steps	Time (second)	Percentage (%)
Logic Optimization	1,859	43.14
Technology Mapping	156	3.62
Placement	854	19.82
Routing	1,341	31.12
Bitstream Generation	99	2.30

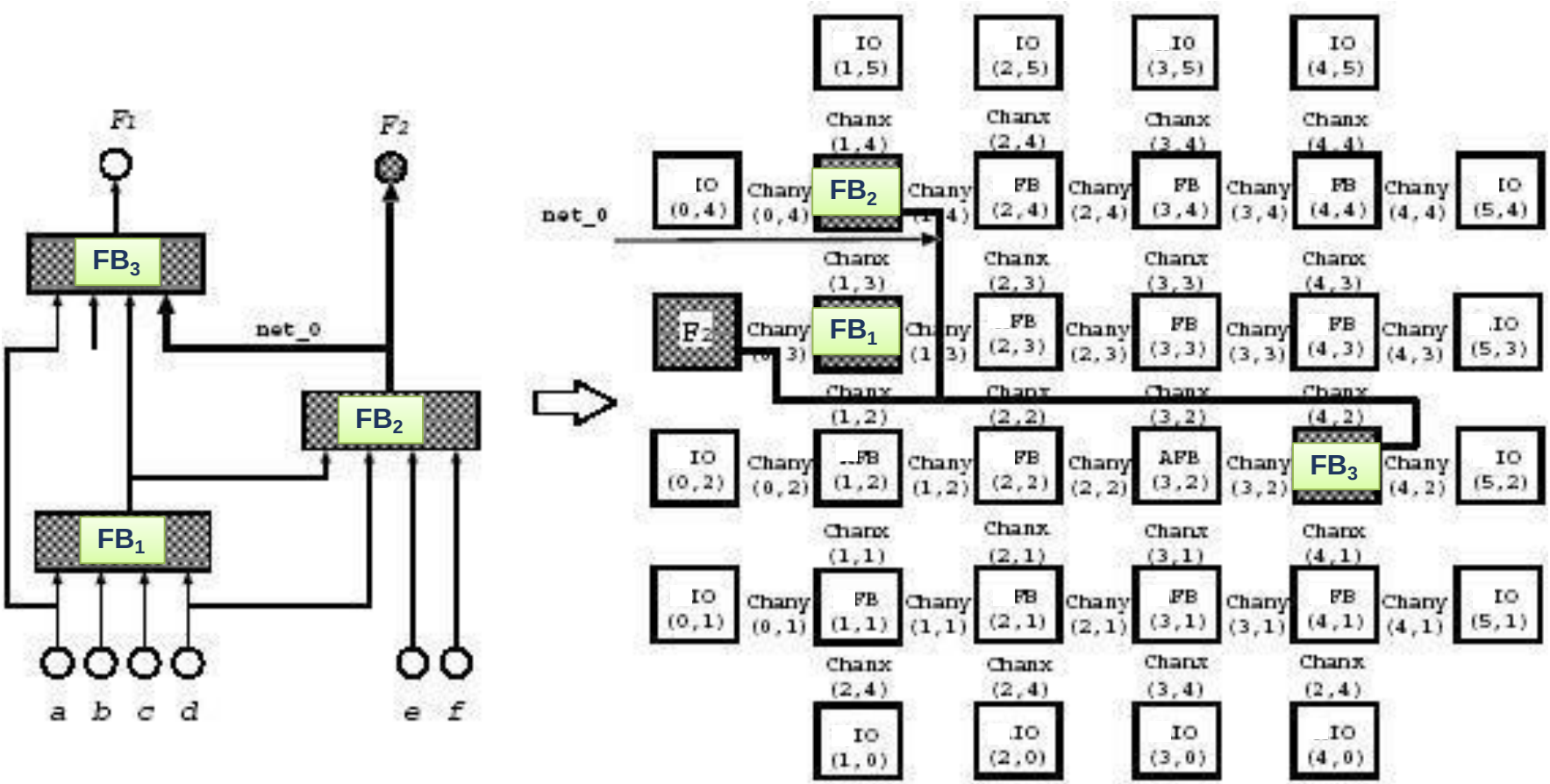
Logic Optimization and routing steps normally consume the major part of the design flow time.

FPGA Technology Mapping



Technology mapping step restructures the primitive logic gates, generated from the logic optimization step, into sets of 4-input functional blocks (we assume one functional block contains only one logic element in this example).

FPGA Placement and Routing



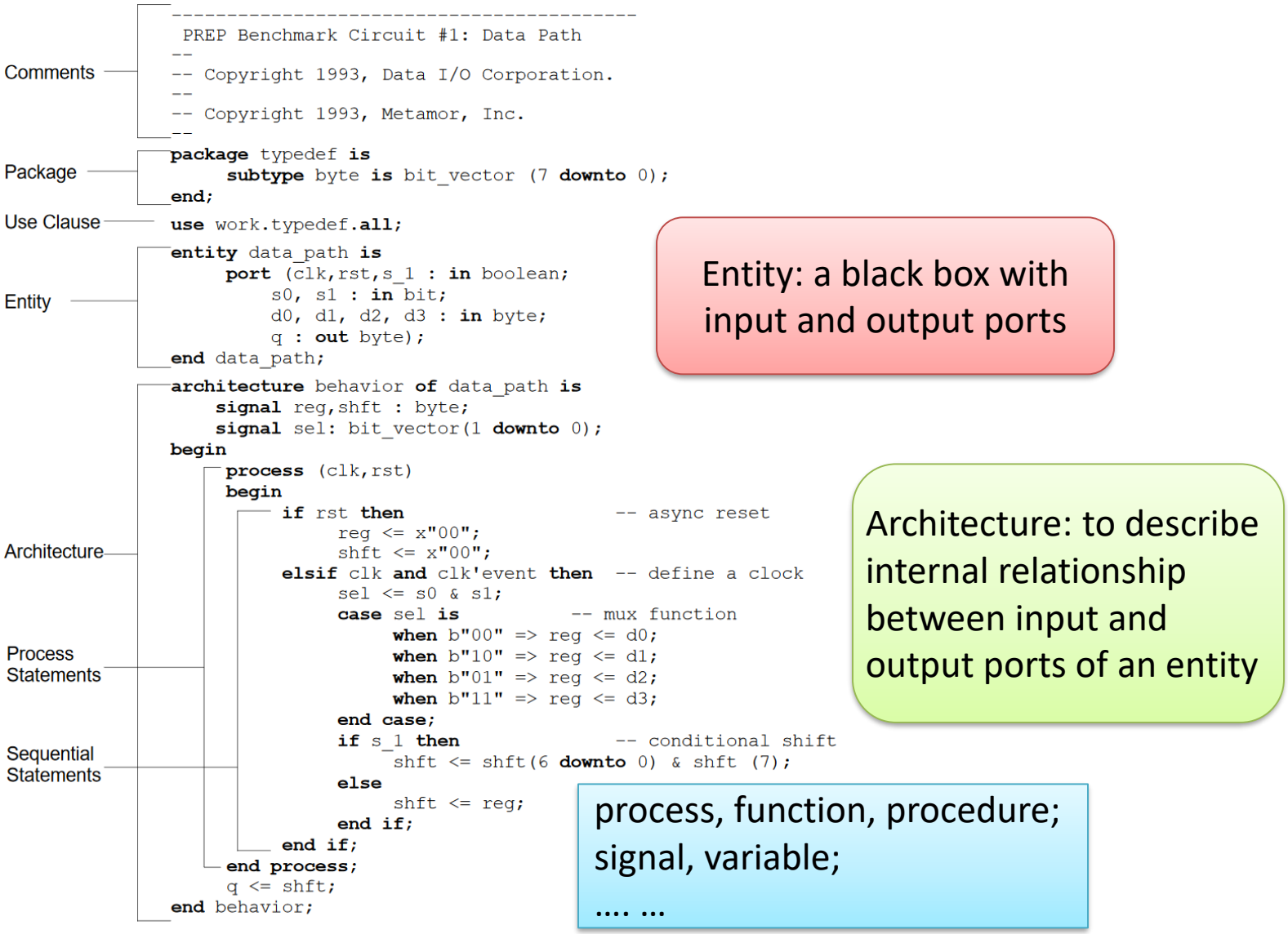
The placement step finds physical locations for functional blocks, while the routing step finds physical routes for logic connections.

HDL

HDL

- A designed circuit can be specified through a schematic diagram, or an HDL program
- HDL: hardware description language
  - For both ASIC and FPGA
- Two common HDLs
  - VHDL
  - Verilog
- From HDL to Bitstream
  - Describe your design using HDL programs
  - Use tools to synthesize, configure and test with FPGA

A Sample of VHDL Program



Entity: a black box with input and output ports

Architecture: to describe internal relationship between input and output ports of an entity

process, function, procedure;
signal, variable;
…. …

HDL vs. Software PL

Hardware	Software
Concurrent execution of tasks. This demands all tasks and events to operate in coherence with a timing reference signal called clock	Sequential execution of tasks and instructions. There is no concept of synchronization to clock reference
Very fast execution. Functional timing in nanosecond scale units is achievable in hardware. And therefore, time critical functions are designed to be in hardware	Slow execution. Minimum timing resolution is 100s of microsecond
Can be parallel	Sequential though it can appear to be parallel for the user
Physical and costs are exorbitant if it has to be redone	Can be recompiled
Need to be first time success.	Can be corrected and recompiled without much effort
Hardware can be one time developed as platform and reused for lifetime if the functionality is the same	Can be redone easily.
Development from paper specification to physical system on chip	Need processing hardware platform for sw development
Need to verify fully imagining all scenario ahead of fabrication and hence verification and validation are unavoidable	Verification is necessary to prove the intent of the design but in the case of minor defects, it can be corrected.

Source: https://link.springer.com/book/10.1007%2F978-3-030-23049-4

Conclusion

- Ubiquitous embedded Systems
- FPGA is effectual for embedded systems
  - Now it is also deployed in other environments
    - e.g., data centres
- FPGA
  - Components
- FPGA differs from software programming
  - It has unique characteristics and methodologies
  - HDL