University of San Carlos | Department of
COMPUTER ENGINEERING

CpE 3101L – Introduction to HDL

Unit 1: Design Flow of Digital Systems

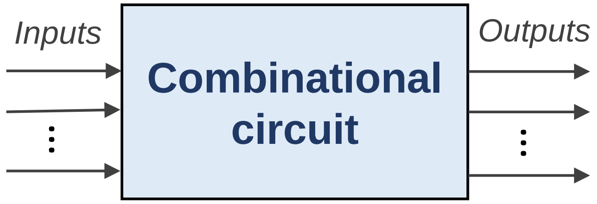
Lecture 1

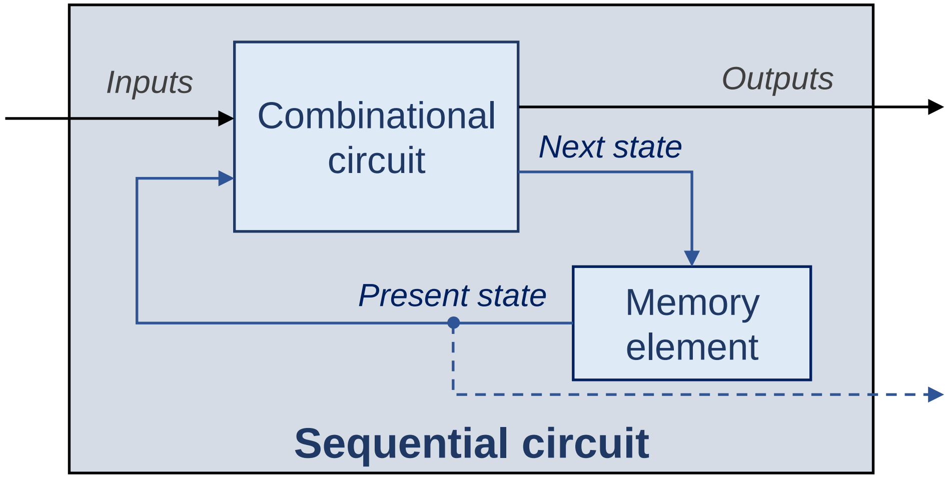
Outline

- Design Methods for Electronic Systems
- Hardware Platforms
- Programmable Logic and FPGAs
- Design Flow in Modern Digital System Design
- Hardware Description Languages (HDLs)
- Introduction to Verilog HDL
- Introduction to Intel® Quartus® Prime Lite Edition

Brief Review of Logic Design

- Combinational Logic
  - No memory
  - Present outputs depend ONLY on the present inputs
- Sequential Logic
  - Has memory (flip-flops, etc.)
  - Present outputs depends on present inputs AND on the past sequence of inputs *(states)*

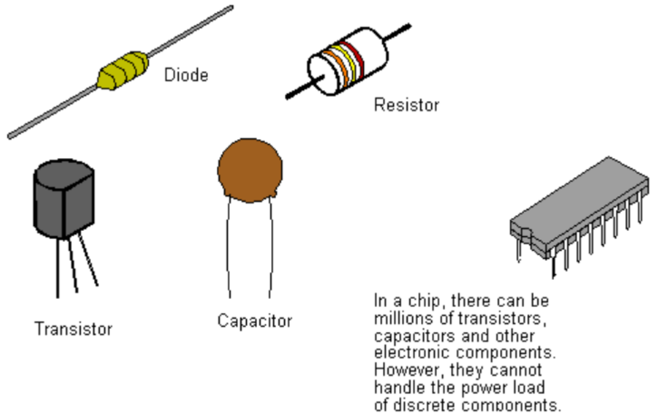




Design of Digital Circuits

- Functionality of circuit (through standard methods)
  - Truth tables and Boolean functions *(SOP/POS/other forms)*
  - Other combinational logic components: *adders, comparators, decoders, multiplexers, etc.*
  - State tables, state diagrams, and flip-flop equations
  - Other sequential logic components: *flip-flops, registers, shift registers, counters, etc.*

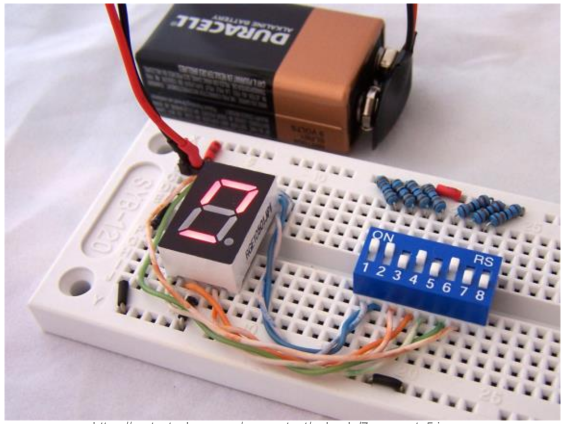
- Hardware Implementation
  - Discrete electronic components
  - Integrated circuits (ICs)
    - Logic gate ICs (TTL logic family)


Diode
Resistor
Transistor
Capacitor
In a chip, there can be millions of transistors, capacitors and other electronic components. However, they cannot handle the power load of discrete components.

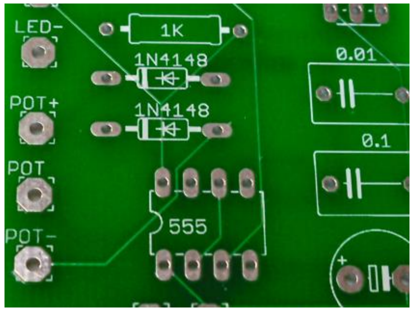
https://i.pcmag.com/imagery/encyclopedia-terms/discrete-component-discrete.fit_lim.size_1200x99999.gif

Construction of Digital Circuits

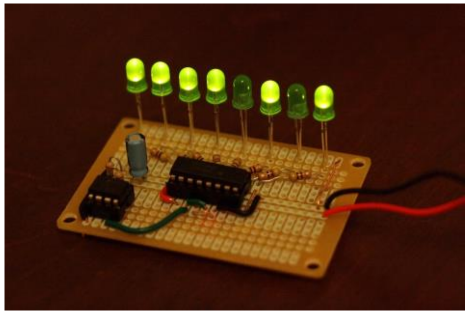
- Purpose: To verify the functionality of a digital circuit through:
  - Standard methods *(logic circuit analysis)*
  - Computer-aided design (CAD) or Electronic design automation (EDA) tools
    - Circuit simulation software
    - Logic simulation and testing
  - Prototyping
    - Process of testing "an idea"
    - Creating a preliminary model
    - Test a real, working system


https://protostack.com.au/wp-content/uploads/7segment_5.jpg

- Circuit prototyping methods
  - Breadboard
  - Printed circuit board (PCB)



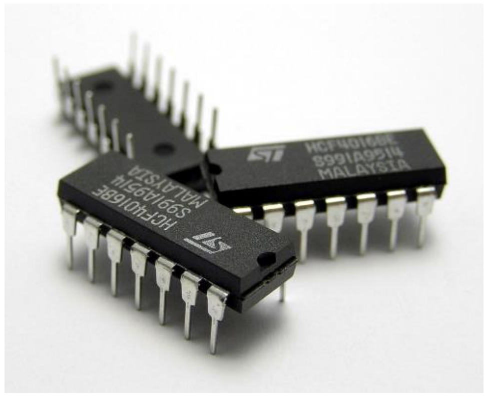
https://spectrum.ieee.org/image/MzAzODk3NA.jpeg


https://live.staticflickr.com/1262/1440357613_e468cd0899_b.jpg

Design Methods for Electronic Systems

- IC technology has provided a variety of implementation formats:
  - Defines the technology to be used *(TTL, CMOS, etc.)*
  - Defines how the switching elements or components are organized *(regularity and granularity)*
  - Defines how the system functionality will be materialized *(use of discrete components, logic gates, other IC-based components, mixed, etc.)*

- This affects the way systems are designed
- It sets the limits of the system complexity



https://upload.wikimedia.org/wikipedia/commons/thumb/8/80/Three_IC_circuit_chips.JPG/1200px-Three_IC_circuit_chips.JPG

Organization of Design Components

- Regularity
  - Has a strong impact on the design effort
  - Because the reusability of a regular design can be simple
  - But structure may limit its usability and performance
  - Example: *32-bit adder through cascading 2 16-bit adder ICs, etc.*

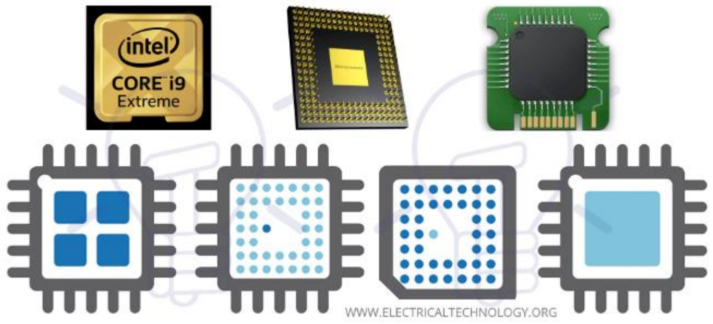
- Granularity
  - Level of functionality in one design object (Example: IC)
  - Affects the number of required design objects and design effort
  - Fine-grain: *Logic gates*
  - Medium-grain: *Arithmetic and logic units (ALUs)*
  - Coarse-grain: *Intellectual property (IP) components – processors, network interfaces, etc.*

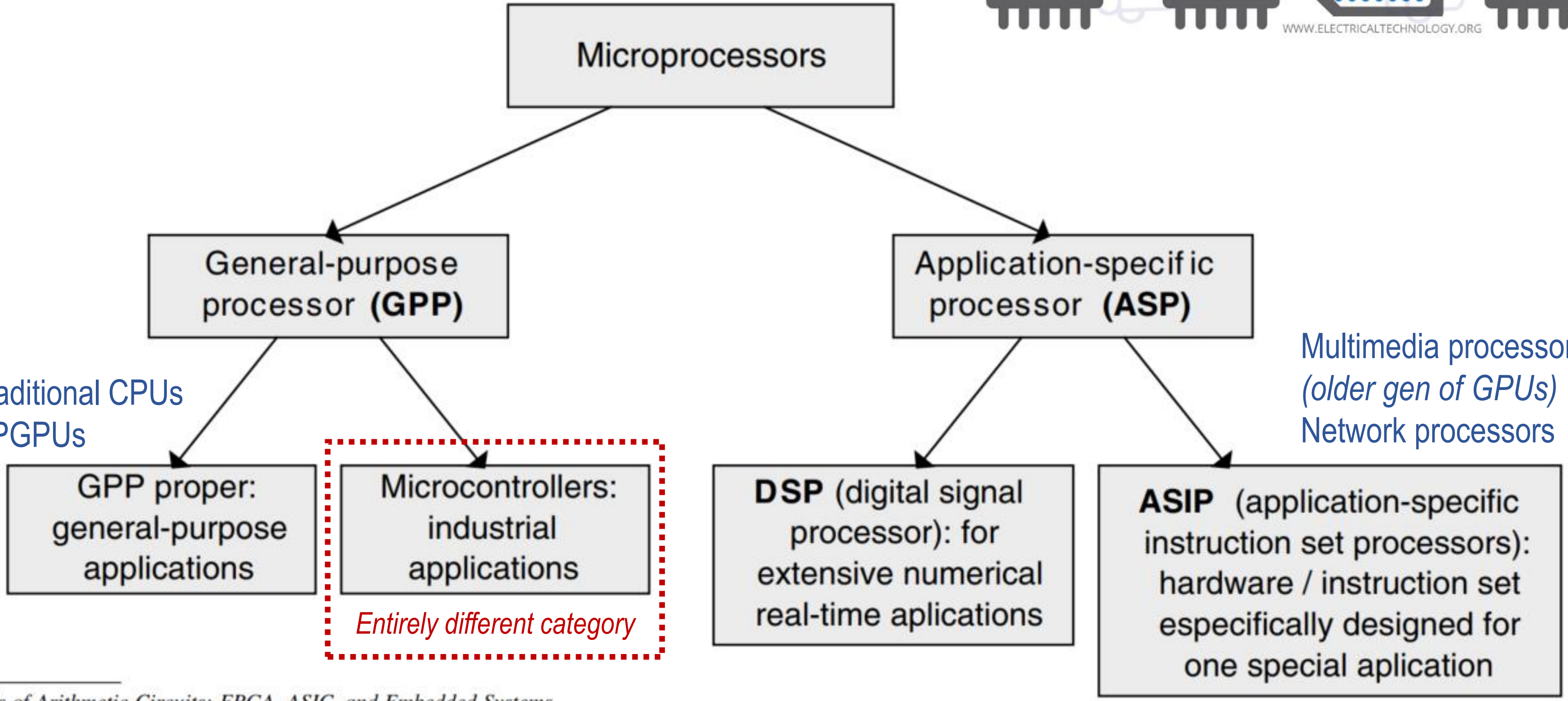
Hardware Platforms

- 3 main structures for implementing its functionality:
- Dedicated systems
  - Structure is fixed at design time
  - Example: *application-specific integrated circuits (ASICs)*
- Programmable systems
  - Architecture is fixed but datapath is configured by instructions in a program (through assembly, C/C++ programming languages)
  - Example: *traditional microprocessor-based computer*
- Reconfigurable systems
  - Structure can be altered by changing configuration data
  - Example: *programmable logic devices (PLDs)*

Programmable Systems

- Instruction set processors
- System-on-chips (SOCs)




Microprocessors
General-purpose processor (GPP)
Application-specific processor (ASP)
Traditional CPUs
GPGPUs
GPP proper: general-purpose applications
Microcontrollers: industrial applications
Entirely different category
DSP (digital signal processor): for extensive numerical real-time aplications
ASIP (application-specific instruction set processors): hardware / instruction set especifically designed for one special aplication
Multimedia processors
(older gen of GPUs)
Network processors

Dedicated Systems

- Application-specific integrated circuits (ASICs)

- Non-standard ICs customized for a particular use rather than for general purpose tasks
- Full-custom and semicustom ASIC

- Advantages:
  - Increased speed
  - Lower power consumption
  - Lower cost *(for mass production)*
  - Better design security *(difficult reverse engineering)*
  - Better I/O control
  - More compact board designs

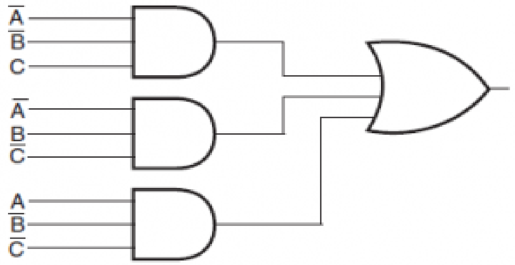
DRAGONMINT
https://www.buybitcoinworldwide.com/img/dragonmint/dragonmint.jpeg

ASIC Bitcoin Miner (Dragonmint 6T)

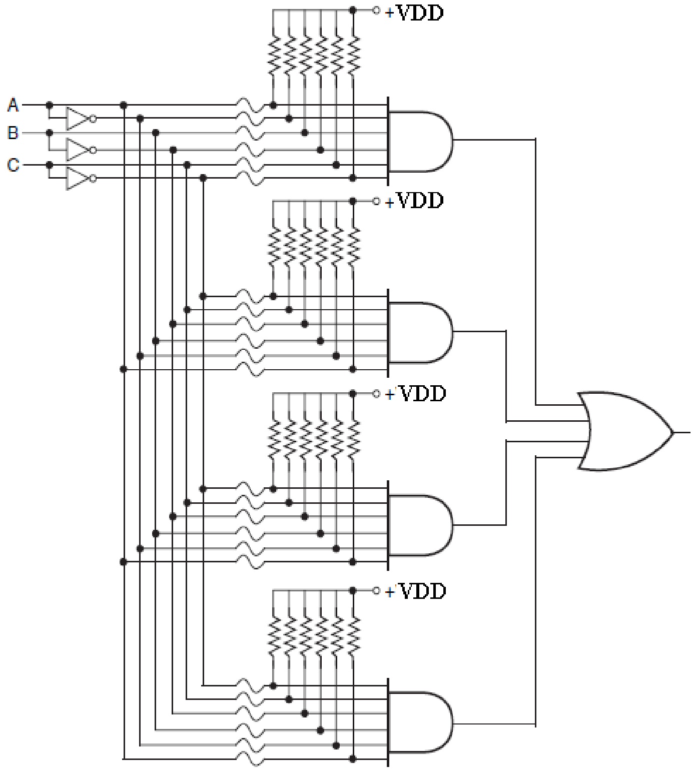
- Disadvantages:
  - Long development periods and turnaround time (TAT)
  - Expensive for low-volume production
  - Very high non-recurring engineering (NRE) cost
  - Design can't be changed *(after production on silicon)*

Reconfigurable Systems

- Categories of logic devices:
- **Fixed** – Standard ICs that perform one or a set of functions



- **Programmable** – Programmable Logic Devices (PLDs)
  - Standard, off-the-shelf parts that can be modified at any time to perform any number of functions (reusable)
  - Also known as **reconfigurable systems**

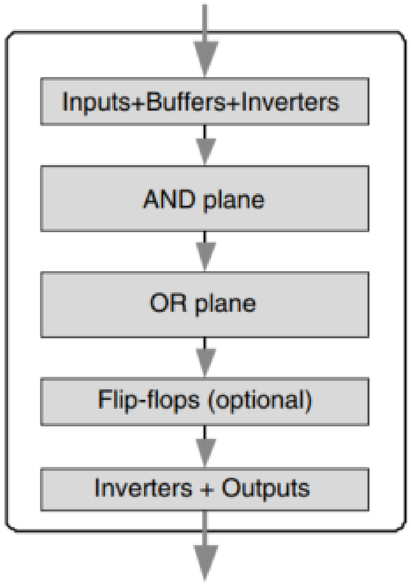


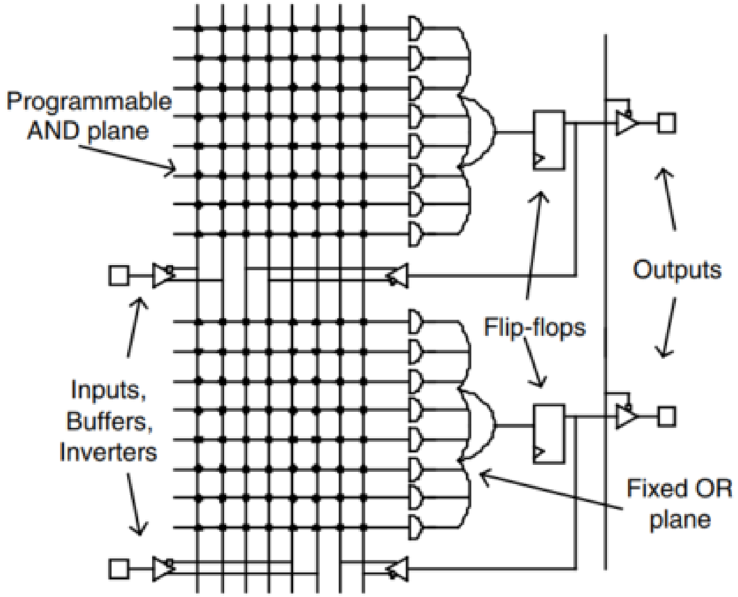
https://www.electronics-tutorial.net/wp-content/uploads/2015/09/PLD2.png

Generic Structure of a PLD

Programmable Logic Devices (PLDs)

- Based on rewritable memory technology
  - To modify the design, the device only needs to be "reprogrammed"
- Made up of a fully connected set of macrocells (some combinational logic and a flip-flop)

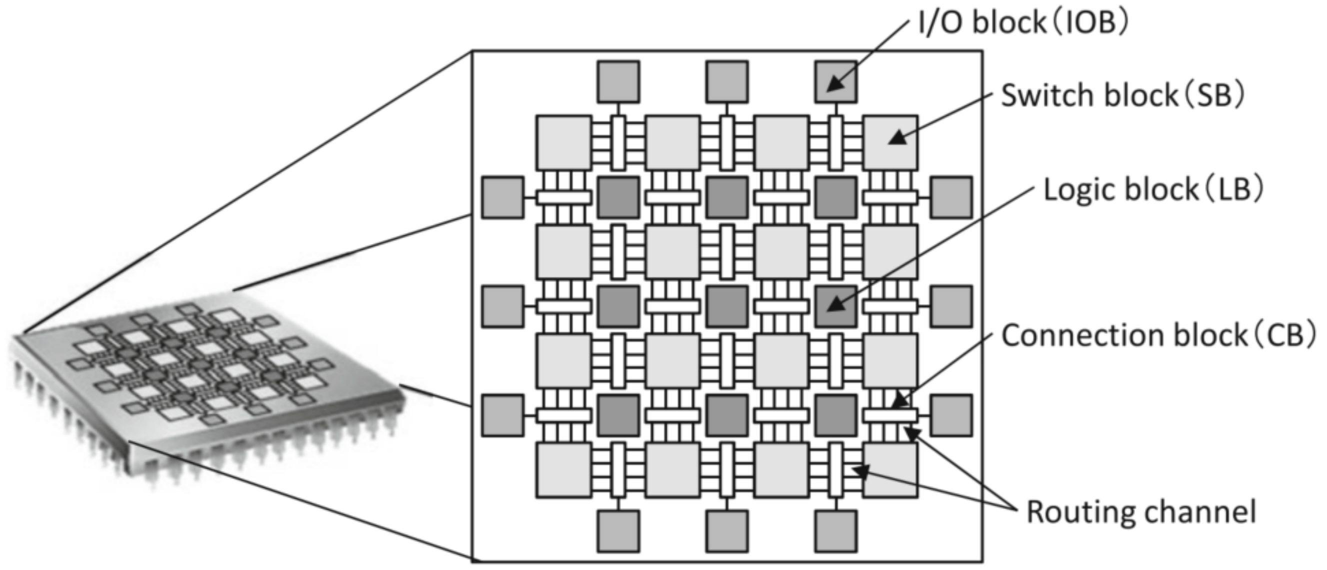




Synthesis of Arithmetic Circuits: FPGA, ASIC, and Embedded Systems
By Jean-Pierre Deschamps, Géry J. A. Bioul, and Gustavo D. Sutter
Copyright © 2006 John Wiley & Sons, Inc.

Field Programmable Gate Arrays (FPGAs)

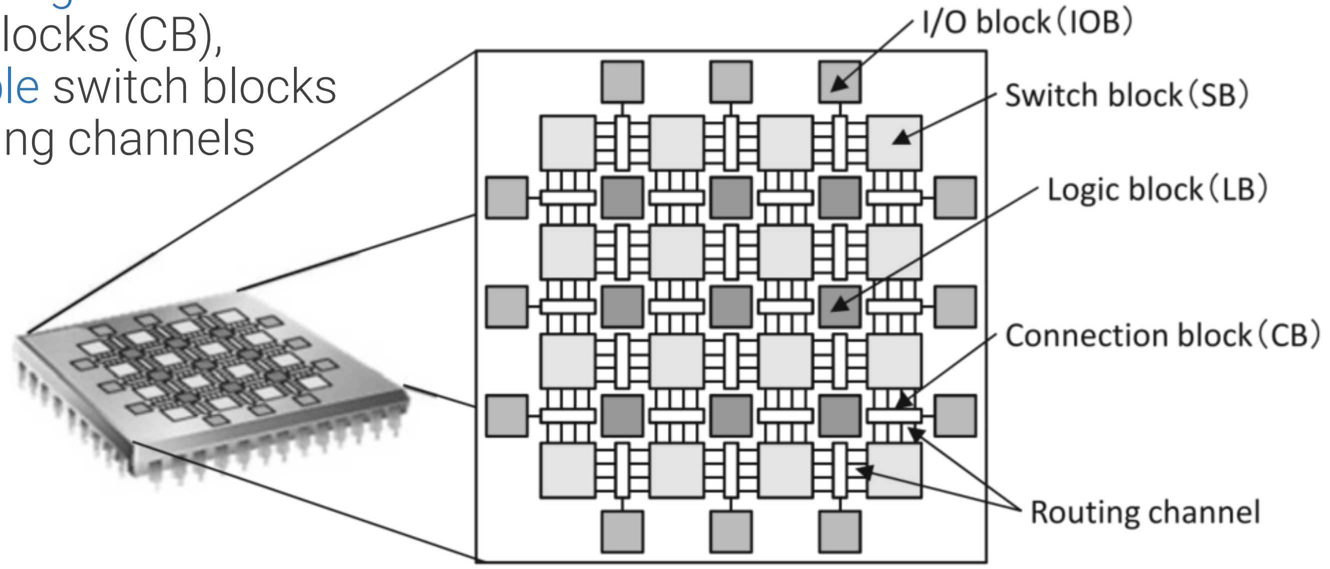
- A PLD which combines multiple logic blocks in a device for a high degree of "programming" (reconfiguring)
- Can be used to implement any digital circuit
- Has a gate-array-like structure
- Mass-produced with a blank (initial) state



M. Iida (✉)
Kumamoto University, Kumamoto, Japan
e-mail: iida@cs.kumamoto-u.ac.jp

Field Programmable Gate Arrays (FPGAs)

- Logic Block (LB) or Logic Element (LE)
  - Programmable logic through lookup tables (LUTs)
- Input/Output Block (IOB)
  - Programmable block that connects I/O pins and internal wiring elements
- Wiring Element
  - Consists of programmable connection blocks (CB), programmable switch blocks (SB), and wiring channels

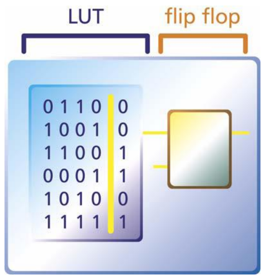


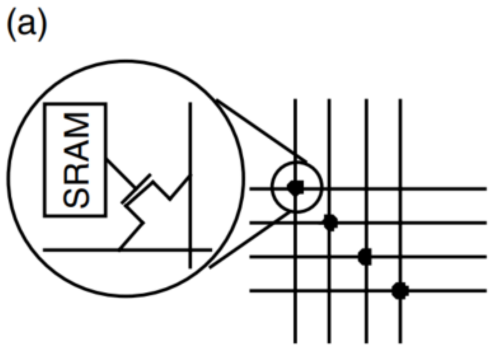
M. Iida (✉)
Kumamoto University, Kumamoto, Japan
e-mail: iida@cs.kumamoto-u.ac.jp

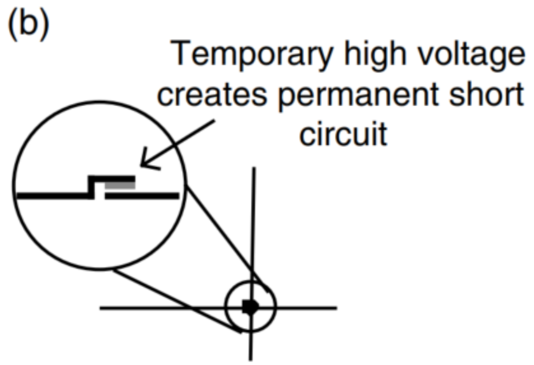
© Springer Nature Singapore Pte Ltd. 2018
H. Amano (ed.), *Principles and Structures of FPGAs*,
https://doi.org/10.1007/978-981-13-0824-6_2

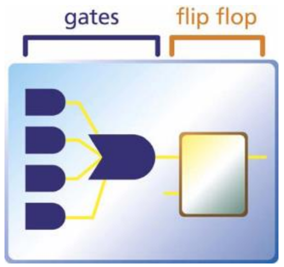
FPGA Programmability

- SRAM-based Programmable
  - SRAM cells control FPGA connections *(a)*
  - Allows fast in-circuit reconfiguration
  - Can be programmed an unlimited number of times
  - Used by Xilinx™, Altera™ *(now Intel®)* FPGAs
- Anti-fuse Technology
  - Anti-fuse remains in a high-impedance state until it is programmed into a low-impedance or “fused” state *(b)*
  - Can only be programmed once on *one-time programmable (OTP)* devices
  - Used by Actel™, Quicklogic™



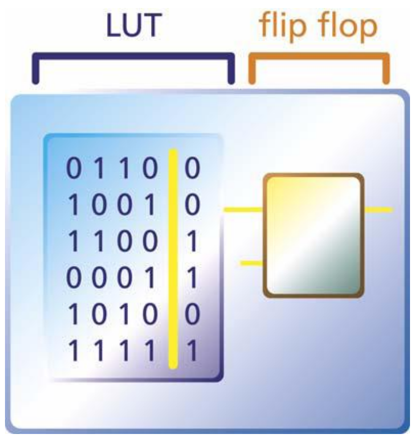






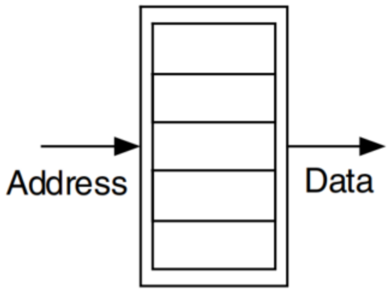
Synthesis of Arithmetic Circuits: FPGA, ASIC, and Embedded Systems
By Jean-Pierre Deschamps, Géry J. A. Bioul, and Gustavo D. Sutter
Copyright © 2006 John Wiley & Sons, Inc.

Lookup Tables (LUTs)

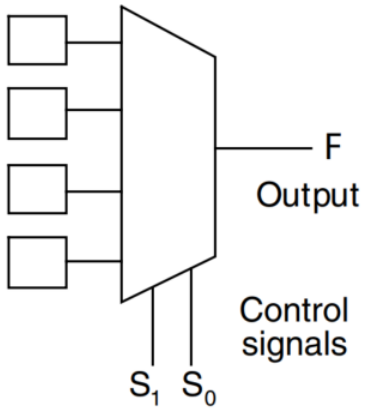


- LUTs are used to implement logic functions
  - Logic blocks implemented as memory or mux with memory:
    - Memory: A $2^n \times 1$ ROM can implement any *n*-bit function *(a)*
    - Mux & memory: Mux control inputs are the LUT inputs *(b)*
    - Memory contents example for different logic functions *(c)*

(a)



(b)



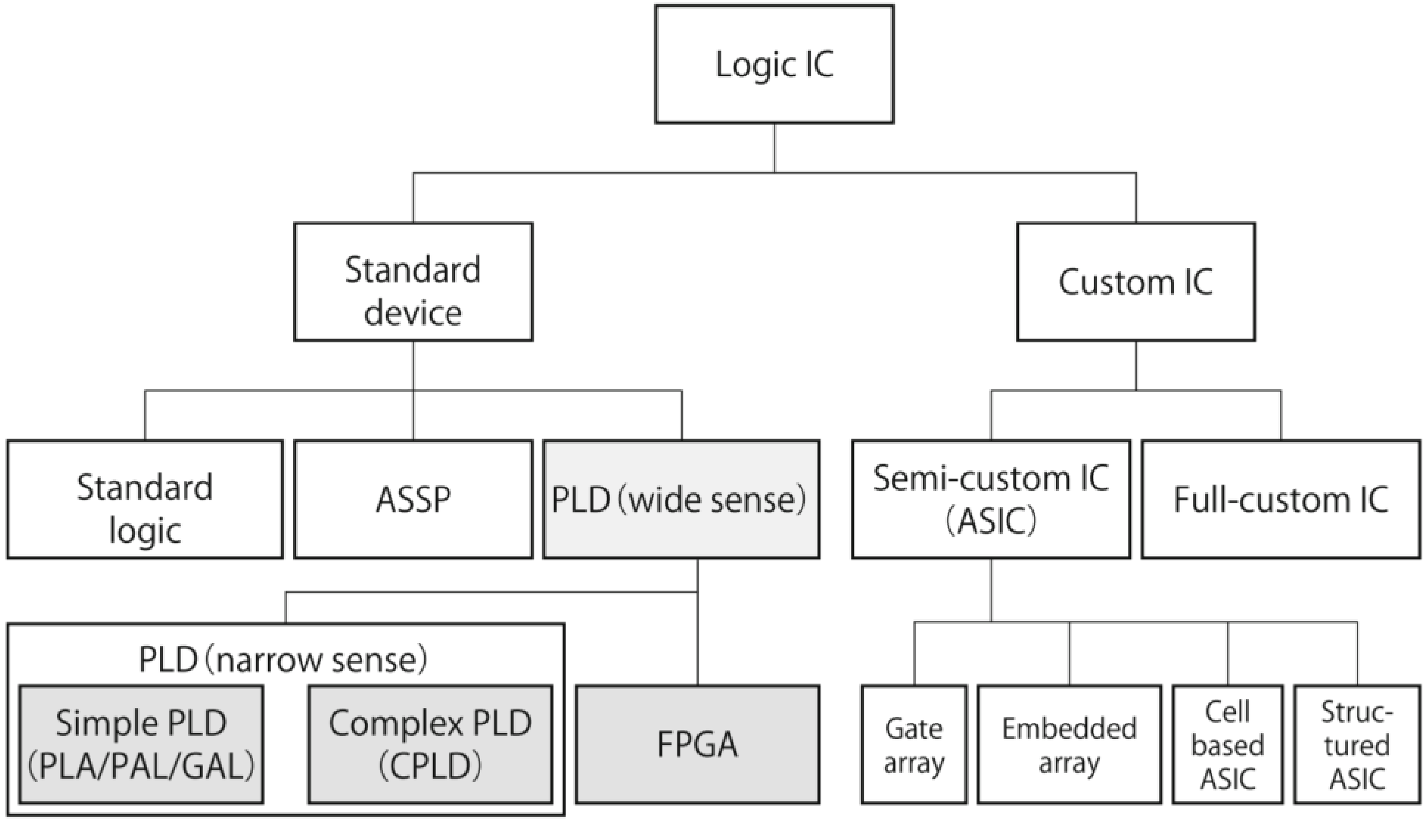
(c)

INPUTS	AND	OR	XOR	NAND
0 0	0	0	0	1
0 1	0	1	1	1
1 0	0	1	1	1
1 1	1	1	0	0

. . .

Synthesis of Arithmetic Circuits: FPGA, ASIC, and Embedded Systems
By Jean-Pierre Deschamps, Géry J. A. Bioul, and Gustavo D. Sutter

Position of FPGAs in Logic Devices

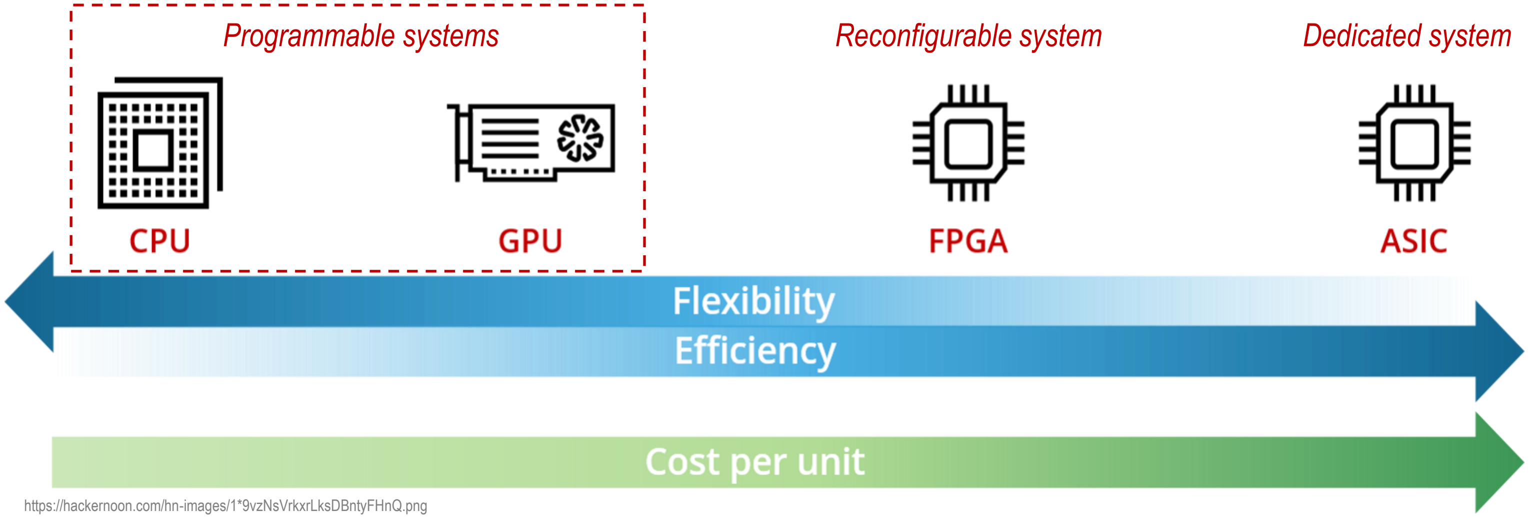


T. Sueyoshi (✉)
Kumamoto University, Kumamoto, Japan
e-mail: sueyoshi@cs.kumamoto-u.ac.jp

© Springer Nature Singapore Pte Ltd. 2018
H. Amano (ed.), *Principles and Structures of FPGAs*,
https://doi.org/10.1007/978-981-13-0824-6_1

CPU, GPU, FPGA, and ASICs

Tradeoffs



https://hackernoon.com/hn-images/1*9vzNsVrkxrLksDBntyFHnQ.png

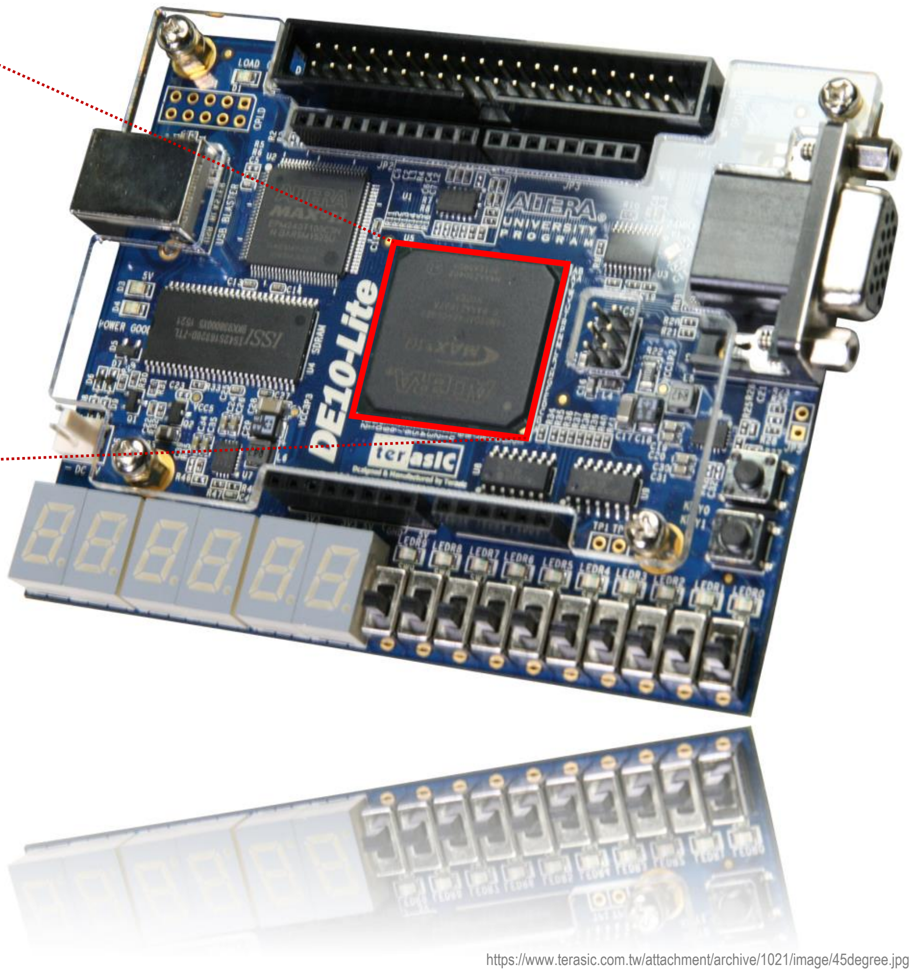
	CPU	GPU	FPGA	ASIC
Compute Adaptability (to a variety of situations)	High	Medium	Low	None
Compute power	Medium	High	High	Medium
Latency	Medium	High	Low	Ultra low
Throughput	Low	High	High	High
Parallelism	Low	High	High	High
Power efficiency	Medium	Low	Medium	High

https://hackernoon.com/hn-images/1*gnkYBIN1qVBdo3CmlMo7BA.png

Intel® MAX 10 FPGA

- 50K programmable logic elements
- 1,638 Kbit M9K Memory
- 144 18x18 Multiplier
- 4 Phase-locked loops (PLLs)
- Integrated dual ADCs

- On-Board USB Blaster (Normal type B USB connector)
- 64MB SDRAM, x16 bits data bus
- 10 LEDs
- 10 Slide Switches
- 2 Push Buttons
- Six 7-Segments Display
- 5V DC input

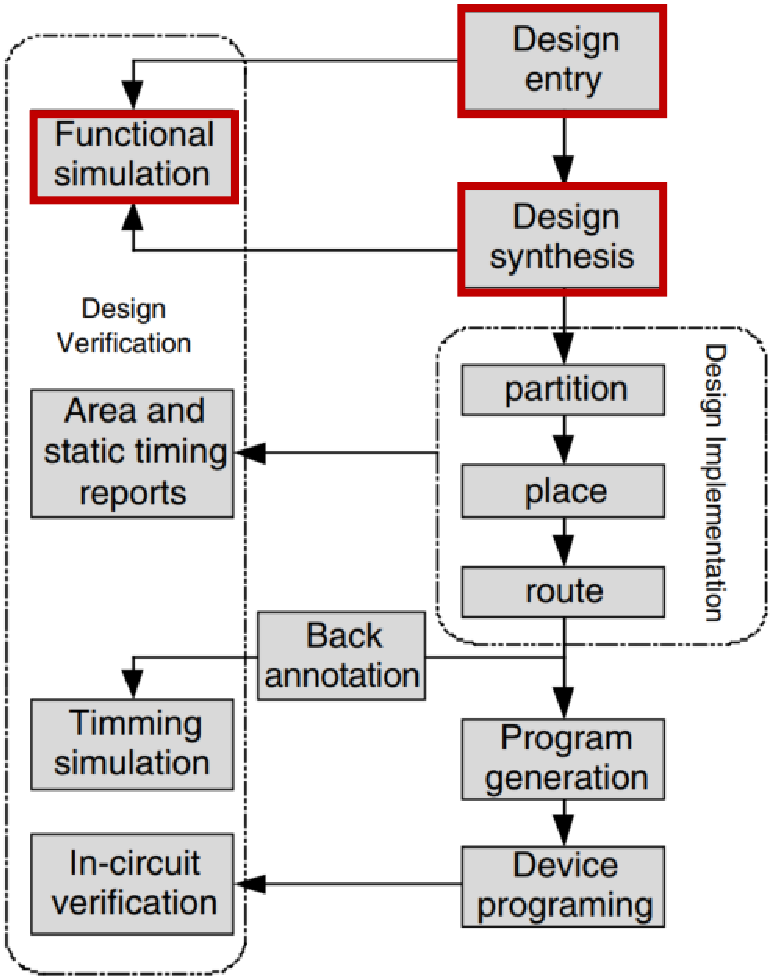


https://www.terasic.com.tw/attachment/archive/1021/image/45degree.jpg

Terasic DE10-Lite FPGA Board

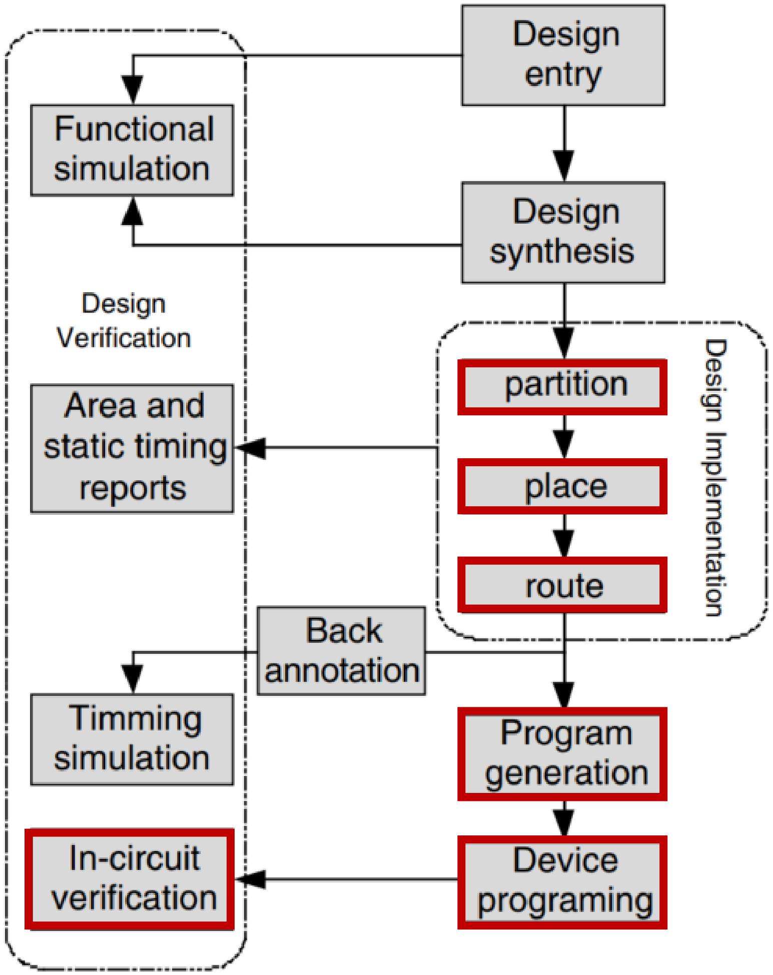
Design Flow in Modern Digital System Design

- FPGA design flow is similar with semicustom ASIC design's
  - Use of CAD-based design tools
- **Design Entry:** creates design files using hardware description language (HDL)
- **Design *(Logic)* Synthesis:** entered design is synthesized into a circuit that consists of logic elements in FPGA chip *(derives a list of logic elements and their interconnections or netlist)*
- **Functional *(Logic)* Simulation:** displays behavior of designed system
  - Uses signal waveforms
  - Detects functional errors
  - Does not check timing and delays



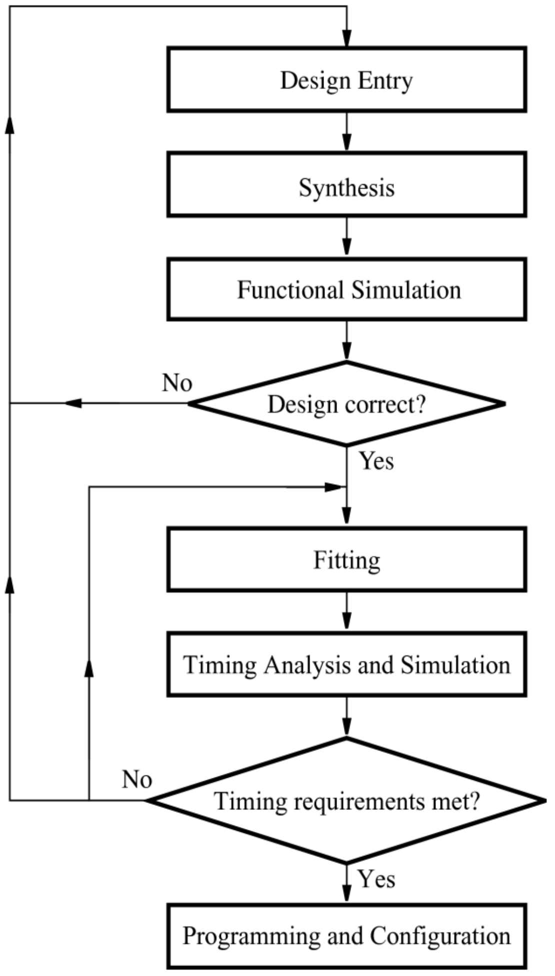
Design Flow in Modern Digital System Design

- Partition *(Mapping):* assigns LEs to physical elements for implementing the logic function in the target FPGA
- Place: maps logic into specific locations
- Route: connects the mapped logic
- Program Generation: generates a bit-stream file to program the device
- Device Programming: downloads the bit-stream file to the FPGA for reconfiguration
- In-circuit Verification: checks the actual behavior of the implemented system on the target FPGA



Simplified Design Flow (Intel)

- Typical FPGA CAD flow:
- Design Entry: creates design files using HDL
- Synthesis: entered design is synthesized into a circuit that consists of logic elements in FPGA chip
- Functional Simulation: displays behavior of designed system
- Fitting: combined partition, place, and route processes
- Timing Analysis and Simulation: analysis and simulation of propagation delays in fitted circuit
- Programming and Configuration: implementation of designed circuit in physical FPGA device



© Intel® Corporation – FPGA University Program, March 2019

Hardware Description Languages (HDLs)

- A computer-based language that *describes the hardware of digital systems* in a textual form
  - Resembles an ordinary computer programming language
  - Specifically oriented to describe hardware structures and behavior of digital systems

- A documentation language used to represent digital systems in a form read by both humans and computers
- Suitable as an exchange language between designers
- Used in several steps in the design flow of digital systems: *design entry, design synthesis, functional simulation,* and *timing/fault simulations.*

Hardware Description Languages (HDLs)

- 2 standard HDLs supported by IEEE in the public domain:

- VHDL: *VHSIC HDL (Very high-speed integrated circuit)*
  - Department of Defense-mandated language
  - A rich, strongly typed language *(with more data types, conversion)*
  - Deterministic and more verbose than Verilog *(self-documenting)*
  - It often catches errors missed by Verilog

- Verilog
  - Easier language than VHDL to describe, learn, and use
  - Weakly typed and more concise with efficient notation
  - Deterministic and all data types are predefined with bit-level representation

Introduction to Verilog HDL

- Began as a proprietary HDL of Cadence Design Systems
- Was transferred to Open Verilog International (OVI), a consortium of companies and universities
- Initially approved as a standard HDL in 1995 (Verilog-1995)
- Revised version was approved in 2001 (Verilog-2001)
- Syntax is somewhat like that of the C language
- Semantics (i.e., “meaning”) is based on *concurrent hardware operation* and is totally different from sequential execution of C

Verilog HDL: Focus in CPE 3101L

- Introduce a design methodology based on the concept of CAD modeling of digital systems using a standard, typical HDL: Verilog
- Emphasis will be on the modeling *(design entry)*, synthesis *(design synthesis)*, and verification *(functional simulation)*
- Focus on Verilog synthesis constructs for hardware design
  - Instead of writing quick/short codes, *we focus on style and constructs that are CLEAR and SYNTHESIZABLE* to accurately describe hardware
- Introduce Verilog simulation constructs for functional simulations

Introduction to Intel® Quartus® Prime

- Perform Laboratory Exercise #1

End of Unit 1