EE2000 Logic Circuit Design

Chapter 6 – Programmable logic devices

Outline

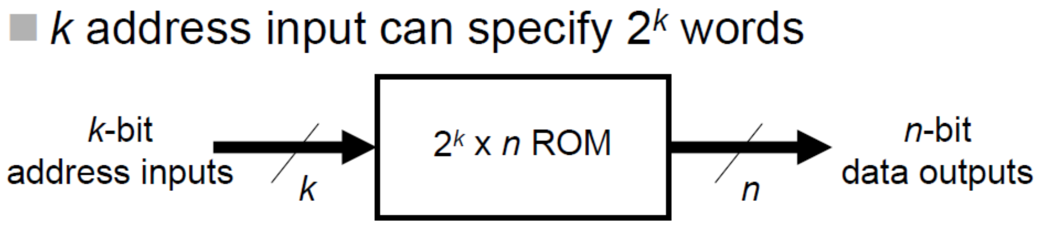
- 6.1 Programmable logic devices
- 6.2 ROM and PROM
- 6.3 Fuse-Programmable Arrays
- 6.4 Programming Logic Array (PLA)
- 6.5 Programmable Array Logic (PAL)
- 6.6 Field Programmable Gate Array Logic (FPGA)

6.1 Programmable logic devices

- Implementation technologies introduced so far are not programmable
  - Fixed integrated circuits
- Programmable Logic Device (PLD)
  - Integrated circuit with internal logic gates connected together
  - These internal connections can be reconfigured (erased / programmed) to form a specific logic circuit
- Three types of basic PLDs
  - ROM, PLA, PAL

6.2.1 Read-Only Memory (ROM)

- An essential memory device in which **permanent** binary information is stored (even when the power is turned off, the information is still there).
- *k*-bit address inputs and *n*-bit data outputs
  Inputs: address for the memory
  Outputs: the word (*n* data bits) stored in ROM selected by the *k*-bit input address
- *k* address input can specify 2^k words



- Features:
  - advantage of simple and less connections
  - disadvantage of doubling the size with an additional input
  - cheap for mass production but expensive for development

Other ROMs

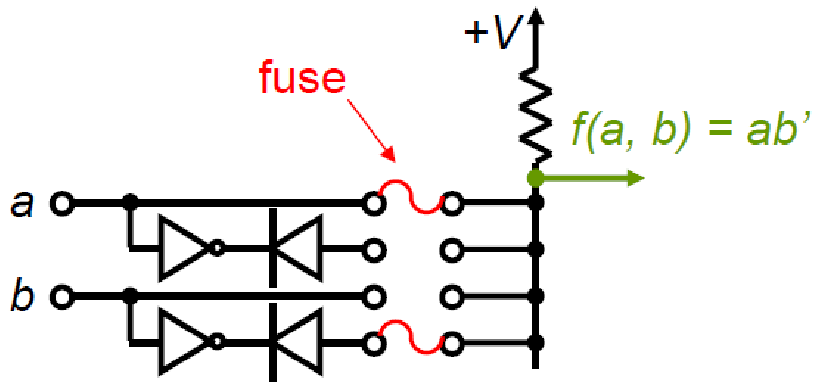
- ROM usually works as a lookup table
- No **data inputs** as ROM does not have a **write** operation (only read operation)
- ROMs that can be written / rewritten
  PROM
  EPROM (Erasable PROM)
  EEPROM (Electrically Erasable PROM)
  FEPROM (Field Erasable PROM)
  FLASH Memory

6.2.2 PROM

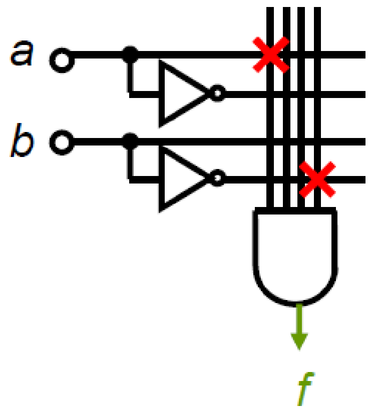
- Programmable Read-Only Memory
- Perform same functions as ROM
- Oldest programmable logic device
- Fuse or anti-fuse programmed
- One-time writing process has been deferred to the end-user
- Advantage: easy programming and inexpensive
- Disadvantage: non-reprogrammable

6.3 Fuse-Programmable Arrays

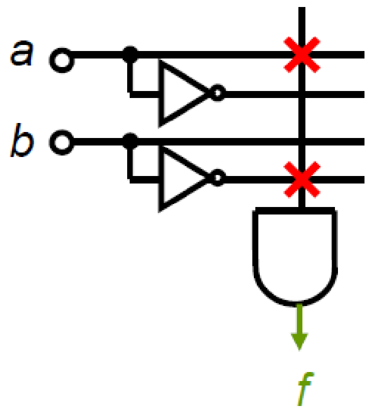
- Programmable AND Arrays



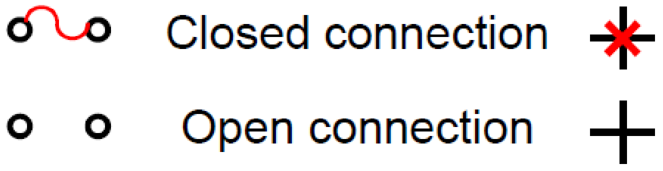
Fused-programmable AND arrays



Conventional symbol

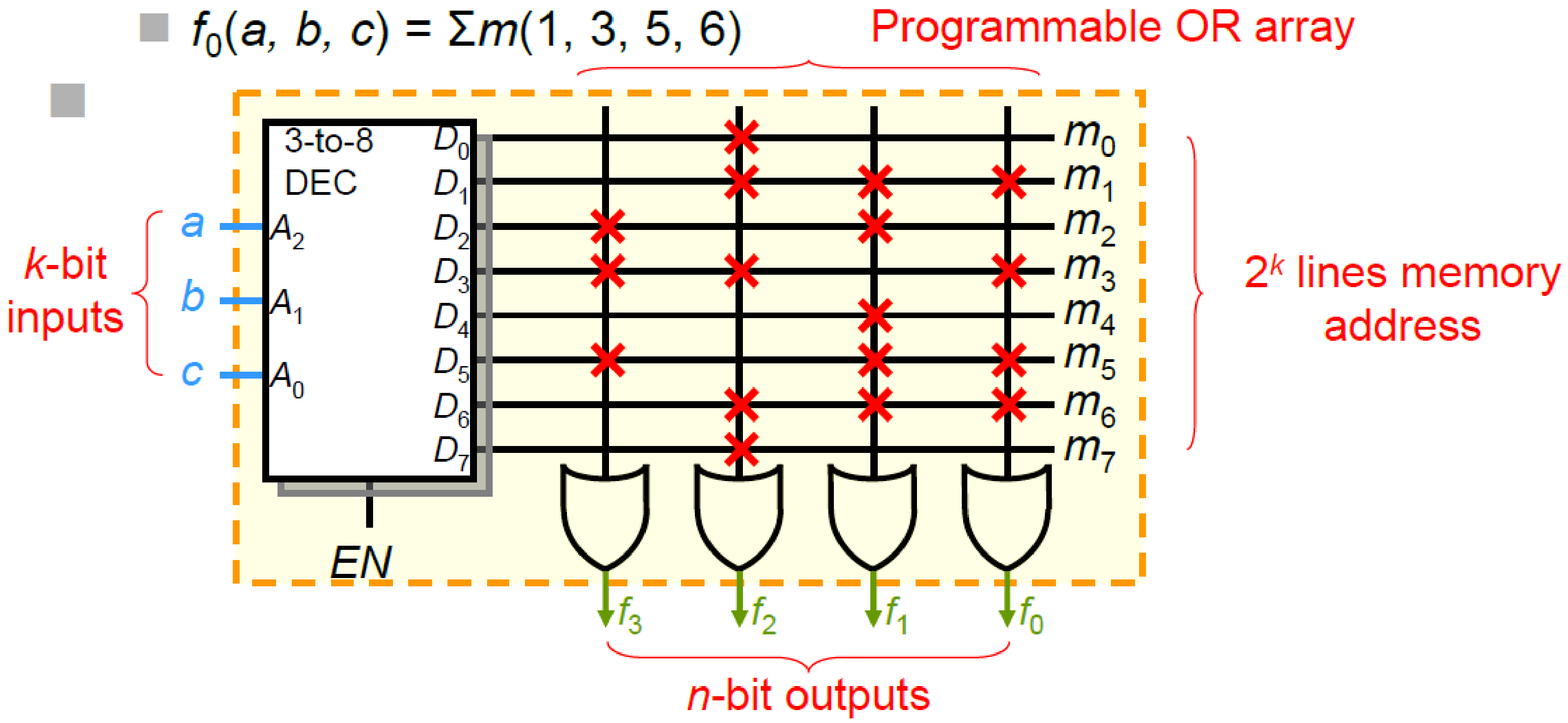


Array logic symbol



Example: 8 x 4 PROM

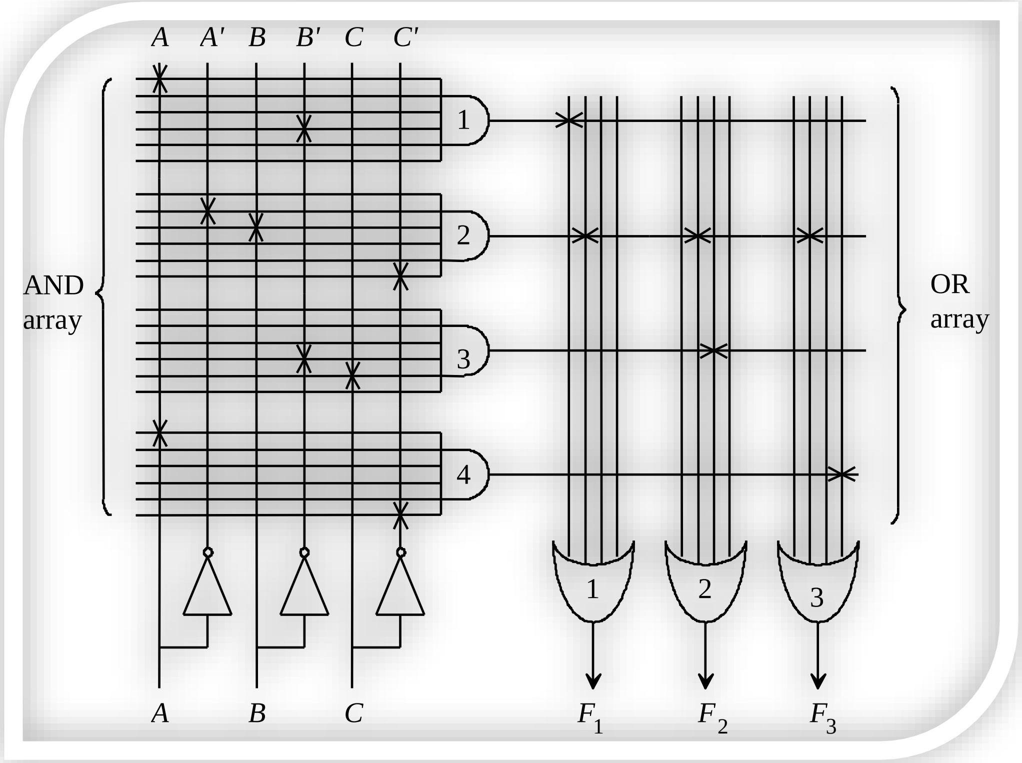
- Realize the following functions with a 8 x 4 PROM
  - $f_3(a, b, c) = \Sigma m(2, 3, 5)$
  - $f_2(a, b, c) = \Sigma m(0, 1, 3, 6, 7)$
  - $f_1(a, b, c) = \Sigma m(1, 2, 4, 5, 6)$
  - $f_0(a, b, c) = \Sigma m(1, 3, 5, 6)$
-



Programmable OR array
3-to-8 DEC
D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7
A_2 A_1 A_0
a b c
k-bit inputs
EN
m_0 m_1 m_2 m_3 m_4 m_5 m_6 m_7
2^k lines memory address
f_3 f_2 f_1 f_0
n-bit outputs

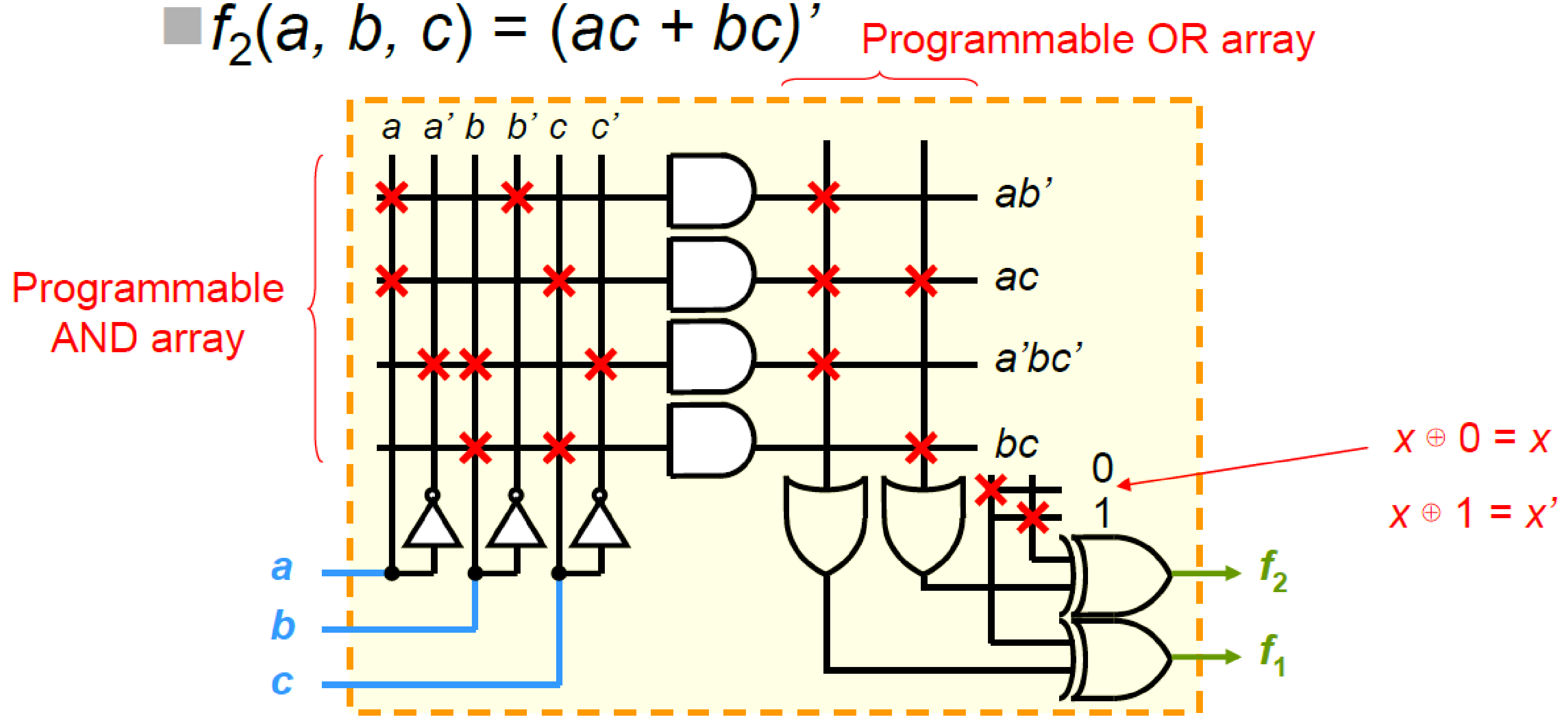
6.4 Programming Logic Array (PLA)

- Programmable device similar to PROM but with different internal structure
- The device is used to implement logic function in the form of sum-of- product terms



Example: PLA

- Implement these 2 functions with a PLA
  - $f_1(a, b, c) = ab' + ac + a'bc'$
  - $f_2(a, b, c) = (ac + bc)'$



- How many product terms for f_1 and f_2?

Minimization in PLA

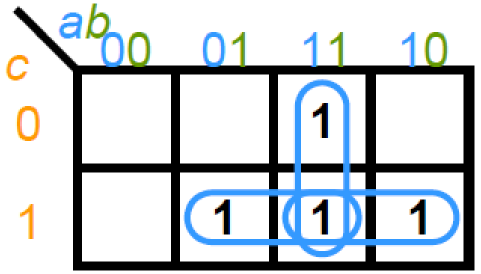
- Reduce the number of distinct product terms, so as to
  Minimize the number of AND gates
  Minimize the size of PLA

- Step 1) Simplify the Boolean functions, and also their **complementary functions** to minimum number of terms

- Step 2) Find the combination to produce the minimum number of distinct product terms (or maximum common terms)

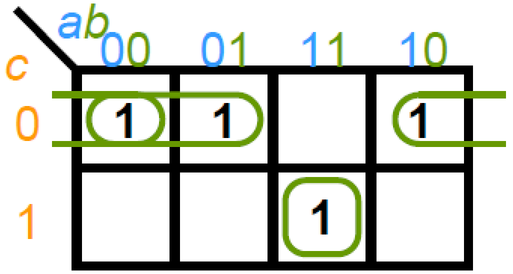
Minimization in PLA: Example

- Implement these 2 functions with a PLA
  - $f_1(a, b, c) = \Sigma m(3, 5, 6, 7)$
  - $f_2(a, b, c) = \Sigma m(0, 2, 4, 7)$
- Step 1) Simplify by K-map first:



c \ ab	00	01	11	10
0			1	
1		1	1	1

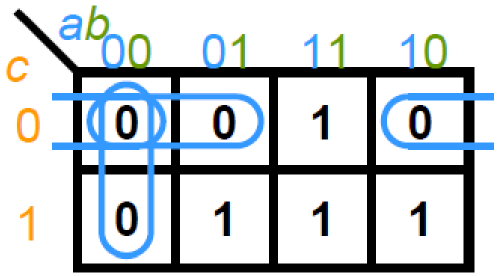
$f_1 = ac + ab + bc$



c \ ab	00	01	11	10
0	1	1		1
1			1	

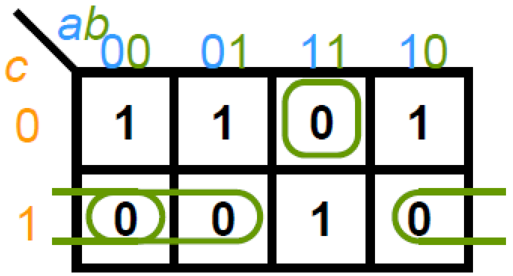
$f_2 = \underline{a'c'} + \underline{b'c'} + abc$

- Now find the simplified complementary functions



c \ ab	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$f_1' = \underline{a'c'} + a'b' + \underline{b'c'}$



c \ ab	00	01	11	10
0	1	1	0	1
1	0	0	1	0

$f_2' = a'c + b'c + abc'$

Minimization in PLA: Example

- Since there are two functions f_1 and f_2
- There are 4 complemented and uncomplemented combinations (why?)
  - (i) f_1 and f_2
  - (ii) f_1 and f_2'
  - (iii) f_1' and f_2
  - (iv) f_1' and f_2'
- We may implement design (i), (ii), (iii), or (iv)
  - How to choose?
  - The design (iii) give the min. no. of distinct product terms (shared 2 common terms)

Minimization in PLA: Example

- Tabular the result of last step in the PLA programming table
  - Design (iii): f_1' and f_2

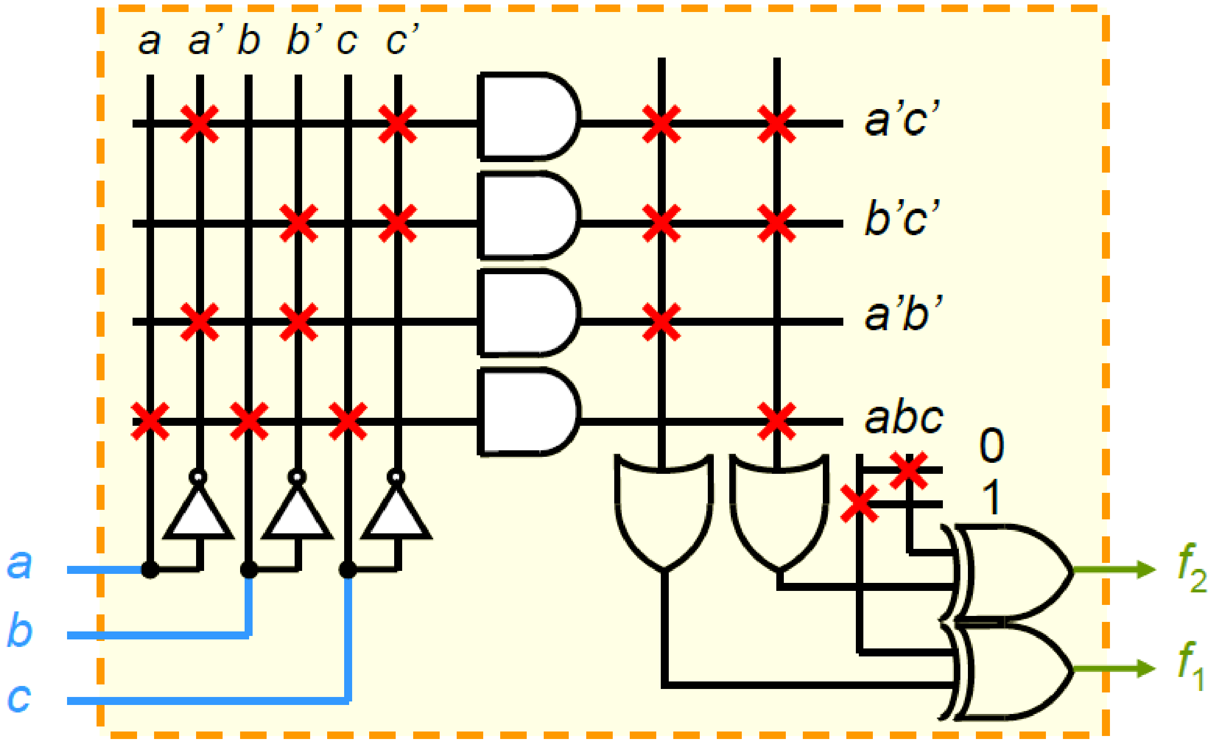
	Product Term	Inputs $a\ b\ c$	Outputs $f_1\ f_2$
$a'c'$	1	0 – 0	1 1
$b'c'$	2	– 0 0	1 1
$a'b'$	3	0 0 –	1 –
abc	4	1 1 1	– 1
			C T

T: Uncomplemented output

C: Complemented output

Minimization in PLA: Example

- The final resulting PLA fuse map



6.5 Programmable Array Logic (PAL)

- PLA: both the AND and the OR arrays are programmable
- PAL: the AND array is programmable and the OR array is fixed
- PLA is a general design for implementing sum-of-product terms, whereas the PAL has fixed sum-of-product terms
- Design for PAL device is easier, but not as flexible as that for PLA

PAL Example

- Implement these functions with a PAL
  - $w(a, b, c, d) = \Sigma m(2, 12, 13)$
  - $x(a, b, c, d) = \Sigma m(7, 8, 9, 10, 11, 12, 13, 14, 15)$
  - $y(a, b, c, d) = \Sigma m(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)$
  - $z(a, b, c, d) = \Sigma m(1, 2, 8, 12, 13)$
- Analysis
  - 4 inputs (*a*, *b*, *c*, *d*), 4 outputs (*w*, *x*, *y*, *z*)

PAL Example

- Simplify the functions to minimum number of terms (steps omitted here)
  - $w(a, b, c, d) = abc' + a'b'cd'$
  - $x(a, b, c, d) = a + bcd$
  - $y(a, b, c, d) = a'b + cd + b'd'$
  - $z(a, b, c, d) = \underline{abc' + a'b'cd'} + ac'd' + a'b'c'd$
  - Note: *z* is sum of 4 product terms

  - By considering output *w* as an input variable
    - $z(a, b, c, d, \underline{w}) = w + ac'd' + a'b'c'd$
    - Now *z* is SOP of 3 product terms!

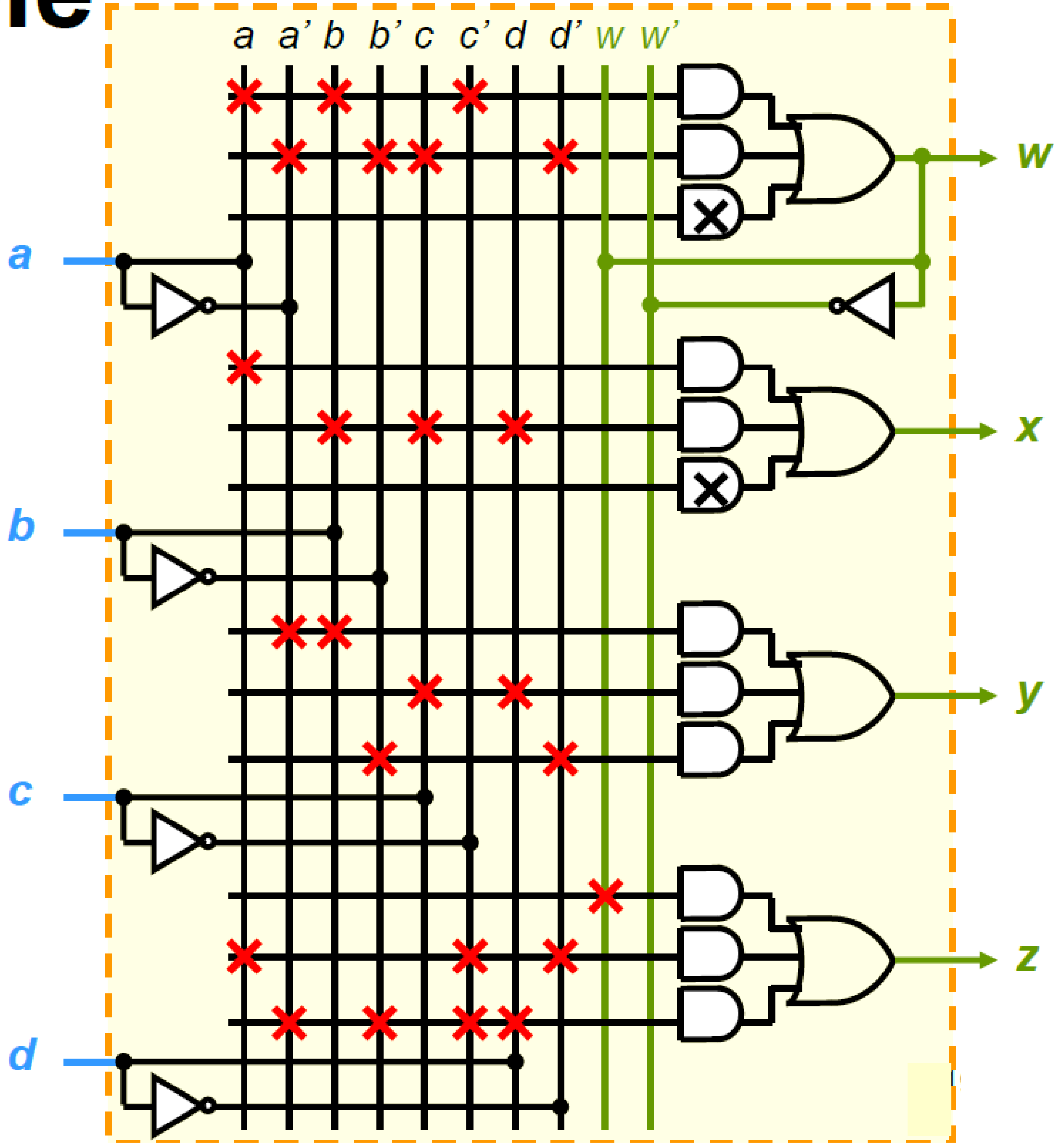
PAL Example

- Tabular the result of last step in the PAL programming table

Product Term		AND Inputs ***a b c d w***	Outputs
1	*abc'*	1 1 0 – –	*w = abc' + a'b'cd'*
2	*a'b'cd'*	0 0 1 0 –	
3		– – – – –	
4	*a*	1 – – – –	*x = a + bcd*
5	*bcd*	– 1 1 1 –	
6		– – – – –	
7	*a'b*	0 1 – – –	*y = a'b + cd + b'd'*
8	*cd*	– – 1 1 –	
9	*b'd'*	– 0 – 0 –	
10	*w*	– – – – 1	*z = w + ac'd' + a'b'c'd*
11	*ac'd'*	1 – 0 0 –	
12	*a'b'c'd*	0 0 0 1 –	

PAL Example

- The final result
- Reuse *w*

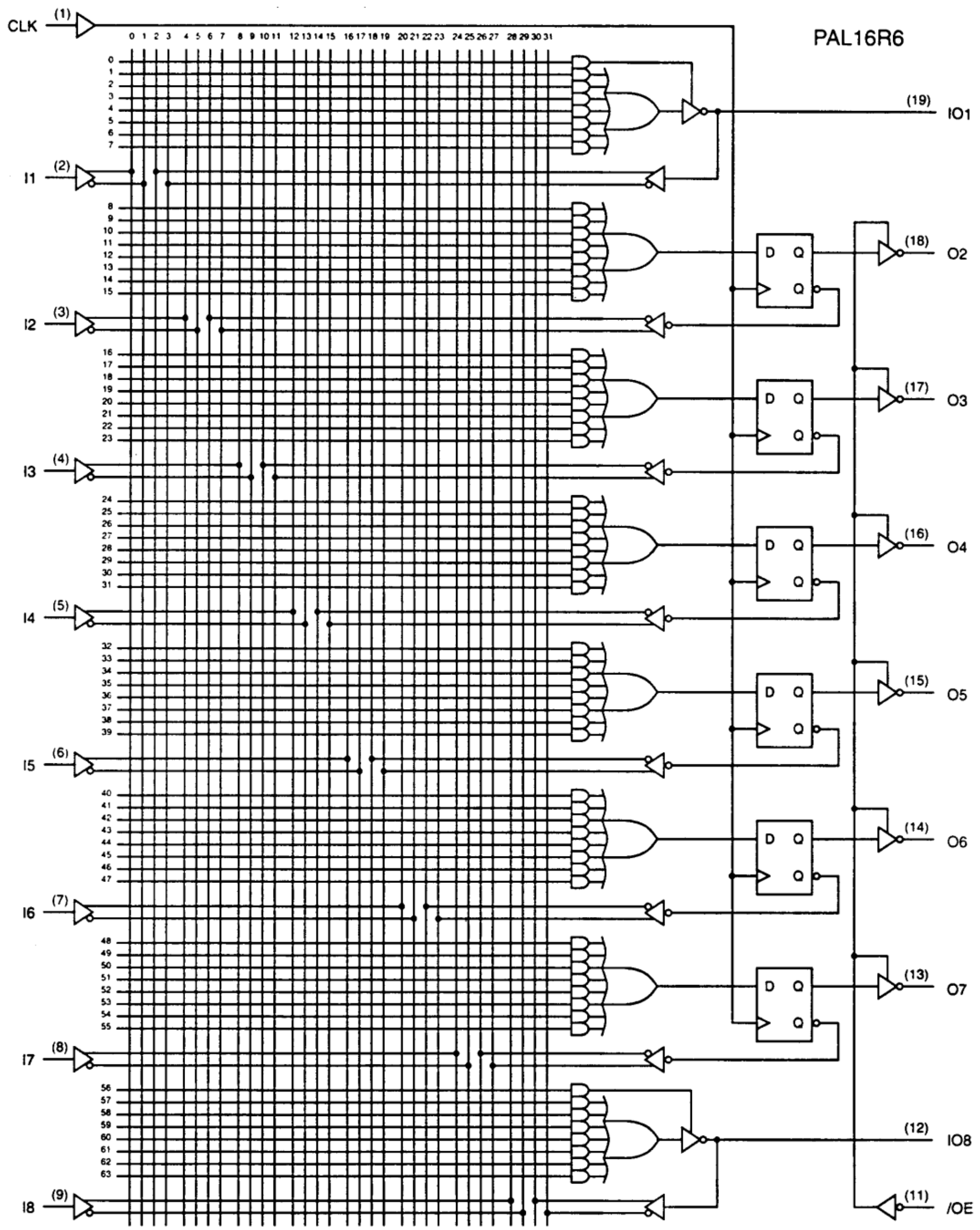


PAL 16R6

A PAL device with 16 inputs, 6 outputs with D-latch (1-bit memory device), and 2 un-buffered outputs

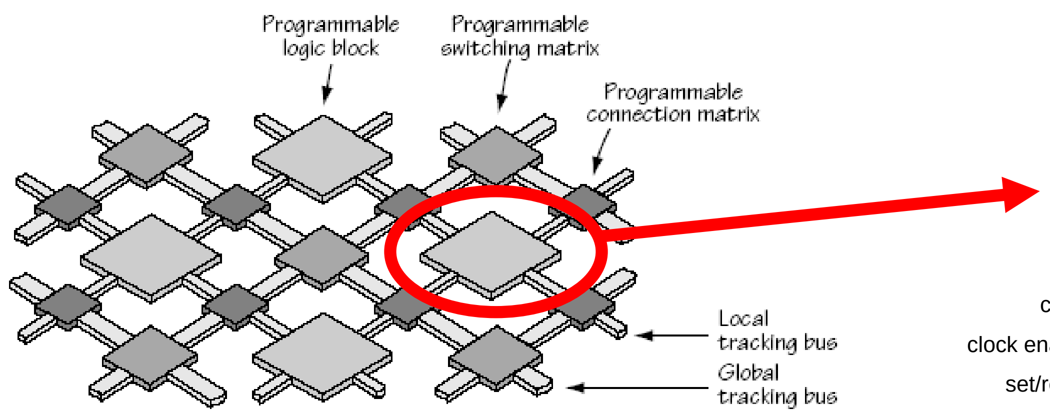
(*8 inputs are feedback from outputs*)

This device can be used for designing sequential circuit.

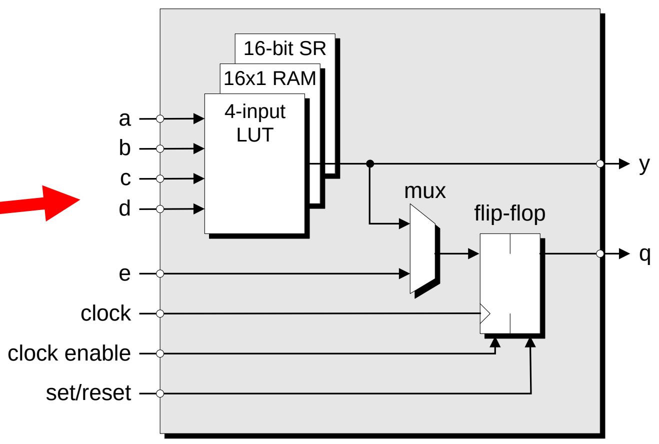


6.6 Field Programmable Gate Array Logic (FPGA)

- FPGAs consist of a large number of logic blocks with programmable interconnects in matrix form.
- Programmable:
  - ➢ The logic block can be "configured" to create the circuit.
  - ➢ The interconnects can also be "configured" to connect logic blocks together as well as to connect the FPGA pin through I/O blocks.



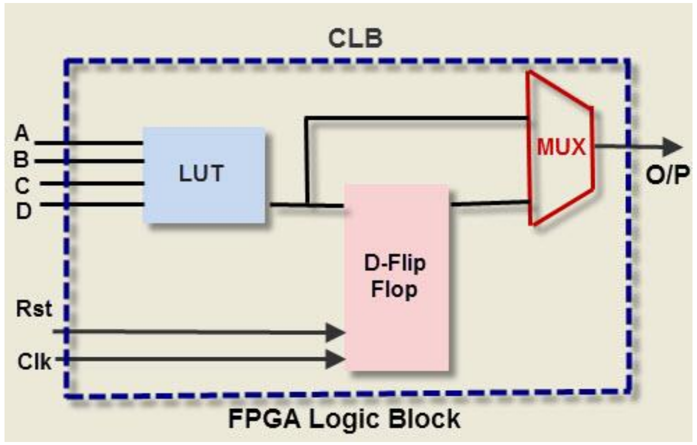
FPGA Fabric



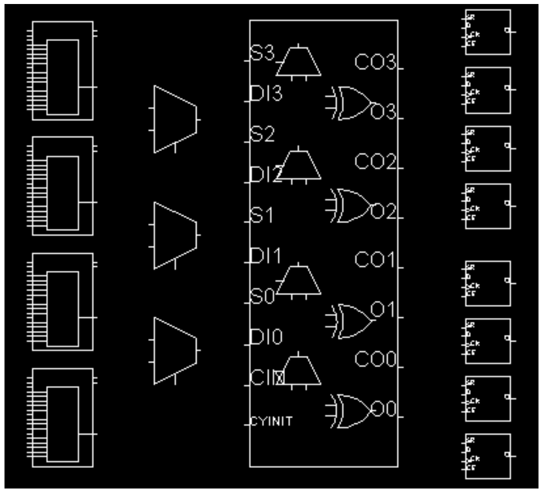
Logic Block

FPGA – Configurable Logic Block

- A configurable logic block (CLB) consists of a look-up-table (LUT)
- The LUT is for implementing the truth table of a logic function
- Additional elements are for control or other functions



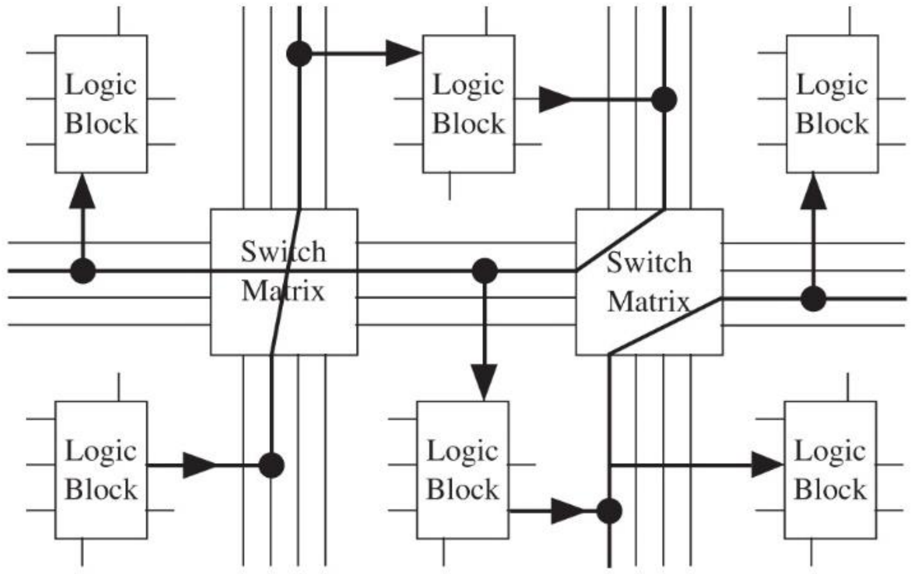
Simple basic logic block (in design level)



A typical logic block (in simulator)

FPGA – Programmable Interconnects

- FPGAs use ***switch matrices*** to provide interconnects for logic blocks
- The connections between logic blocks are achieved by programming the switch matrices to form routing paths



FPGA – I/O Standard

- The FPGA output may be connected to device with different electrical characteristics.
- The FPGA pin I/O standard must be specified.
- I/O Standards:
  - LVTTL: low-voltage transistor-transistor logic; 3.3-V standard that can tolerate 5-V signals.
  - PCI: peripheral component interconnect; has 5-V and 3.3-V versions.
  - LVCMOS: low-voltage complementary metal-oxide semiconductor; LVCMOS2, a 2.5-V standard that can tolerate 5-V signals.
  - LVPECL: low-voltage positive emitter-coupled logic
  - SSTL: stub-series terminated logic
  - AGP: advanced graphics port
  - CTT: center tap terminated
  - GTL: gunning transceiver logic
  - HSTL: high-speed transceiver logic

FPGA – Design Tools (Lab Session)

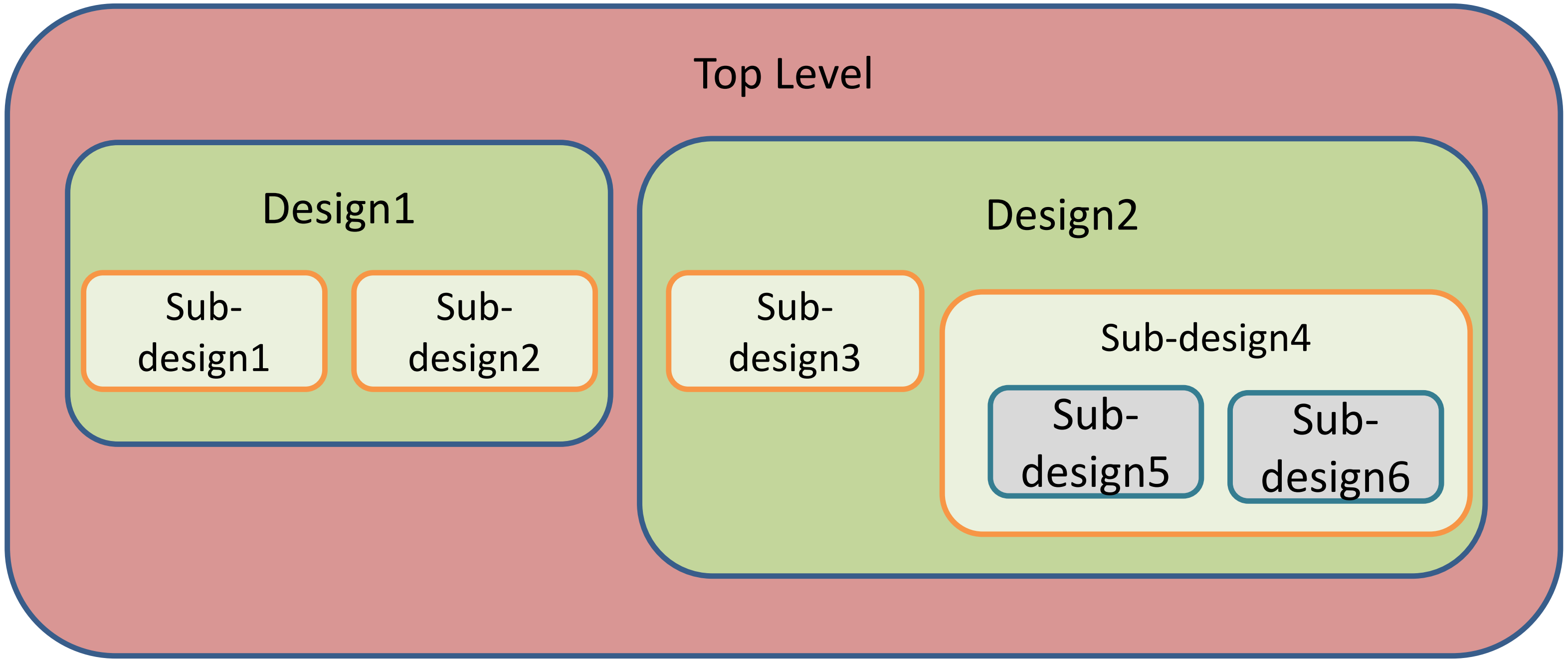
- FPGA is a programmable device
- Use HDL (hardware description language) to describe the circuitry
- Doesn’t behave like “normal” programming language ‘C/C++’
- Describe Logic as collection of Processes operating in Parallel
- Language Constructs for Synchronous Logic
- VHDL and Verilog are common HDLs
- VHDL and Xilinx Vivado are used in this course

FPGA – Design Flow

1. Create a behavioral, RTL, or structural model of the design in a hardware description language such as VHDL or Verilog.
2. Simulate and debug the design.
3. Synthesize the design targeting the desired device.
4. Run a mapping/partitioning program.
5. Run an automatic place and route program.
6. Run a program that will generate the bit pattern necessary to program the FPGA.
7. Download the bit pattern into the internal configuration cells in the FPGA, and test the operation of the FPGA.

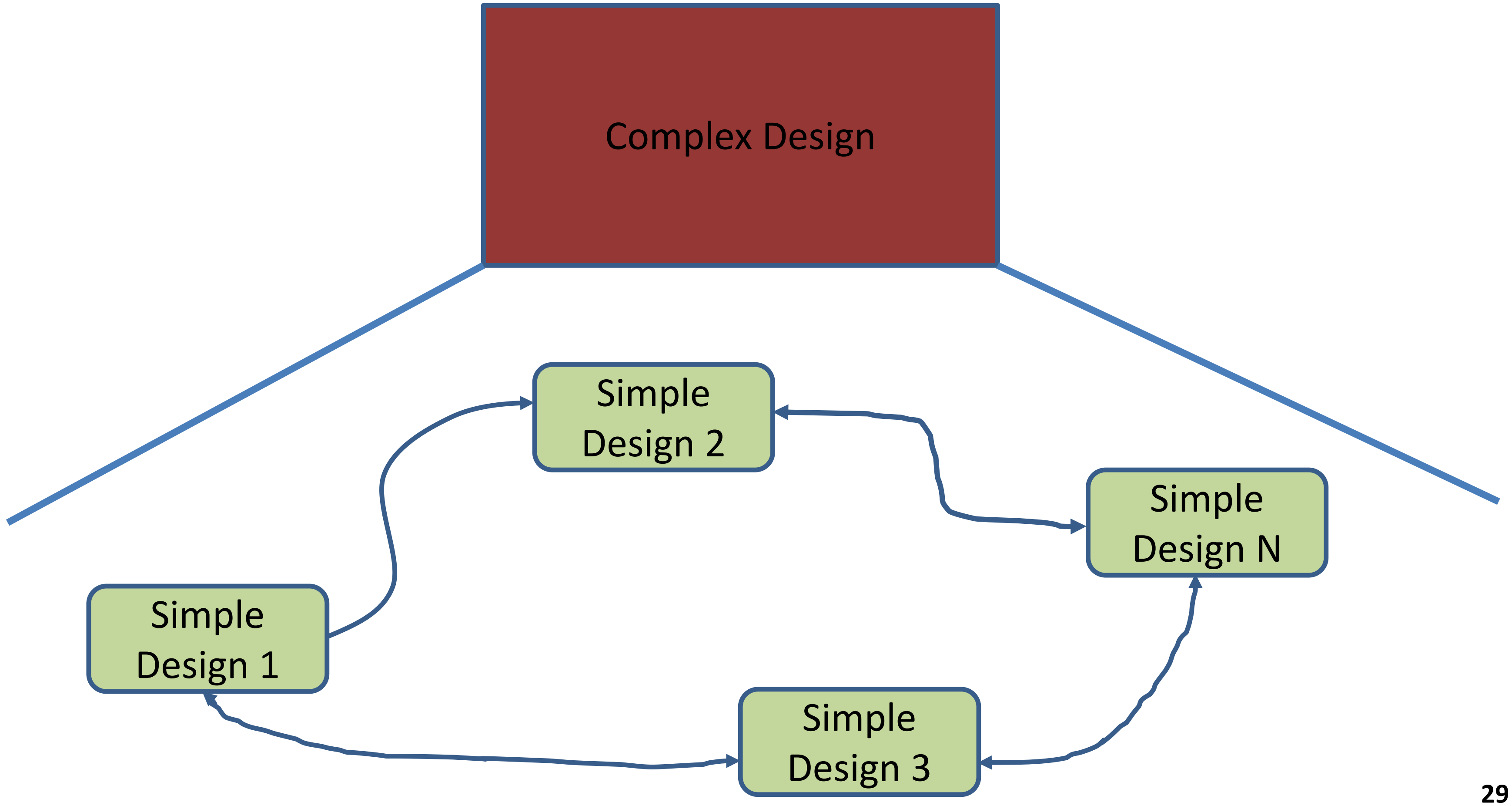
FPGA – Top-Down Design

- FPGA is particular suitable for digital system design.
- A complex system can be made simpler by using hierarchical approach
- Split the design in two or more simple design in order to easy handle the complexity



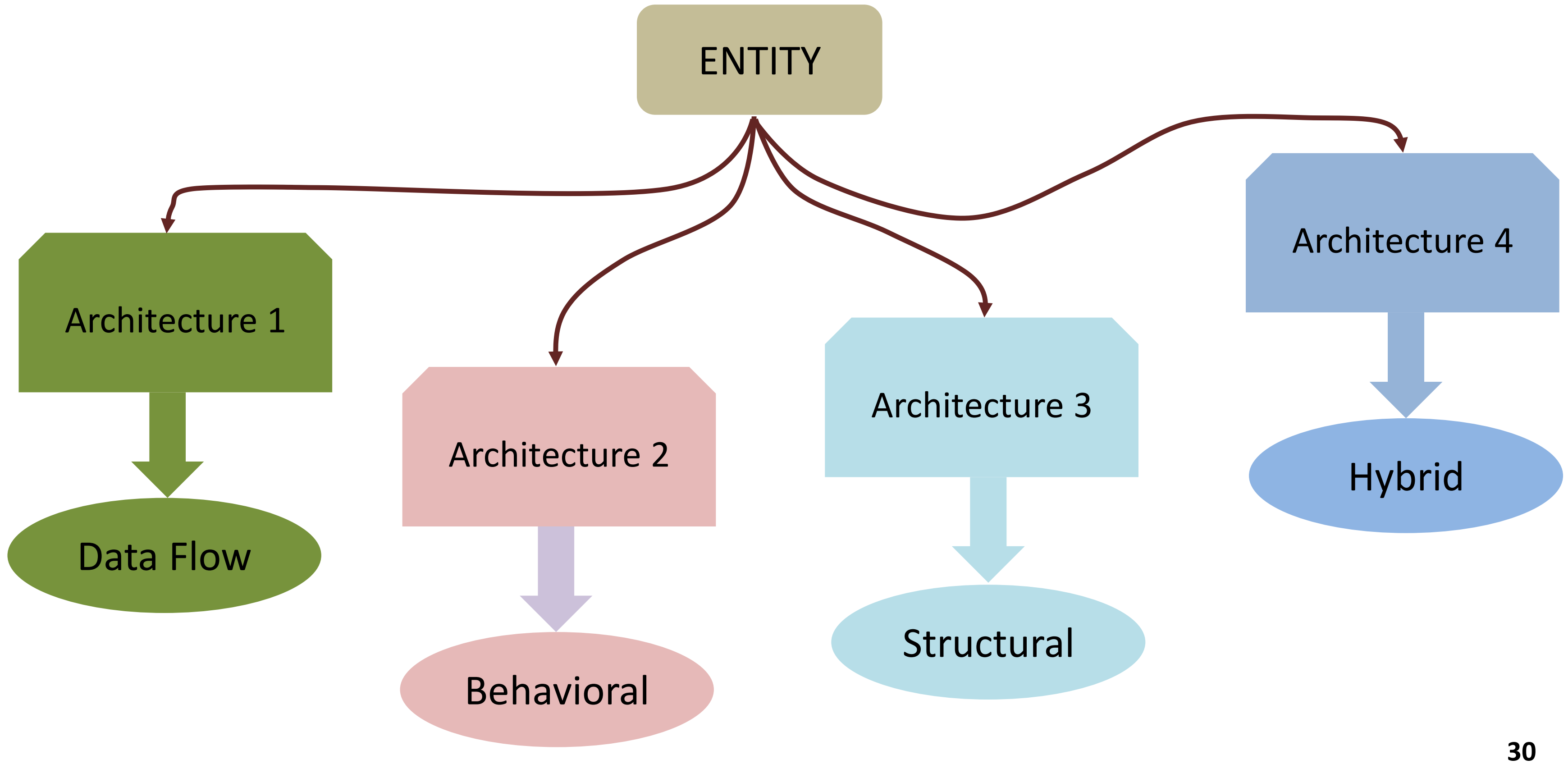
FPGA – VHDL Structural Modeling

- Use VHDL module to describe each simple design



FPGA – VHDL Module

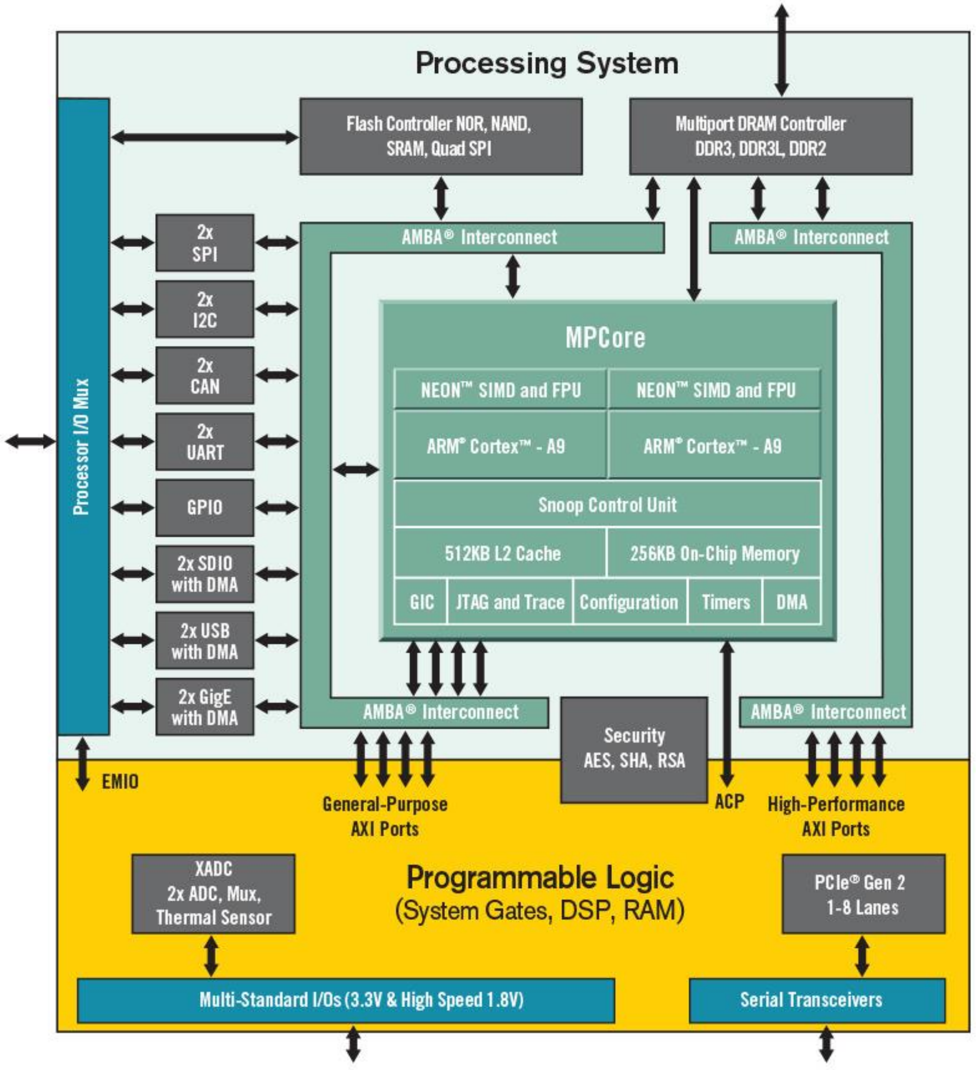
- Use Entity and Architecture to describe the each module (circuit)



FPGA – System on chip (SoC)

Advanced FPGA chip is integrated with CPU and peripheral controllers for advanced system design.

Programmable logic is provided for customer design.



Zynq-7000 (with two ARM Cortex-A9 processors)