EECE 2322
Fundamentals of Digital Logic Design and Computer Organization

Introduction to Programming FPGAs and the Lab (EECE 2323)

Dr. Thomas R. Consi

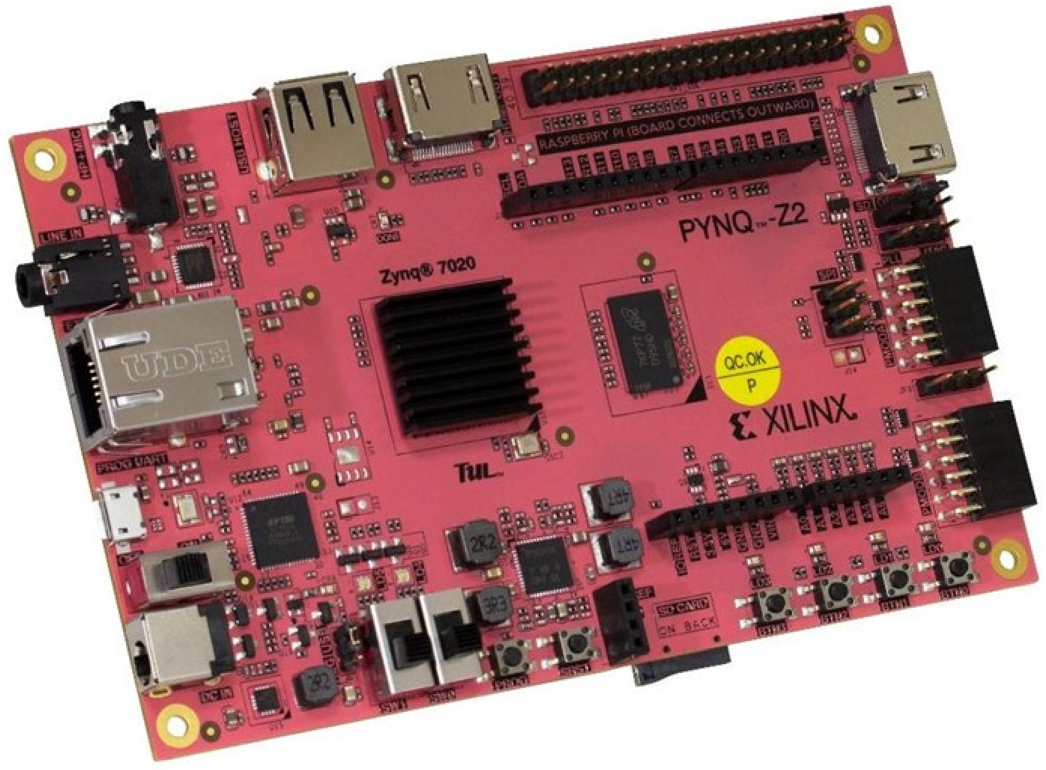
Dept. of Electrical and Computer Engineering

Northeastern University

EECE 2323 Digital Design Lab

Labs Start the Second Week of the Semester

- Companion lab course to EECE 2322
- EECE 2323 meets every week
  - Section 1: Tuesday, 08:00 AM to 10:00 AM
    - **First lab session on Jan. 16, 2024**
  - Section 2: Friday, 11:30 AM to 01:30 PM
    - **First lab Session on Jan. 19, 2024**
  - Section 3: Tuesday, 10:15 AM to 12:15 PM
    - **First lab session on Jan. 16, 2024**
  - Section 4: Friday, 09:15 AM to 11:15 AM
    - **First lab Session on Jan. 19, 2024**
  - Section 5: Friday, 01:45 PM to 03:45 PM
    - **First lab Session on Jan. 19, 2024**
  - Section 6: Tuesday, 12:30 PM to 02:30 PM
    - **First lab session on Jan. 16, 2024**
  - Section 7: Tuesday, 02:45 PM to 04:45 PM
    - **First lab session on Jan. 16, 2024**
  - Section 8: Thursday, 09:15 AM to 11:15 AM
    - **First lab Session on Jan. 18, 2024**
- Lab: Hayden 009D (Basement)



TUL PYNQ Board with Xilinx FPGA
Used in EECE 2323 Lab

https://www.notebookcheck.net/TUL-PYNQ-Z2-An-Arduino-and-Raspberry-Pi-compatible-Xilinx-Zynq-C7Z020-based-PYNQ-development-board.426529.0.html

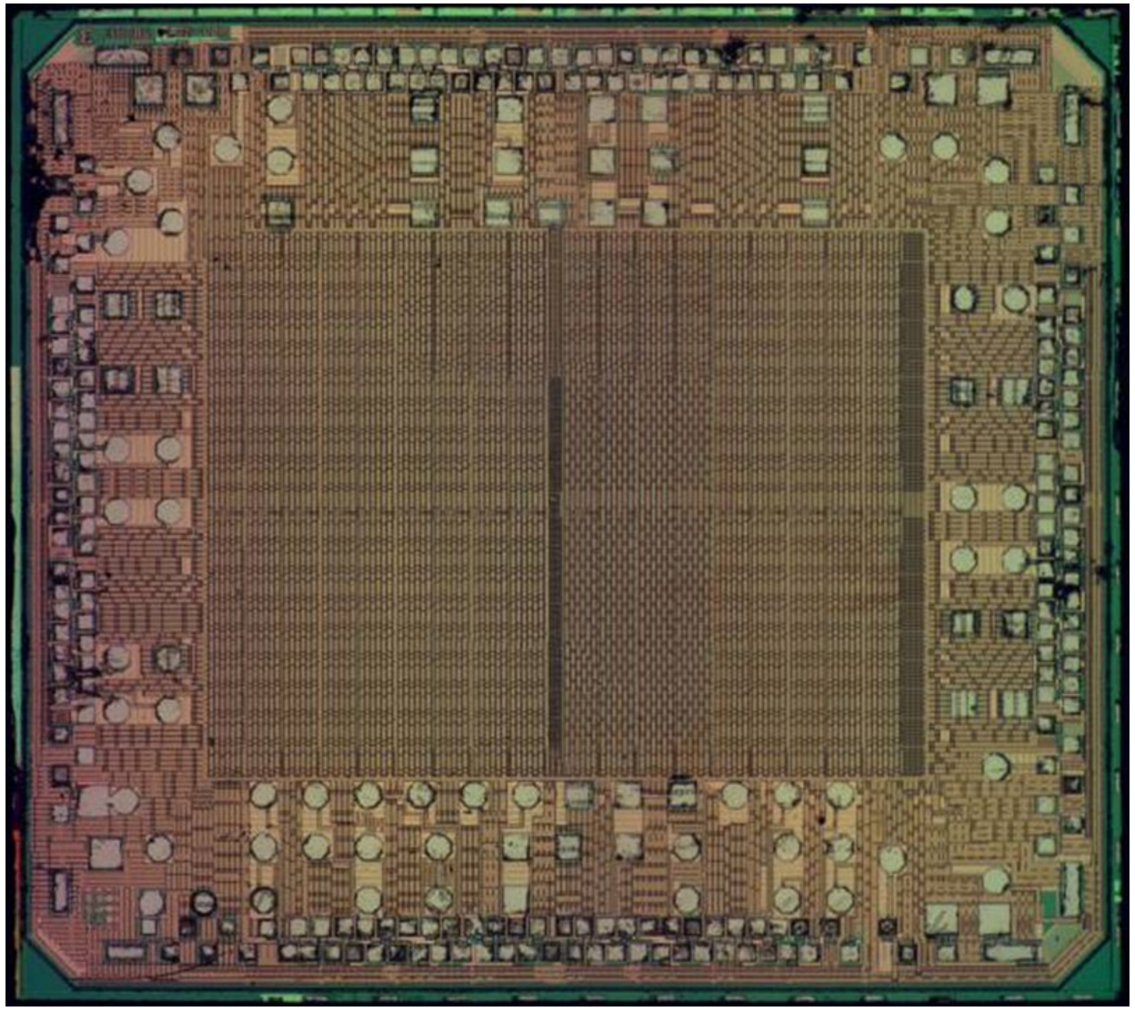
EECE 2323 Digital Design Lab

Taught by the Teaching Assistants

- Section 1: Tuesday, 08:00 AM to 10:00 AM
  - Primary TA: Farzad Niknia
  - TA: Ziyu Liu
  - TA: Meenavalli Vasundhara Rao
  - Prof. Thomas Consi
- Section 2: Friday, 11:30 AM to 01:30 PM
  - Primary TA: Ziyu Liu
  - TA: Farzad Niknia
  - TA: Meenavalli Vasundhara Rao
  - Prof. Thomas Consi
- Section 3: Tuesday, 10:15 AM to 12:15 PM
  - Primary TA: Yiming Xie
  - TA: Soorya Vijayaragavan
  - TA: Meenavalli Vasundhara Rao
  - Prof. John Kimani
- Section 4: Friday, 09:15 AM to 11:15 AM
  - Primary TA: Soorya Vijayaragavan
  - TA: Yiming Xie
  - TA: Shiyue Hou
  - Prof. John Kimani
- Section 5: Friday, 01:45 PM to 03:45 PM
  - Primary TA: Jinkun Zhang
  - TA: Kiran Keshav Kesava Sundaram
  - TA: Iris Wang
  - Prof. Naveen Naik Sapavath
- Section 6: Tuesday, 12:30 PM to 02:30 PM
  - Primary TA: Shiyue Hou
  - TA: Veera Venkata Ram Kiran Jutta
  - TA: Iris Wang
  - Prof. Drew Zagieboylo
- Section 7: Tuesday, 02:45 PM to 04:45 PM
  - Primary TA: Kiran Keshav Kesava Sundaram
  - TA: Jinkun Zhang
  - TA: Iris Wang
  - Prof. Drew Zagieboylo
- Section 8: Thursday, 09:15 AM to 11:15 AM
  - Primary TA: Veera Venkata Ram Kiran Jutta
  - TA: Iris Wang
  - Prof. Naveen Naik Sapavath

Today's Topics

- Hand-Wiring Circuits with Logic Integrated Circuits (IC's or Chips)
- Programmable Logic Devices (PLD)
  - Programmable Array Logic
  - Field-Programmable Gate Array (FPGA)
- System on a Chip (SOC)
- Xilinx Zync SOC
- Tul Pynq Board
- Implementing circuits on an FPGA
  - Hardware Definition Language (HDL)
  - Conventional Program vs. FPGA Code
  - FPGA Development tools
- Lab 0
- Reading: Harris & Harris Appendix A.3 available online
  http://pages.hmc.edu/harris/ddca/ddcarv/DDCArv_AppA_Harris.pdf

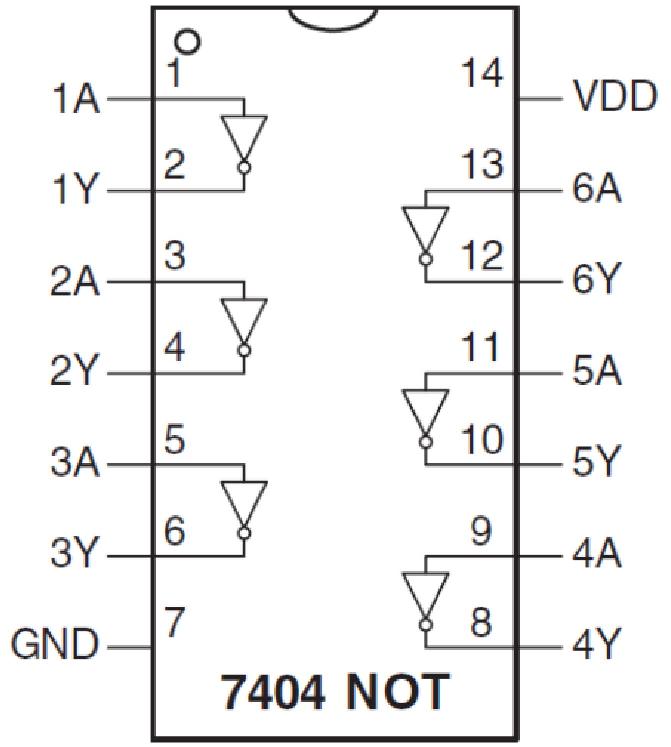


Field-Programmable Gate Array (FPGA)

https://siliconpr0n.org/archive/doku.php?id=azonenberg:xilinx:xc3s50a

Hand-Wiring Logic Circuits with Logic IC's I

- Logic Chips: Integrated circuits with small number of logic functions
- Medium-Scale integration (Hundreds of Transistors)
- Good for prototyping circuits
- Industry-Standard Part Numbering Scheme: 74AAXXX
    - AA = technology used in implement transistors
    - XXX = part number (indicates logic function)
- Key specifications (from the Data Sheet)
    - Logic function (truth table)
    - Pin-out, function of the pins of the chip
    - Electrical characteristics: power, voltage levels, speed of signals, etc.



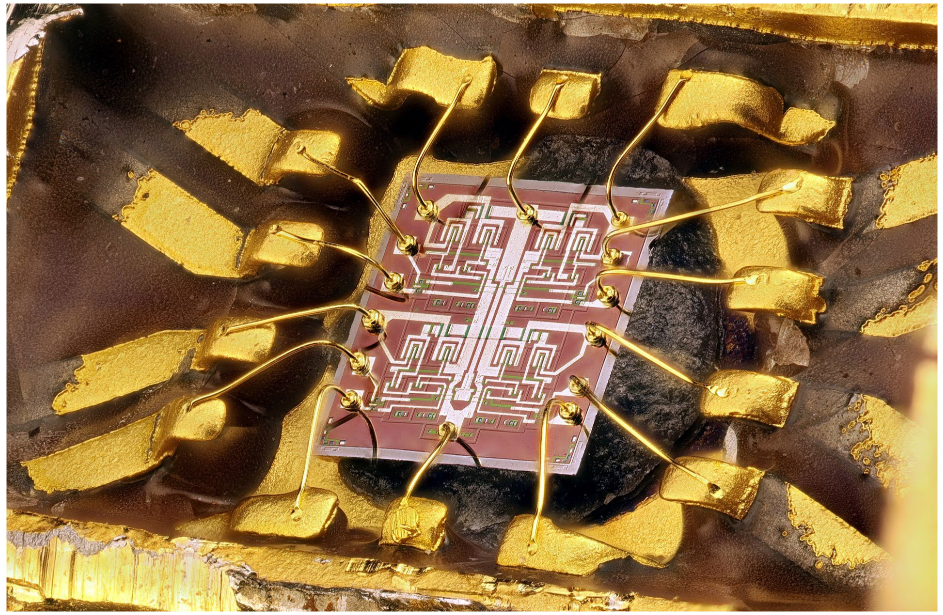
7404 Hex Inverter
6 independent inverters
Common power connections

Digital Design and Computer Architecture, RISC-V ed. © 2022 Elsevier Inc.

Hand-Wiring Logic Circuits with Logic IC's II

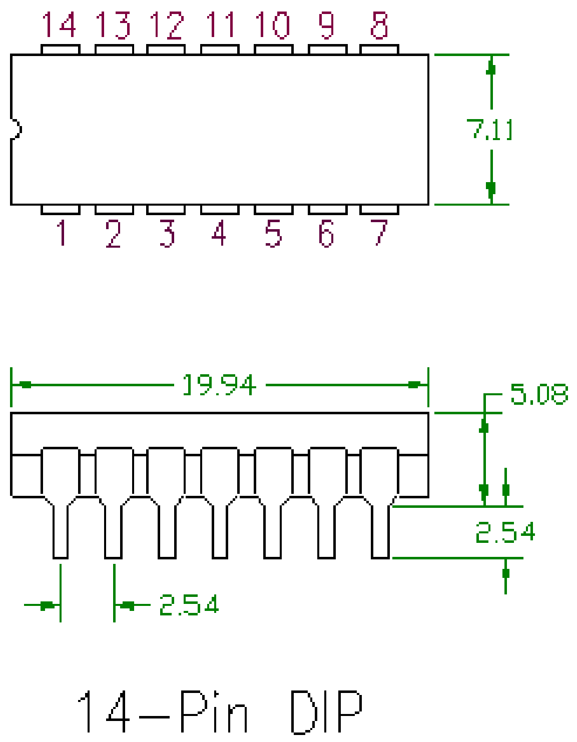
https://www.makestore.in/product/gate-74hc00-nand-7400/

7400 Quad NAND Gate in 14-pin Dual In-line (DIP) Package



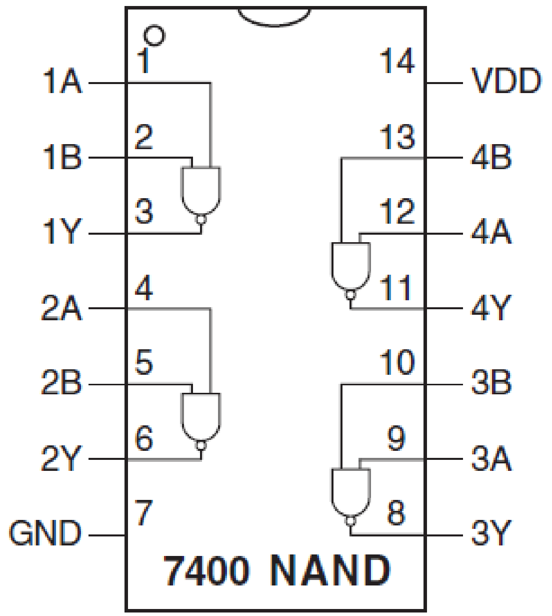
Die of the 7400 Quad NAND Gate
Plastic removed to show circuit die and connecting wires

Wikipdia



Dimensions of 14-Pin Chip Package

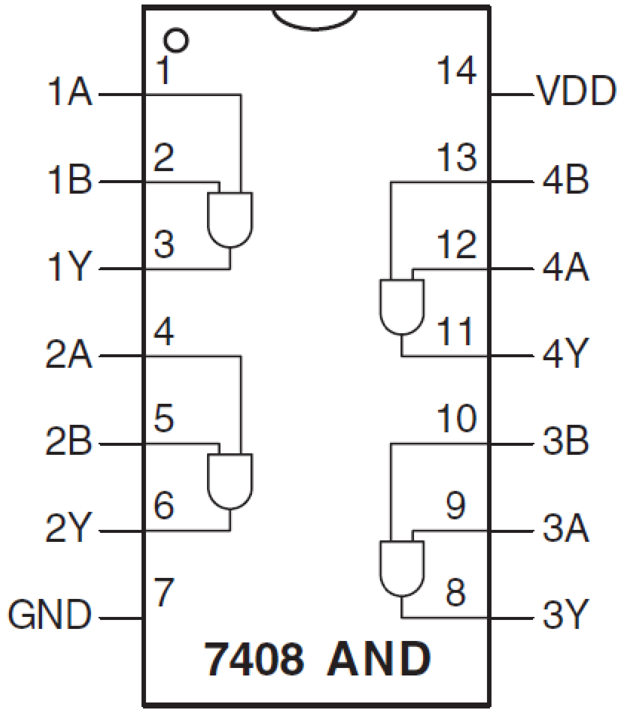
https://www.futurlec.com/74/IC7404.shtml

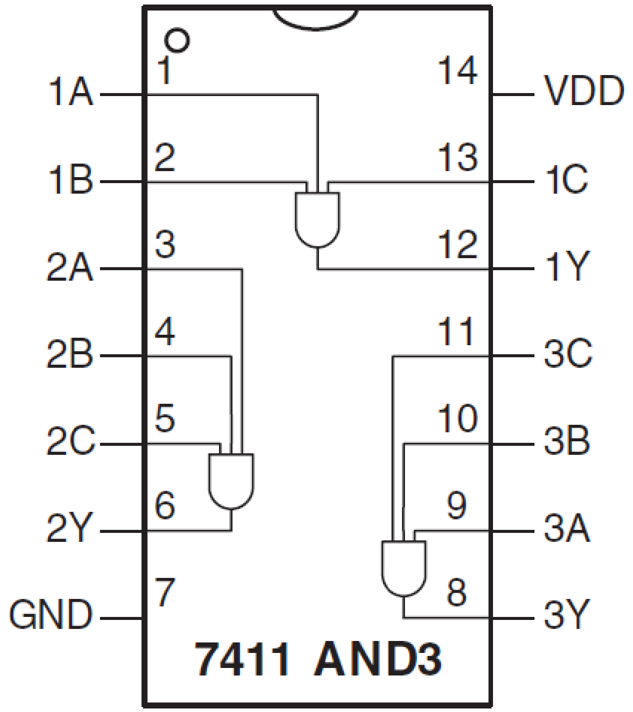


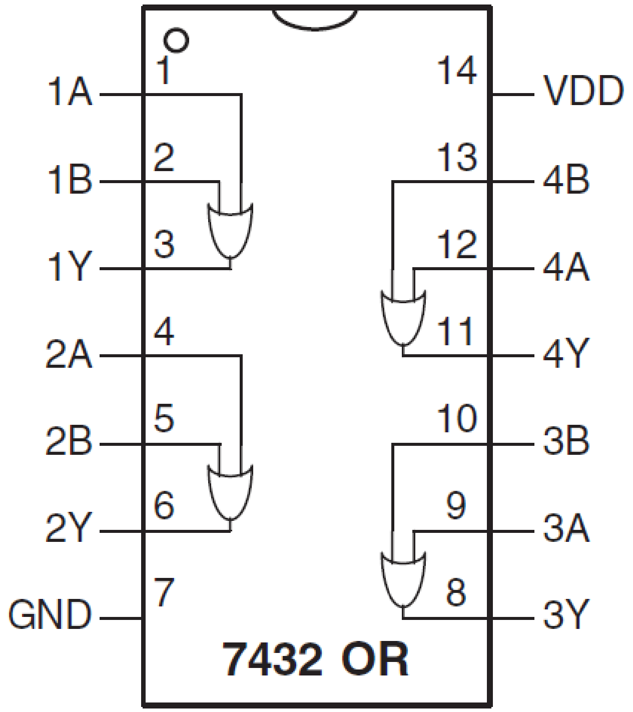
Functional diagram of 7400 chip

Digital Design and Computer Architecture, RISC-V ed. © 2022 Elsevier Inc.

Hand-Wiring Logic Circuits with Logic IC's III

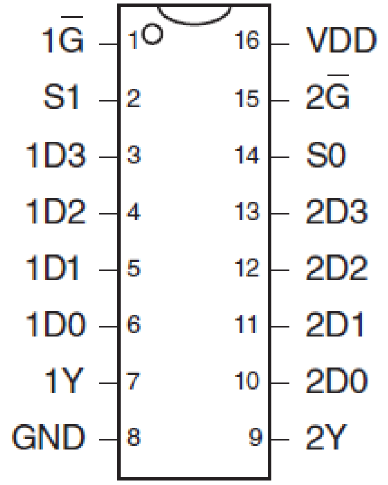






Digital Design and Computer Architecture, RISC-V ed. © 2022 Elsevier Inc.

Hand-Wiring Logic Circuits with Logic IC's IV

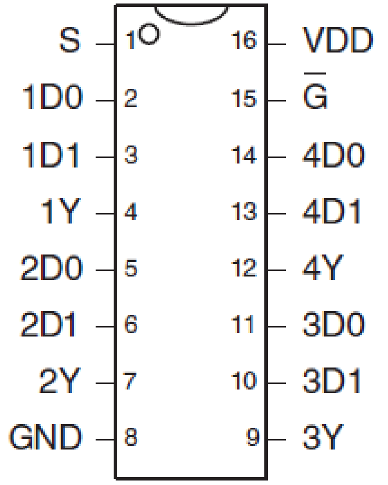


74153 4 :1 Mux

Two 4:1 Multiplexers

$D_{3:0}$:	data
$S_{1:0}$:	select
Y:	output
Gb:	enable

```
always_comb
  if (1Gb) 1Y = 0;
  else     1Y = 1D[S];
always_comb
  if (2Gb) 2Y = 0;
  else     2Y = 2D[S];
```

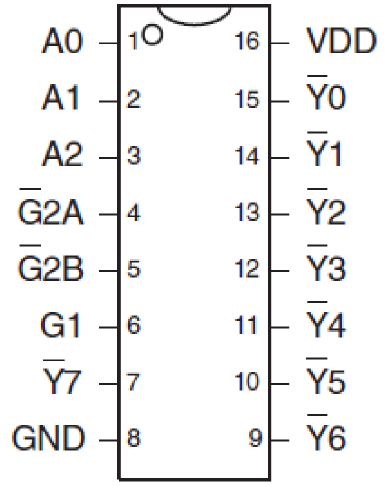


74157 2 :1 Mux

Four 2:1 Multiplexers

$D_{1:0}$:	data
S:	select
Y:	output
Gb:	enable

```
always_comb
  if  (Gb)  1Y = 0;
  else      1Y = S ? 1D[1] : 1D[0];
  if  (Gb)  2Y = 0;
  else      2Y = S ? 2D[1] : 2D[0];
  if  (Gb)  3Y = 0;
  else      3Y = S ? 3D[1] : 3D[0];
  if  (Gb)  4Y = 0;
  else      4Y = S ? 4D[1] : 4D[0];
```

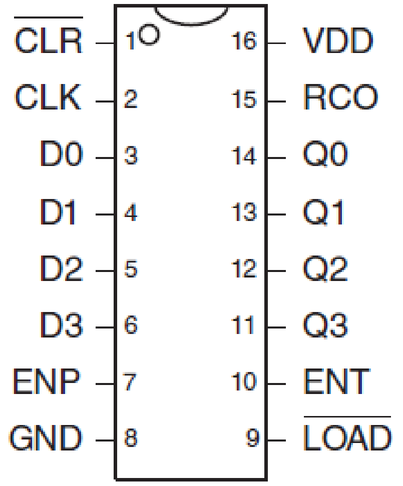


74138 3 :8 Decoder

3:8 Decoder

$A_{2:0}$:	address
$Yb_{7:0}$:	output
G1:	active high enable
G2:	active low enables

G1	$\overline{G2A}$	$\overline{G2B}$	A2:0	Y7:0
0	x	x	xxx	11111111
1	1	x	xxx	11111111
1	0	1	xxx	11111111
1	0	0	000	11111110
1	0	0	001	11111101
1	0	0	010	11111011
1	0	0	011	11110111
1	0	0	100	11101111
1	0	0	101	11011111
1	0	0	110	10111111
1	0	0	111	01111111



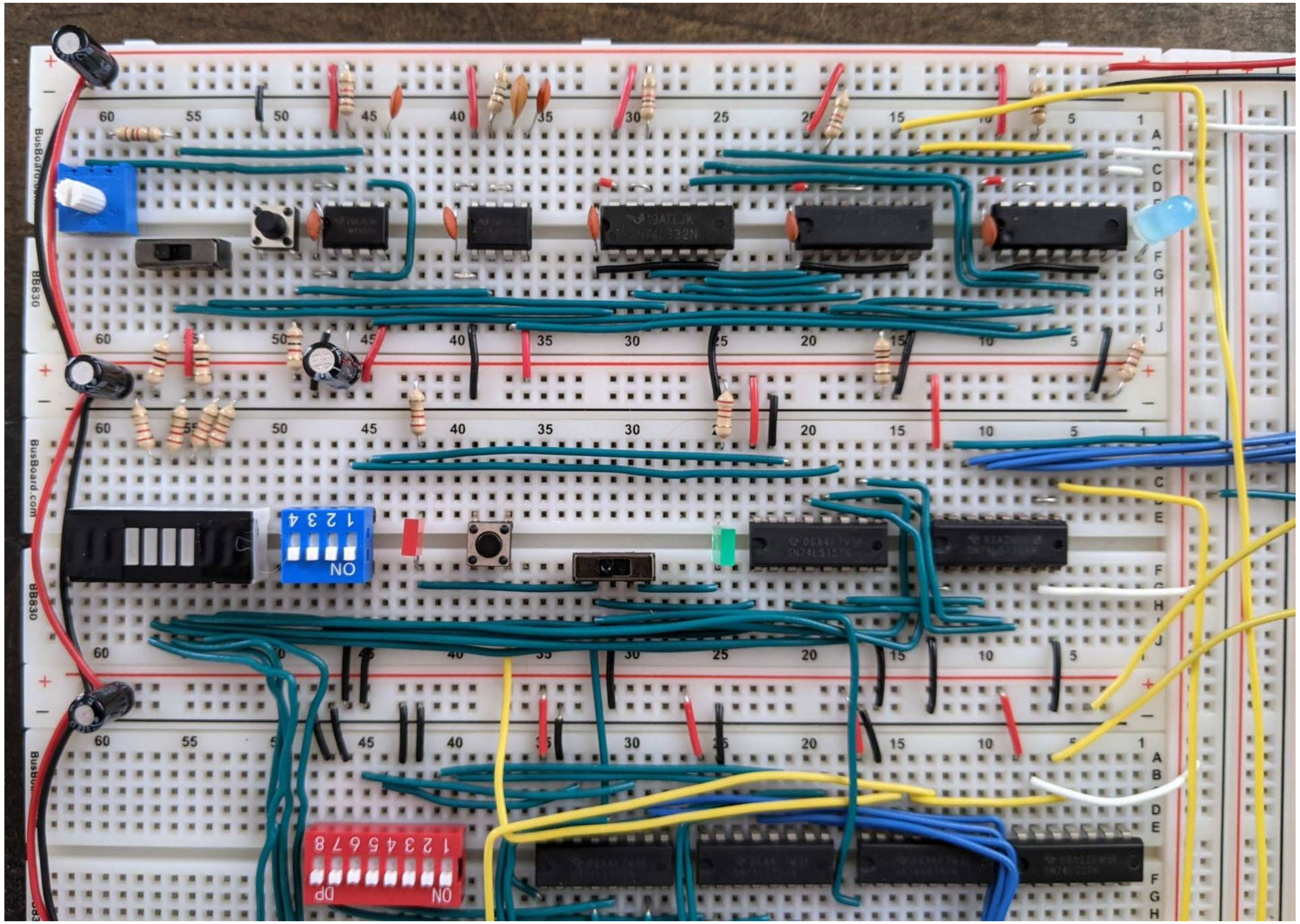
74161/163 Counter

4-bit Counter

CLK:	clock
$Q_{3:0}$:	counter output
$D_{3:0}$:	parallel input
CLRb:	async reset (161)
	sync reset (163)
LOADb:	load Q from D
ENP, ENT:	enables
RCO:	ripple carry out

```
always_ff @(posedge CLK) // 74163
  if (~CLRb) Q <= 4'b0000;
  else if (~LOADb) Q <= D;
  else if (ENP & ENT) Q <= Q+1;

assign RCO = (Q == 4'b1111) & ENT;
```

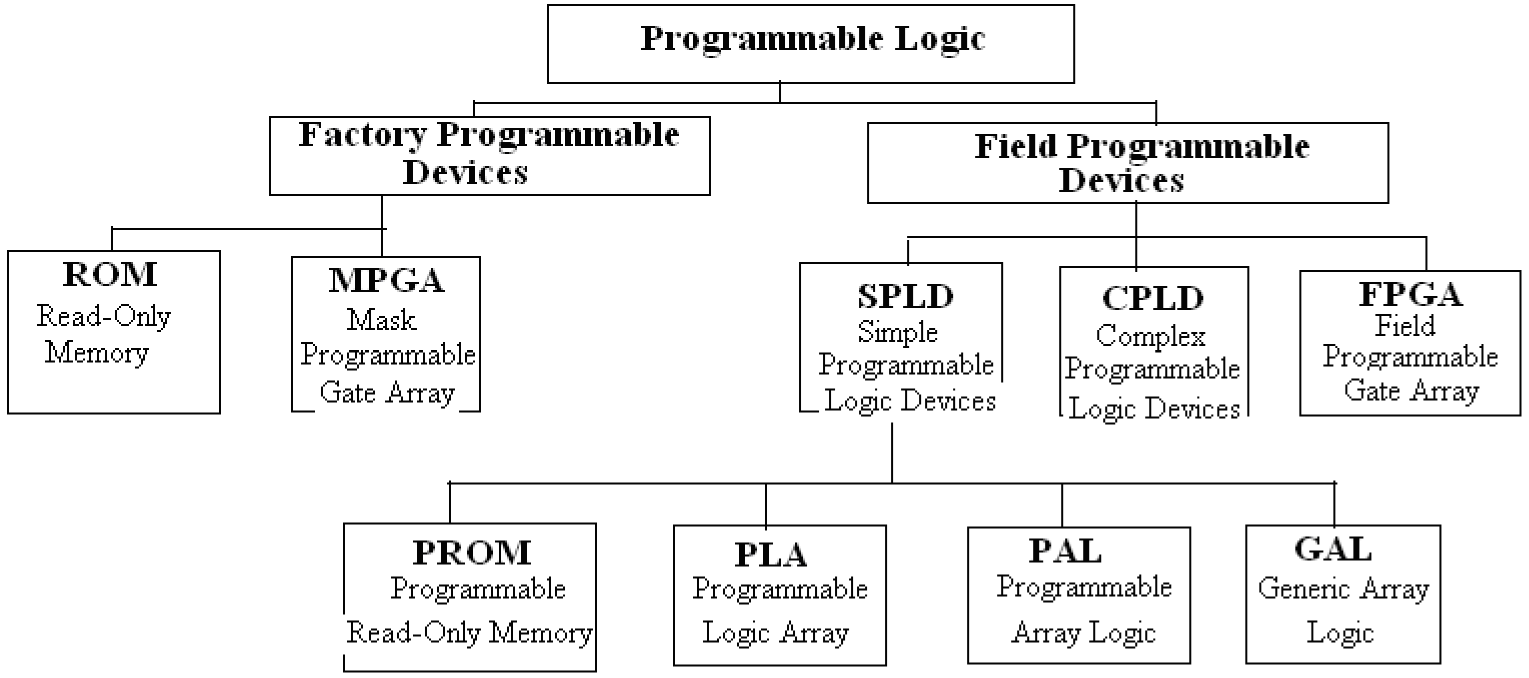
Hand-Wiring
Logic Circuits with Logic IC's V



https://austinmorlan.com/posts/8bit_breadboard/

Programmable Logic Devices

- An array of logic blocks
- Programmable interconnections
  - Factory
  - In-field
- Can implement a wide variety of logic circuits
- Inexpensive
  - Mass produced
  - Greatly reduce number of discrete components in a design
  - No need for wiring of components
- Increased reliability
- Increased circuit density



Programmable Logic
Factory Programmable Devices
Field Programmable Devices
ROM Read-Only Memory
MPGA Mask Programmable Gate Array
SPLD Simple Programmable Logic Devices
CPLD Complex Programmable Logic Devices
FPGA Field Programmable Gate Array
PROM Programmable Read-Only Memory
PLA Programmable Logic Array
PAL Programmable Array Logic
GAL Generic Array Logic

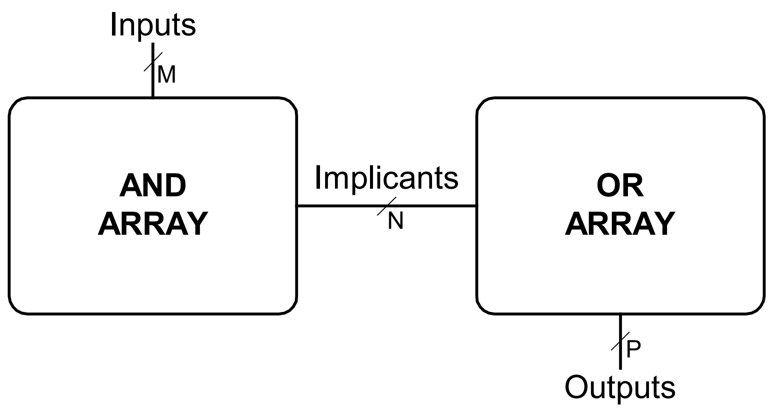
https://alternativepaymentmethods.medium.com/programmable-logic-device-plds-b136071c8b82

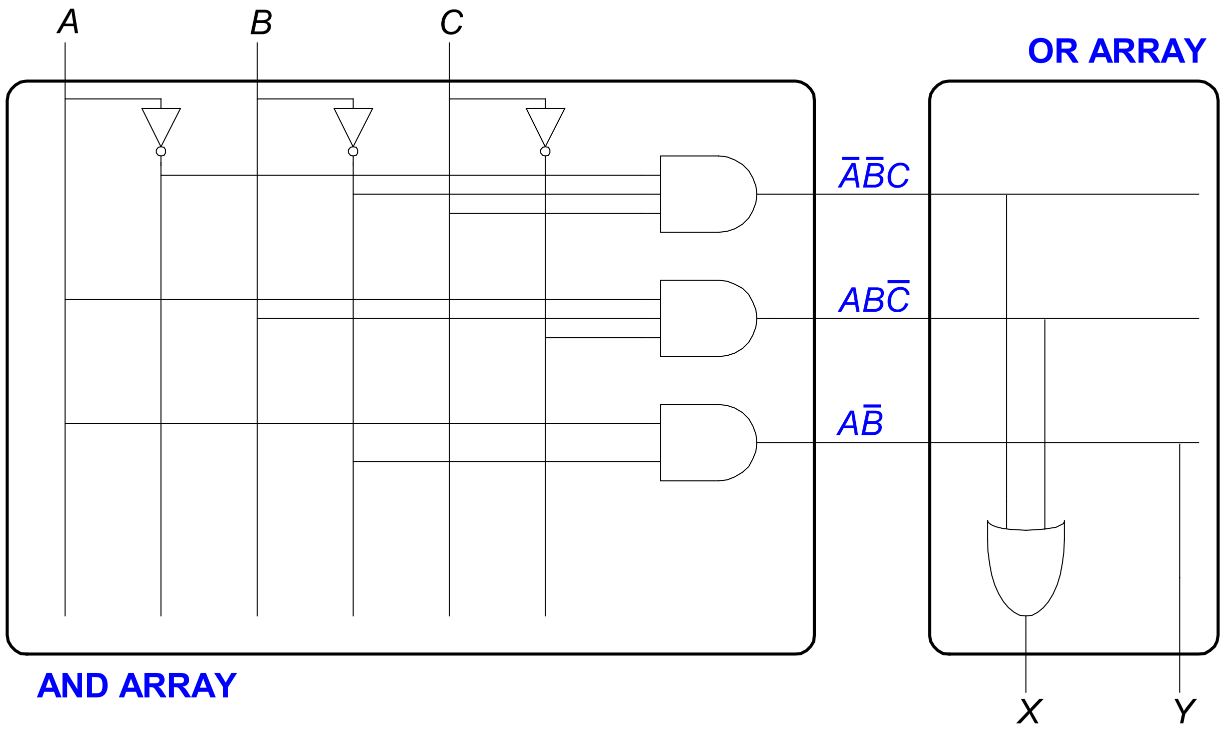
Programmable Array Logic (PAL)

- Combinational Logic Circuits
- Sum of Products Form

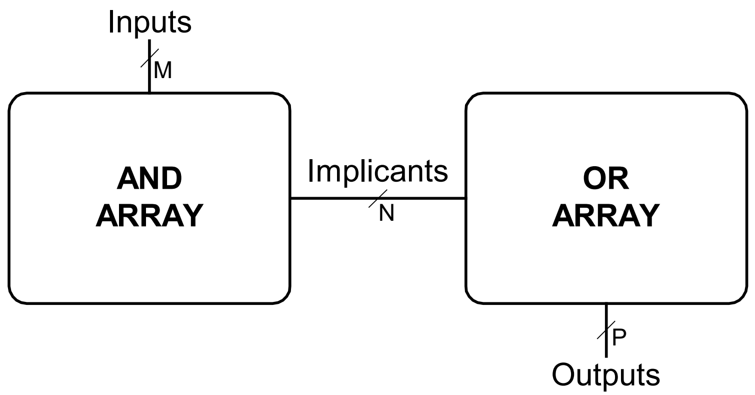
Simple Example:

- 3 inputs, A, B, C
- All inputs inverted
- AND Array
- OR Array

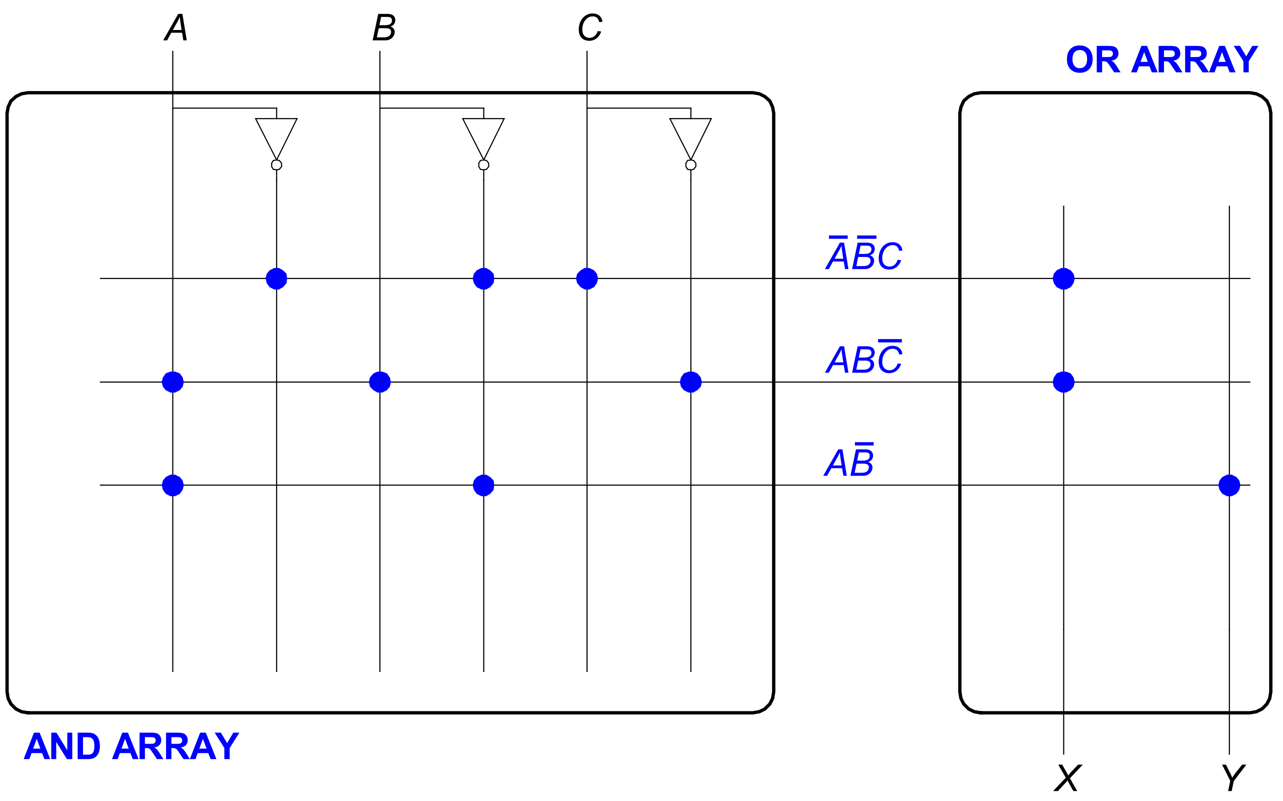




PAL: Dot Notation & Programming

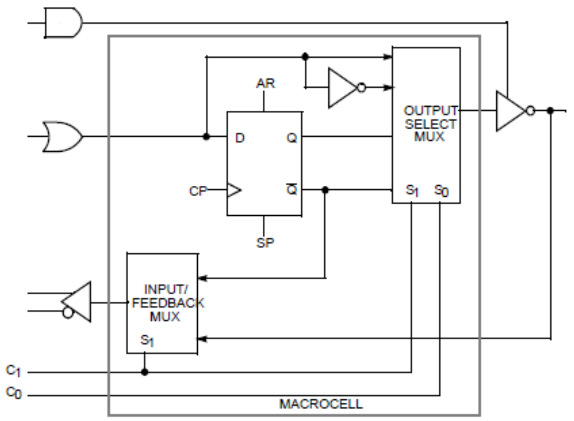


- Designer specifies connections
- Connections designated by a dot
- Factory Programmed
- One-Time programmed
- UV-erasable

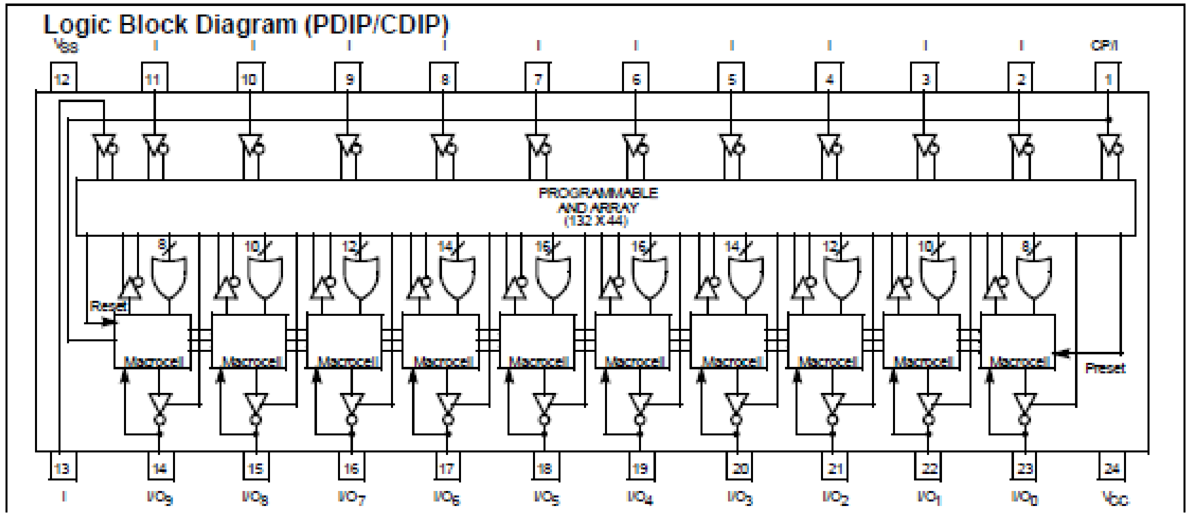


A Real PAL

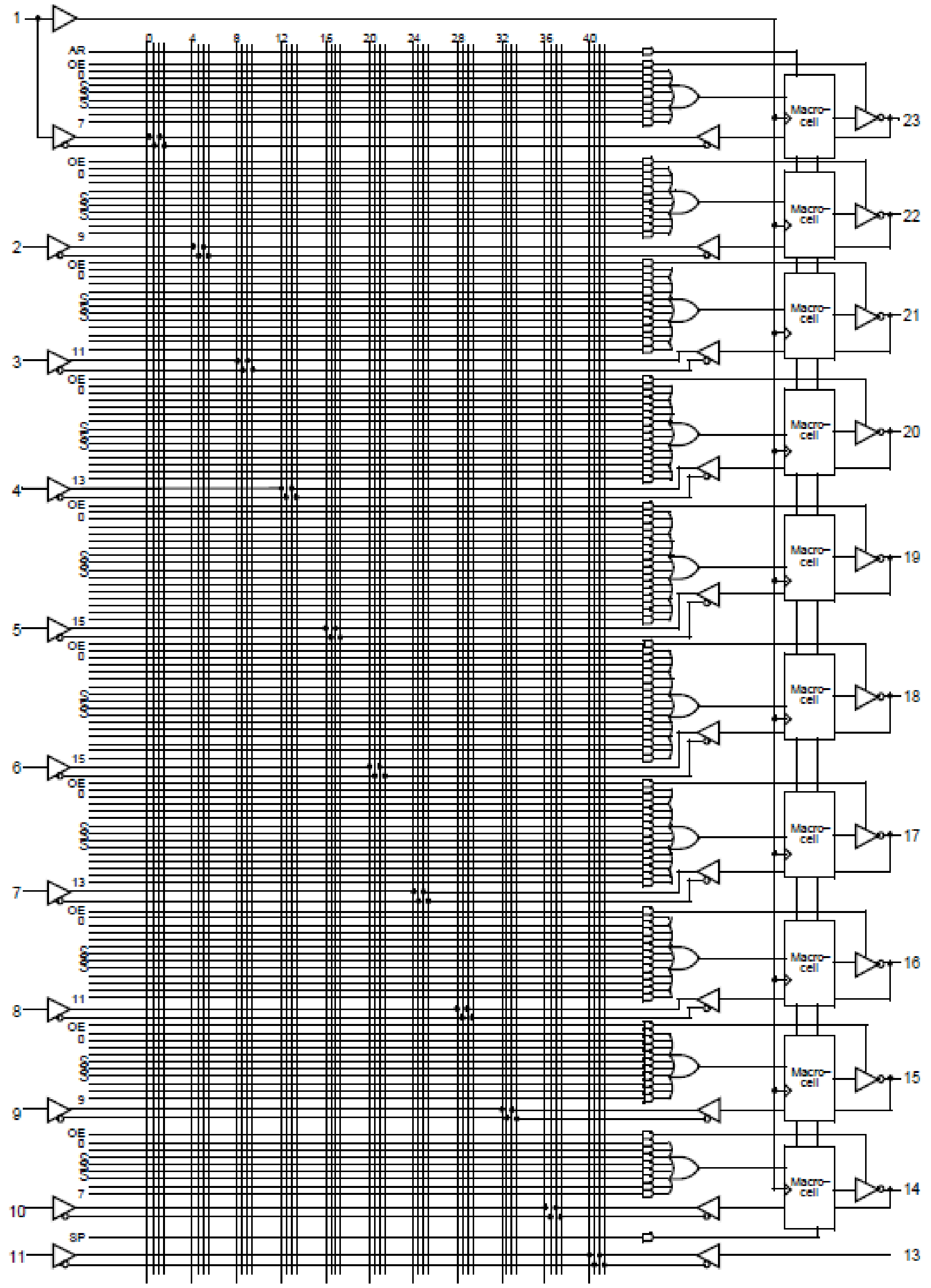
- Cypress Semiconductor
- PAL CE22V10
- Erasable Reprogrammable PAL
- Macrocells on Outputs



Macrocell


Logic Block Diagram (PDIP/CDIP)
PROGRAMMABLE AND ARRAY (132 X 44)
Macrocell
Reset
Preset
CP/I
VSS
VCC

Pinout of DIP Version



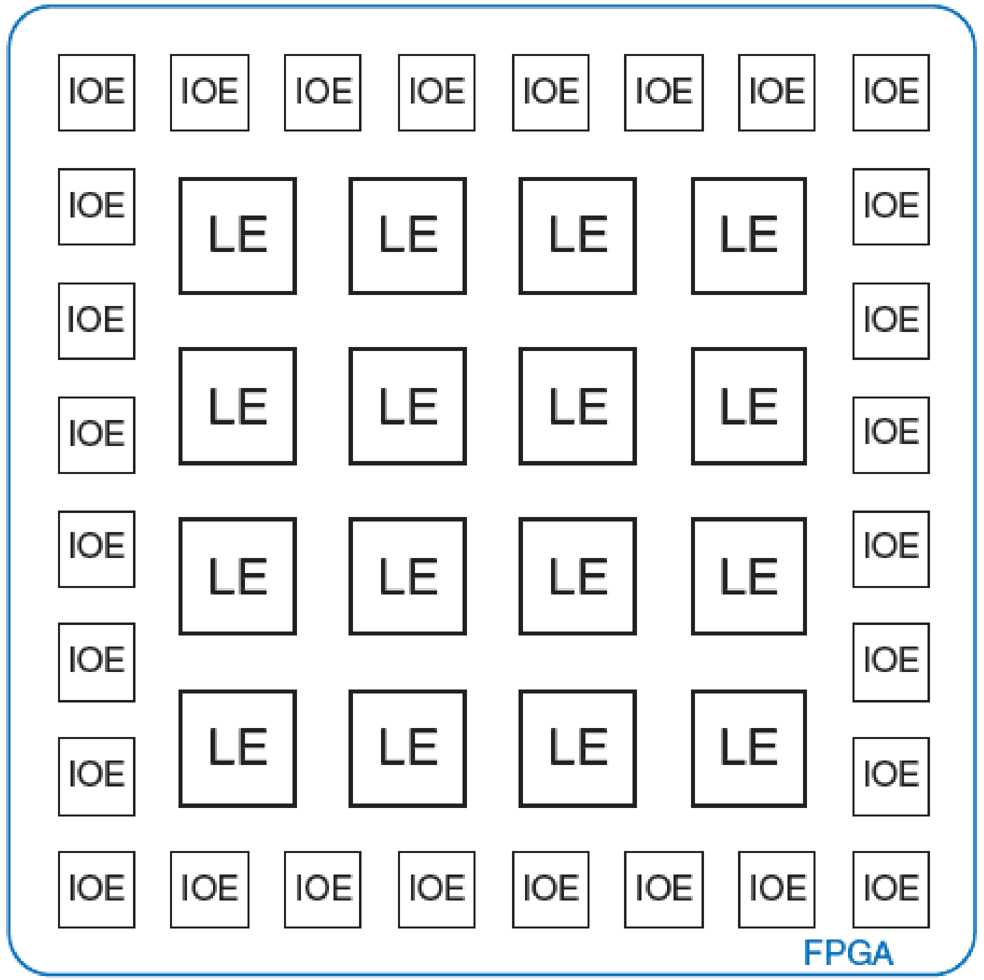
Functional Logic Diagram

Cypress Semiconductor, Document #: 38-03027 Rev. *B

Field-Programmable Gate Arrays (FPGAs)

Composed of:

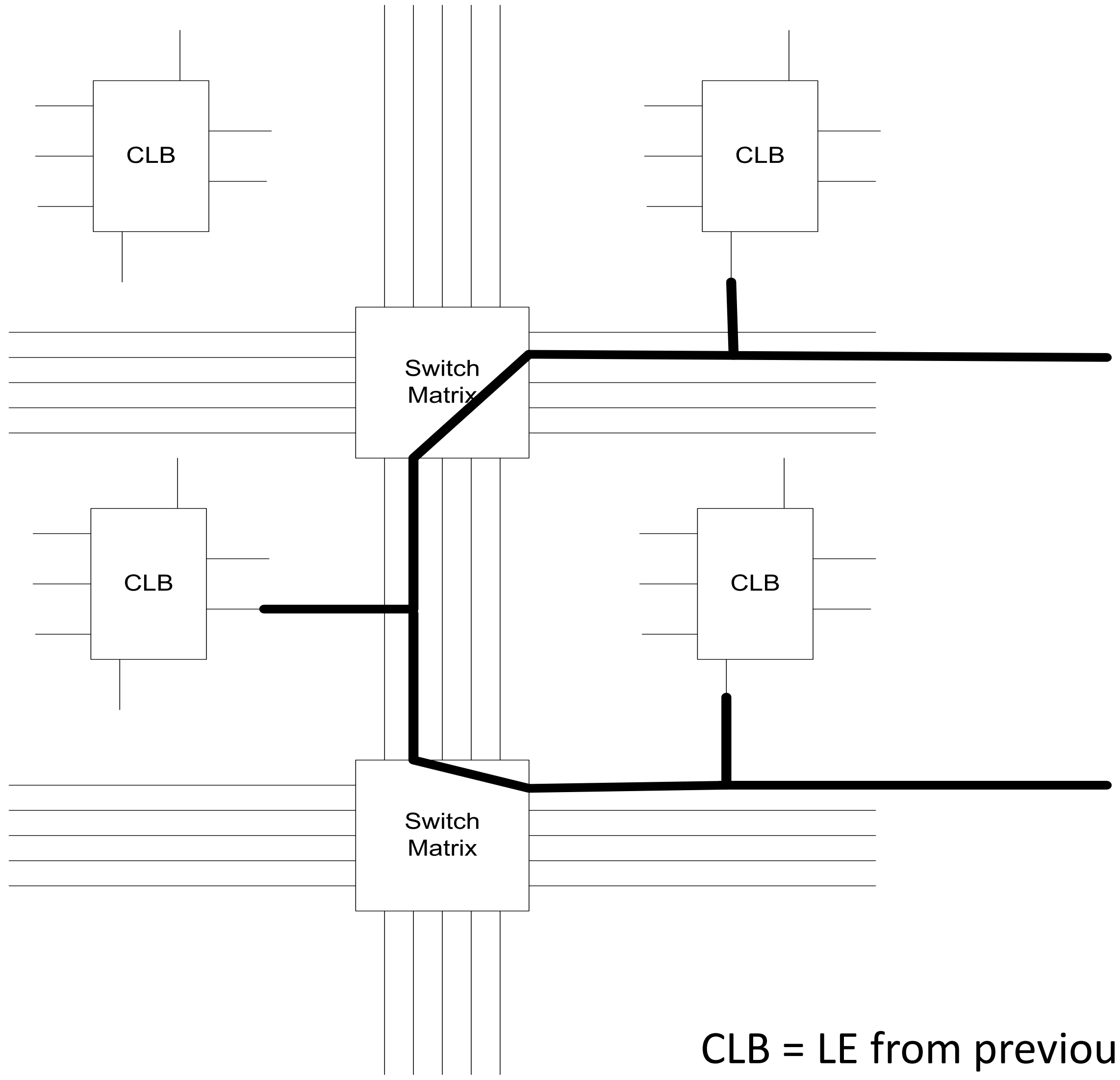
- **LEs** (Logic elements): perform logic
  - Configurable, aka **CLE**
- **IOEs** (Input/output elements): interface with outside world
- **Programmable interconnection:** connect LEs and IOEs
- Some FPGAs include other building blocks such as multipliers and RAMs
- Logic Elements Composed of:
  - **LUTs** (lookup tables): perform combinational logic
  - **Flip-flops:** perform sequential logic
  - **Multiplexers:** (Mux) connect LUTs and flip-flops



General FPGA Layout

Interconnect: Switch Matrixes and Programmable Interface Points

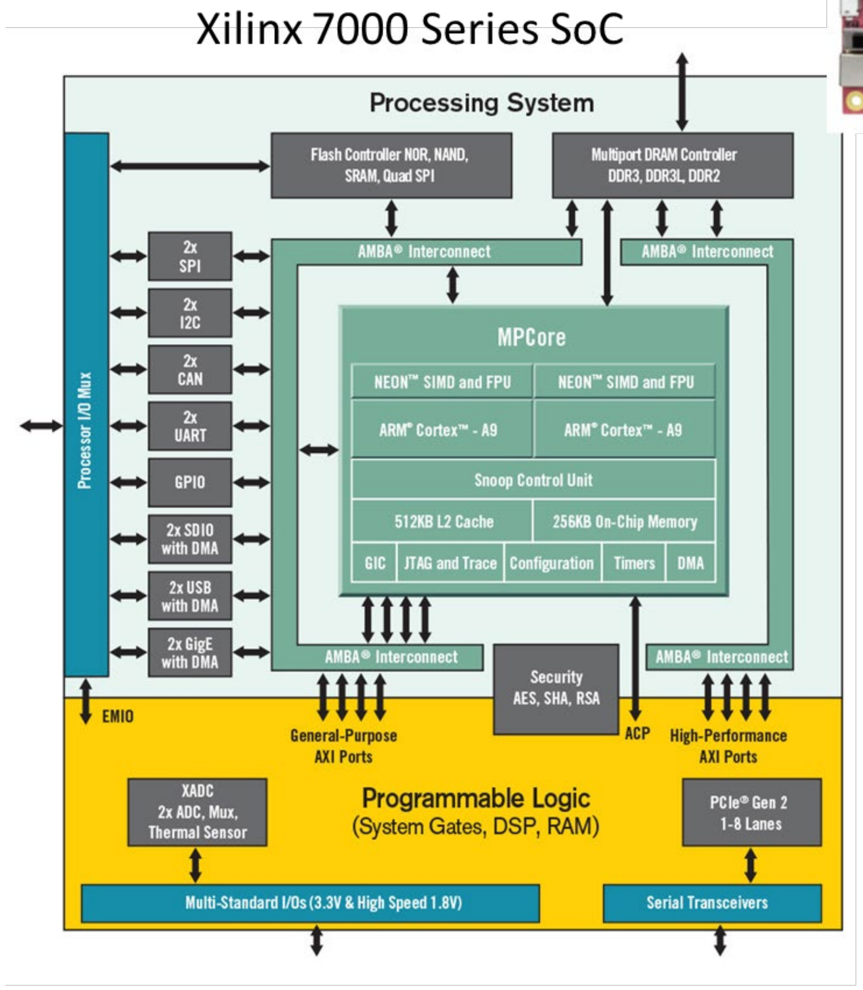
- FPGAs are volatile
- The interconnections are lost on power-down
- On power-up either:
  - Reprogram
  - Boot loader
    - Non-volatile memory
    - Stand-alone applications

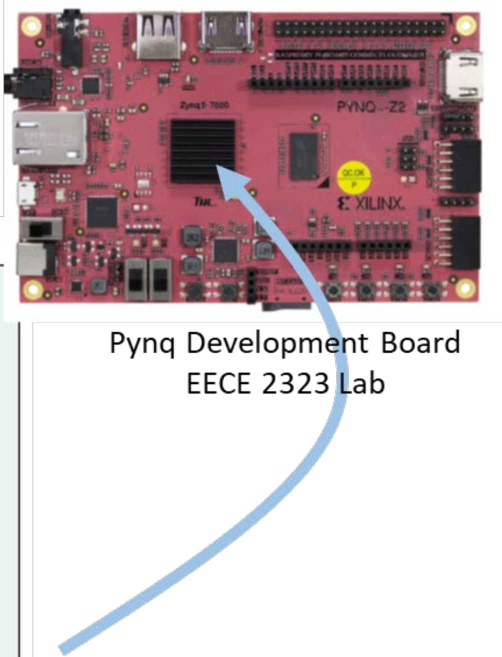


CLB = LE from previous slide

Selected Specifications: Xilinx Zync 7000 series SoC
(SoC = System on a Chip)

- Pynq board uses XC7Z020 chip
- 85,000 logic cells
  - LUT, Flip-Flop, Mux
- 4.9 Mbits of RAM in 36k blocks
- 220 Digital Signal Processing Slices
- 2, 12-bit Analog to Digital Converters
- Dual-Core Cortex-A9 Processor
- Package size: 17mm X 17mm
- Number of pins: 400





Pynq Development Board
EECE 2323 Lab

Computer Program vs Hardware Description Language (HDL)

Computer Program

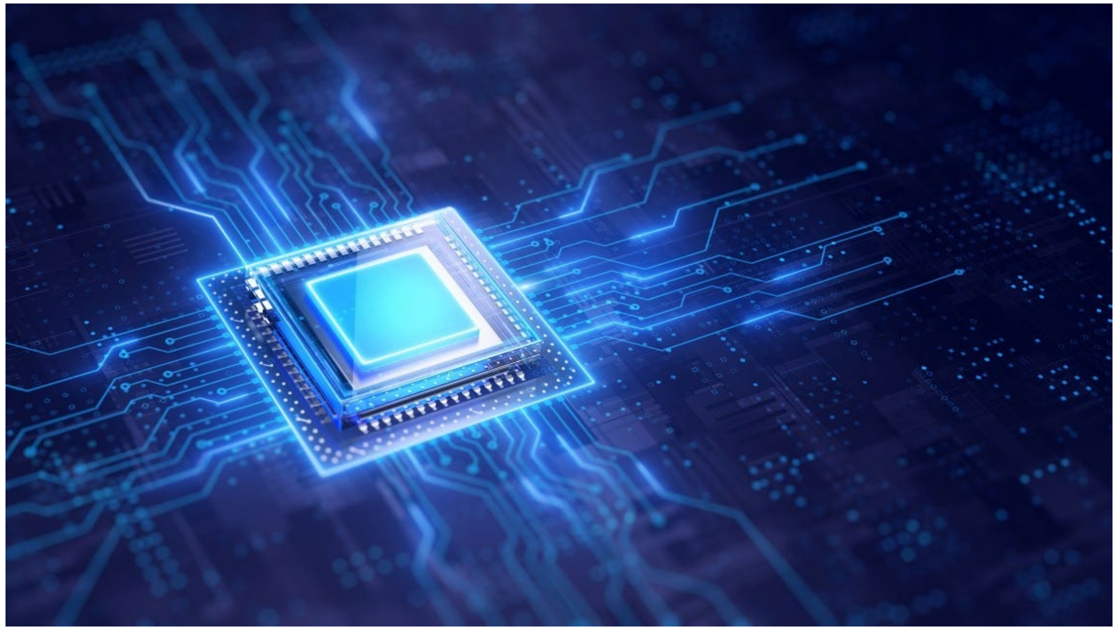
- A list of instructions
- Executed by a general purpose computing device
  - Microprocessor, microcontroller
- Hardware is fixed
- Change function by changing the program
- Development
  - Write program
  - Compile to executable code

https://towardsdatascience.com/6-python-projects-you-can-finish-in-a-weekend-f53552279cc

Computer Program vs Hardware Description Language (HDL)

Hardware Description Language (HDL)

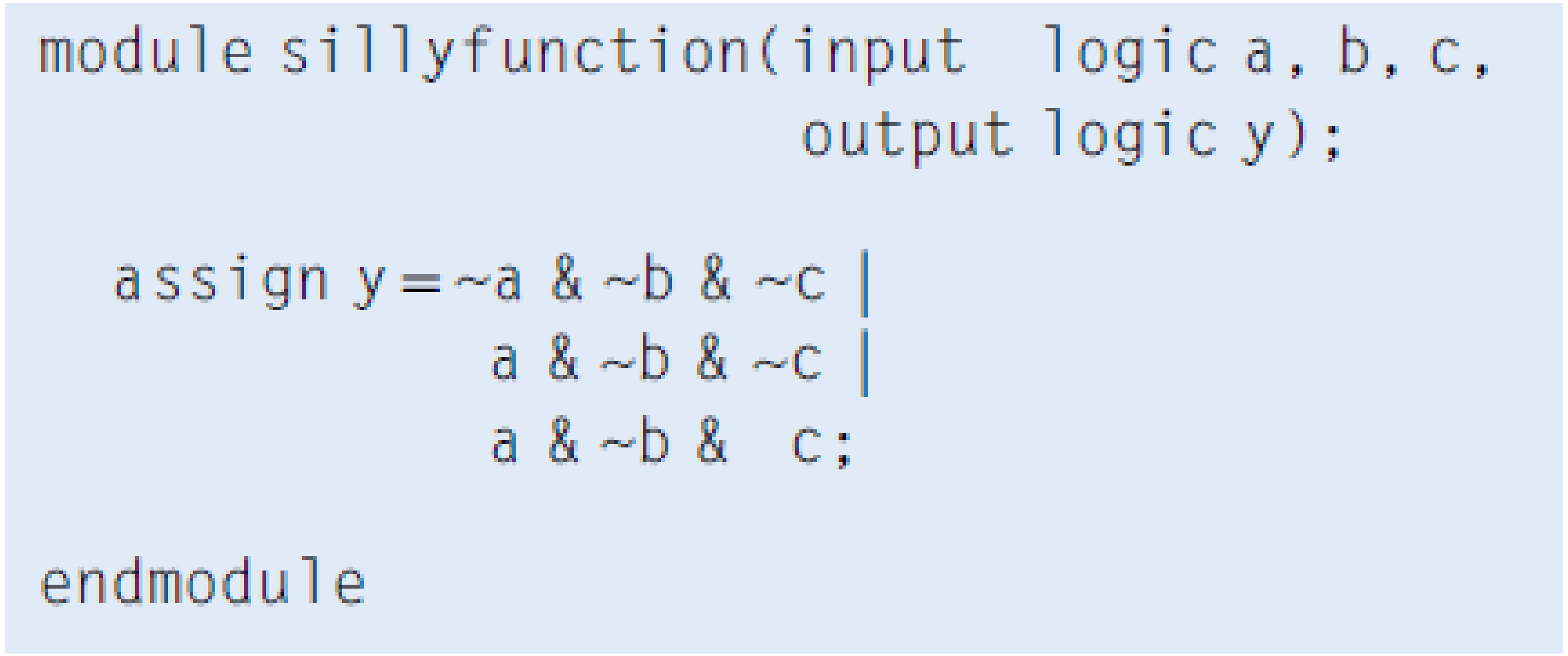
- A description of a circuit
  - Circuit components (e.g. gates)
  - Interconnections
- Causes an FPGA to configure its internal hardware to create a specific circuit
- Circuit performs the desired function
- Change function by creating a new hardware configuration via a new HDL


https://www.microchip.com/en-us/products/fpgas-and-plds

- Major HDLs
  - Verilog, System Verilog
  - VHDL
- High-Level Code Entry (e.g. "C")
- Schematic Diagram Code Entry (Lab 0)

System Verilog Syntax I

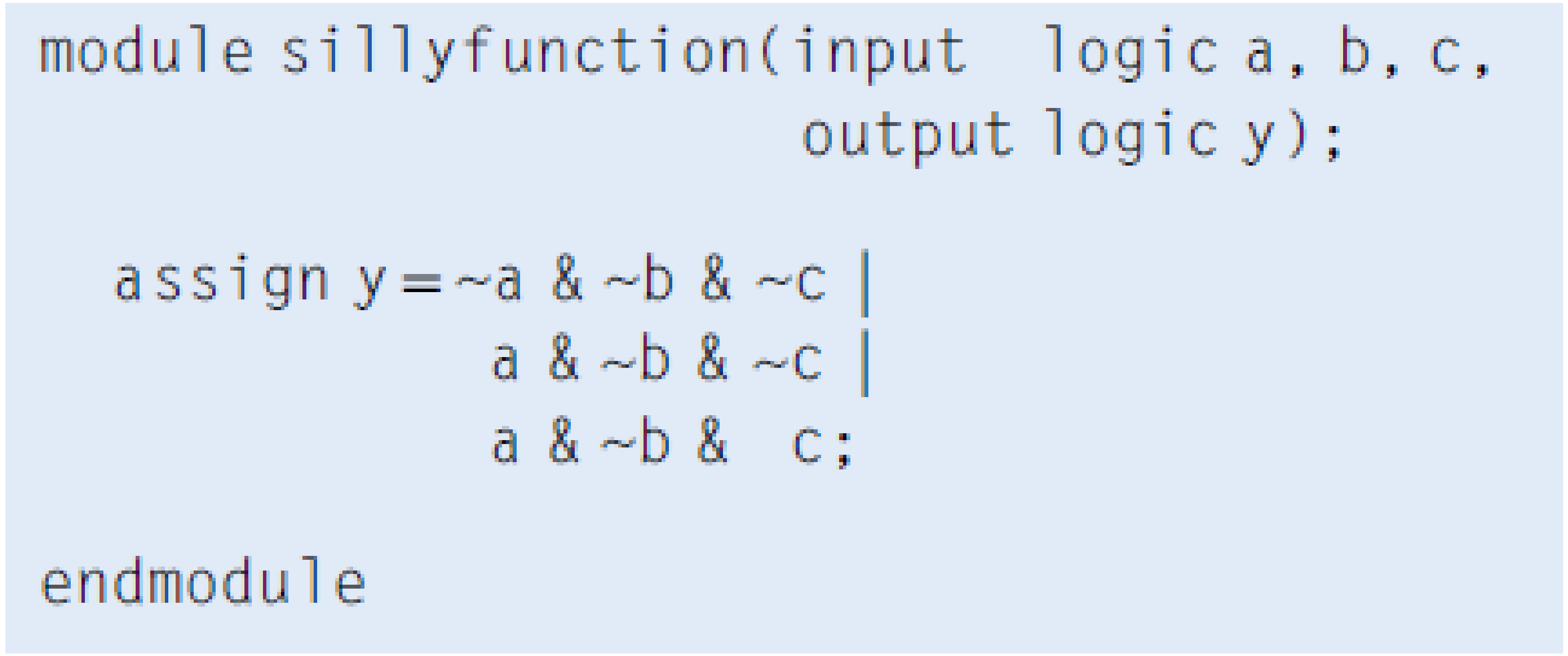
- Declaration Line
  - *module* keyword
  - Name of module
  - List of input and output ports
  - Port declaration
    - Data type
    - Number of bits
  - End in semicolon
- Body of module is expression that represents the Boolean equation
  - Statements end with ;
- Module ends with endmodule keyword



```
module sillyfunction(input  logic a, b, c,
                     output logic y);

  assign y = ~a & ~b & ~c |
              a & ~b & ~c |
              a & ~b &  c;

endmodule
```

- Boolean operators
  - & = AND
  - | = OR
  - ~ = complement

System Verilog Syntax II



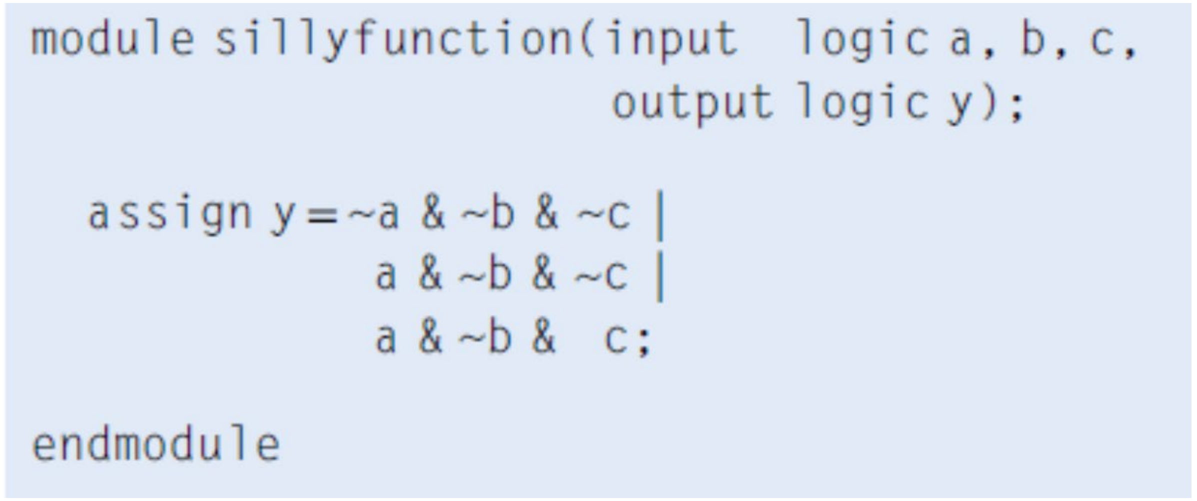
```
module sillyfunction(input  logic a, b, c,
                     output logic y);

  assign y = ~a & ~b & ~c |
              a & ~b & ~c |
              a & ~b &  c;

endmodule
```

assign is used for combinational logic.

Drives inputs to outputs through the logic functions whenever the inputs change. Operates continuously.

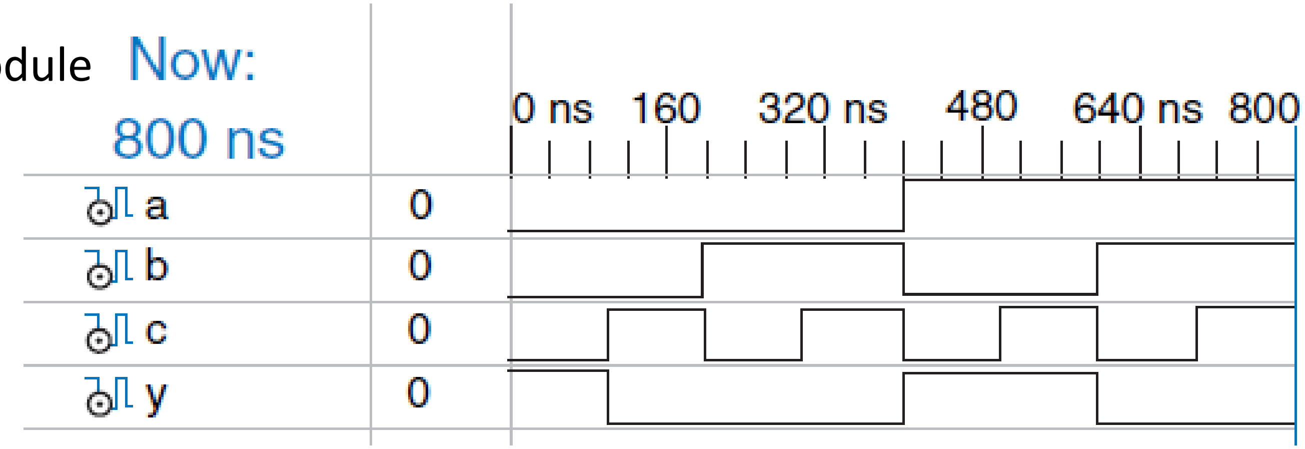
Simulation

- The ultimate goal of an HDL is to generate a circuit
- Testing a complex real circuit can be
  - Difficult
  - Time consuming
- Simulation a way to test HDL code on a computer
  - Create a software model of the HDL defined module to be tested
    - Device Under Test (DUT), Unit Under Test (UUT)
  - Testbench: a module that tests the DUT
    - Feeds DUT different inputs
    - Records or displays resultant outputs



```
module sillyfunction(input  logic a, b, c,
                               output logic y);

  assign y = ~a & ~b & ~c |
              a & ~b & ~c |
              a & ~b &  c;

endmodule
```

Module or Device Under Test (DUT)



Output of Simulation Test of Module

FPGA Source Code Processing

Synthesis

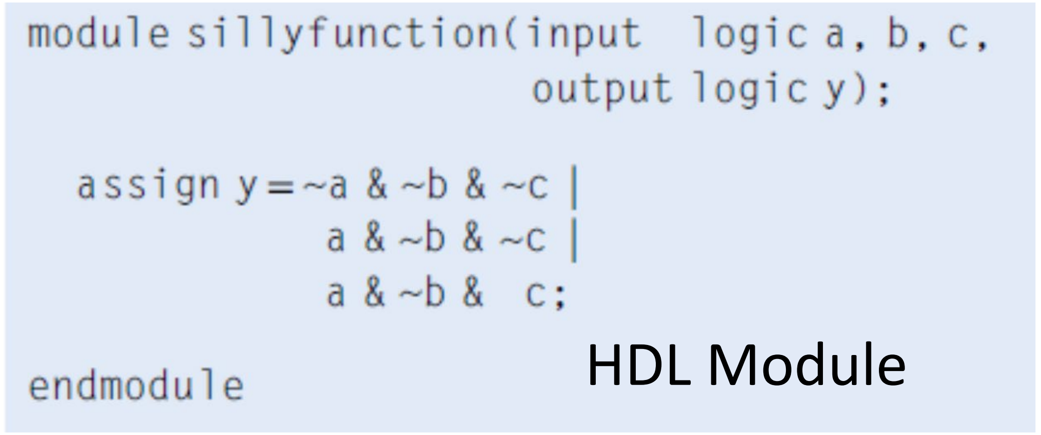
- Creating the netlist that describes the physical circuit
  - Components
  - Interconnecting wires
- Netlist can be drawn as a schematic to visualize circuit
- Simplification and Optimization performed
- HDL code designed for simulation cannot be synthesized
  - e.g. Testbench code

Implementation

- Mapping netlist onto hardware of specific FPGA
- Routing of interconnects

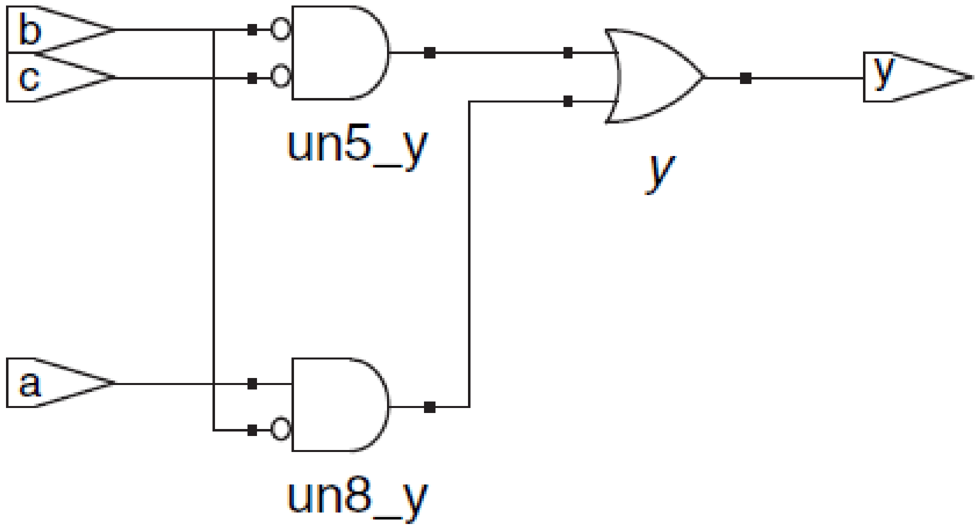
BITSTREAM generation

- Data file sent to FPGA to implement the design on actual hardware



```
module sillyfunction(input  logic a, b, c,
                        output logic y);

  assign y = ~a & ~b & ~c |
              a & ~b & ~c |
              a & ~b &  c;

endmodule
```

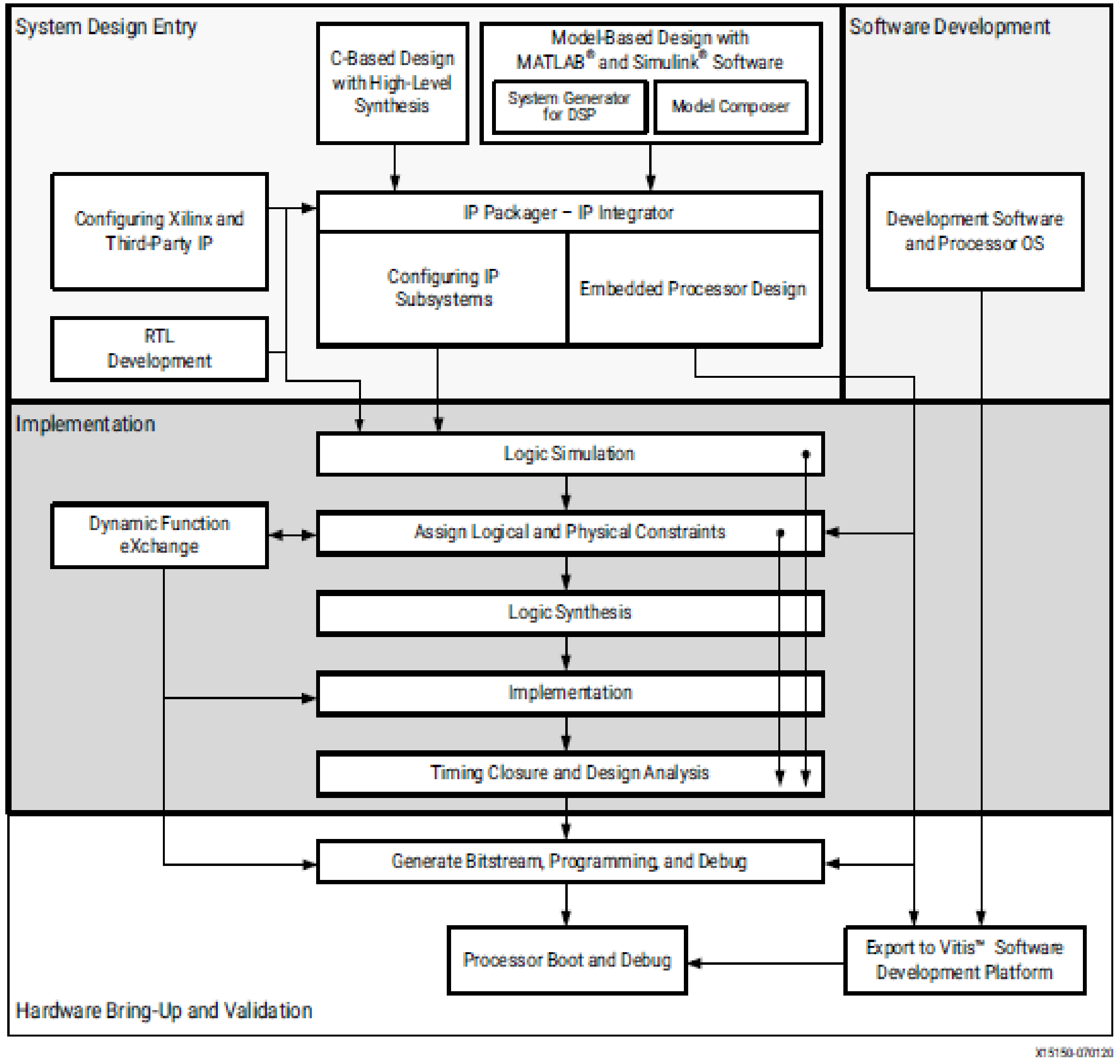
HDL Module



Synthesis results presented as a schematic of the resultant logic circuit

FPGA Development Environment

- Vivado – Xilinx Corp. (now part of AMD)
- Perform all steps of FPGA code development
  - Source code
  - Simulation
  - Synthesis
  - Implementation
  - Bitstream generation
  - Program Hardware (Pynq Board) via USB port
- Full version available on COE VLAB server
  - Vivado 2022
- Free version from Xilinx
  - Limited capability
  - Good enough for this course
  - Run Vivado on your own PC

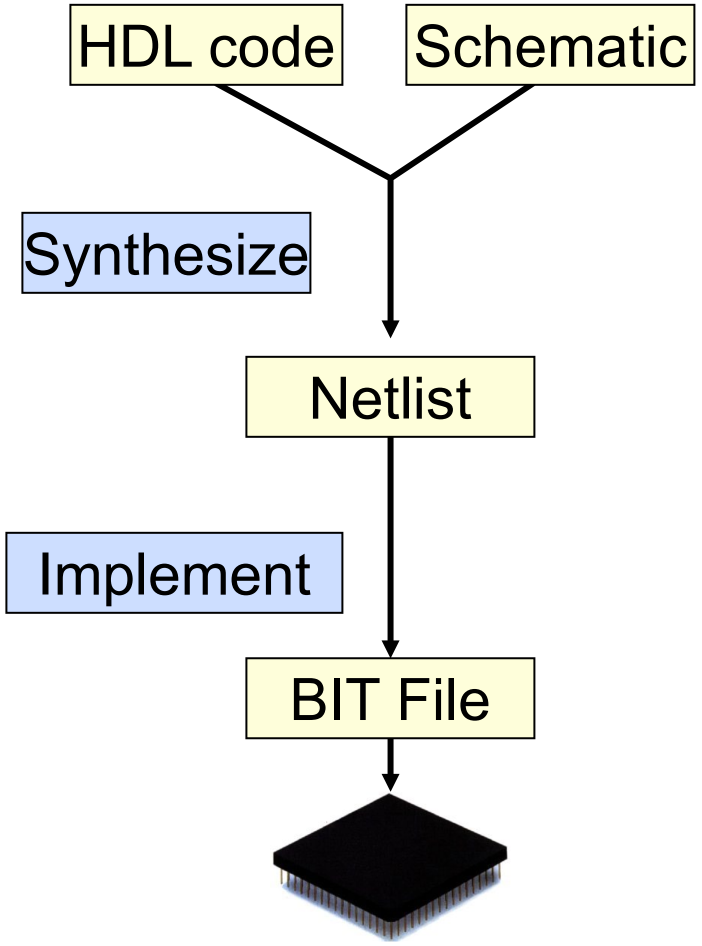


System Design Entry
C-Based Design with High-Level Synthesis
Model-Based Design with MATLAB® and Simulink® Software
System Generator for DSP
Model Composer
Software Development
Configuring Xilinx and Third-Party IP
IP Packager – IP Integrator
Configuring IP Subsystems
Embedded Processor Design
Development Software and Processor OS
RTL Development
Implementation
Logic Simulation
Dynamic Function eXchange
Assign Logical and Physical Constraints
Logic Synthesis
Implementation
Timing Closure and Design Analysis
Generate Bitstream, Programming, and Debug
Processor Boot and Debug
Export to Vitis™ Software Development Platform
Hardware Bring-Up and Validation
X15150-070120

Figure 1-1: **Vivado Design Suite High-Level Design Flow**

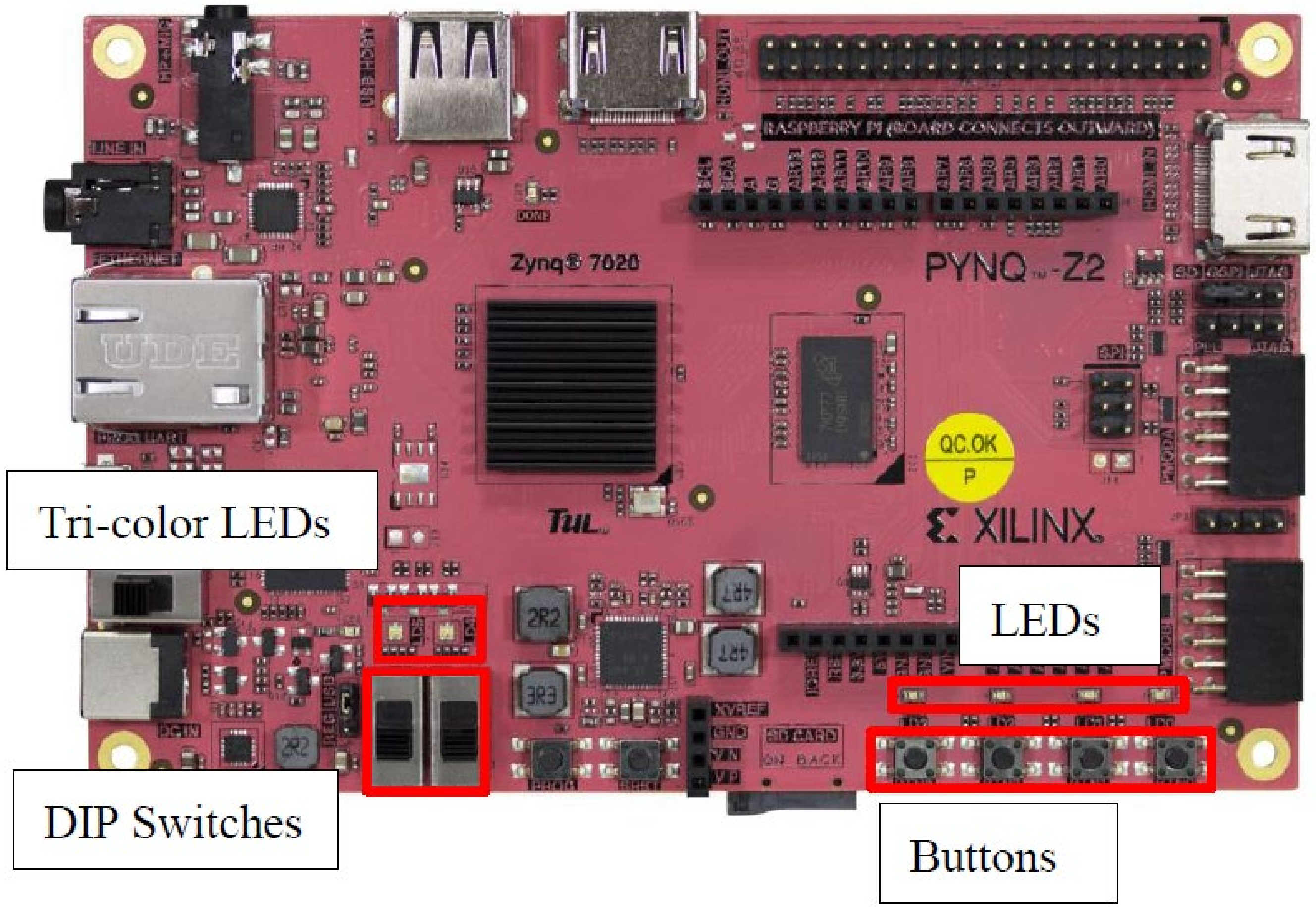
FPGA Design Process

- **Step1:** Create the Design
  - Two design entry methods: HDL(Verilog or VHDL) or schematic drawings
  - Check for syntax errors
- **Step 2:** Simulate the Design
  - Check for behavioral errors
- **Step 3:** Synthesize to create Netlist
  - Translates V, VHD, SCH files into an industry standard format EDIF file
- **Step 4:** Implement design (netlist)
  - Translate, Map, Place & Route Interconnects on Chip
- **Step 5:** Configure FPGA
  - Download BIT file into FPGA
- **Step 6:** Test the design in hardware
  - FPGA & associated circuits



- Netlist: “a list of the electronic components in a circuit and a list of the nodes they are connected to.” (Wikipedia)
- For FPGA’s the components are: Logic elements, Input/output elements, special function elements (e.g. RAM, multiplier)
- EDIF: Electronic Design Interchange Format, standard format for netlists.

Mapping Code I/O to Hardware: The Constraint File

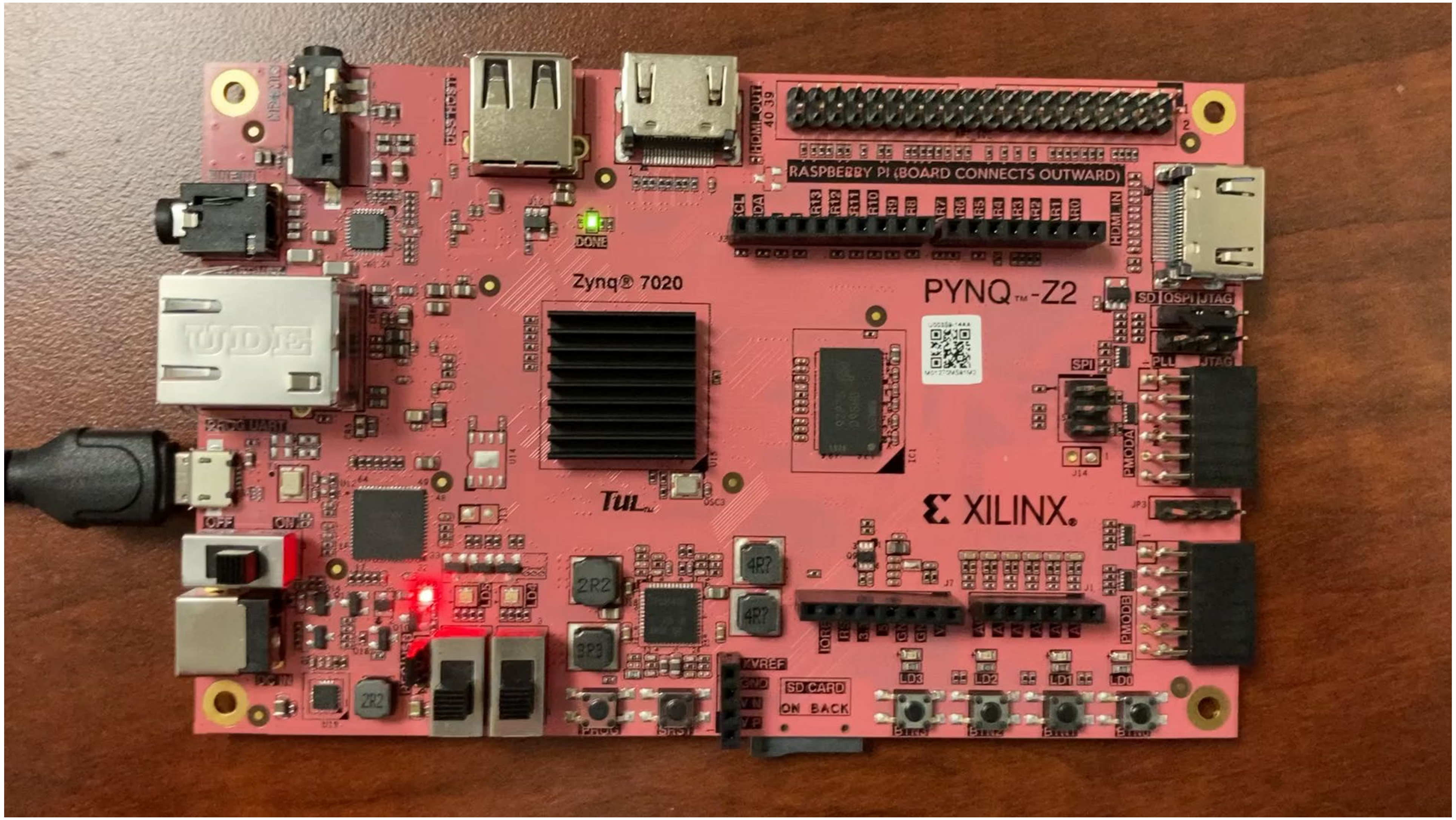


```
set_property -dict {PACKAGE_PIN M20 IOSTANDARD LVCMOS33} [get_ports a]
set_property -dict {PACKAGE_PIN M19 IOSTANDARD LVCMOS33} [get_ports b]
set_property -dict {PACKAGE_PIN R14 IOSTANDARD LVCMOS33} [get_ports c]
```

Signal	PL Pin
Pushbuttons	
BTN0	D19
BTN1	D20
BTN2	L20
BTN3	L19
Switches	
SW0	M20
SW1	M19
LEDs	
LED0	R14
LED1	P14
LED2	N16
LED3	M14
Tri-color LEDs	
LD4 Blue	L15
LD4 Red	N15
LD4 Green	G17
LD5 Blue	G14
LD5 Red	M15
LD5 Green	L14

Tul Pync-Z2 User Manual

XOR GATE


Zynq® 7020
PYNQ™-Z2
XILINX
RASPBERRY PI (BOARD CONNECTS OUTWARD)
DONE
SD CARD
ON BACK
LD3
LD2
LD1
LD0
PMODA
PMODB
HDMI_IN
HDMI_OUT

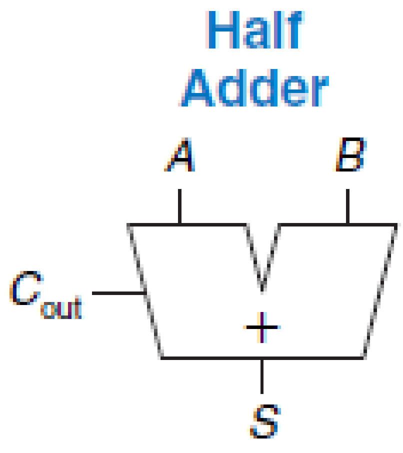
Lab 0: Learn Vivado by Building a Full Adder

Start with a Half Adder

- Adds two single bit numbers
- Two inputs: A, B
- Two Outputs:
  - Sum, S
  - Carry bit, C_{out}

$$\begin{array}{r} 0 \\ +\underline{0} \\ 0 \end{array} \quad \begin{array}{r} 1 \\ +\underline{0} \\ 1 \end{array} \quad \begin{array}{r} 1 \\ +\underline{1} \\ 10 \end{array} \quad \begin{array}{r} 1 \\ 1 \\ +\underline{1} \\ 11 \end{array}$$

Carry Carry

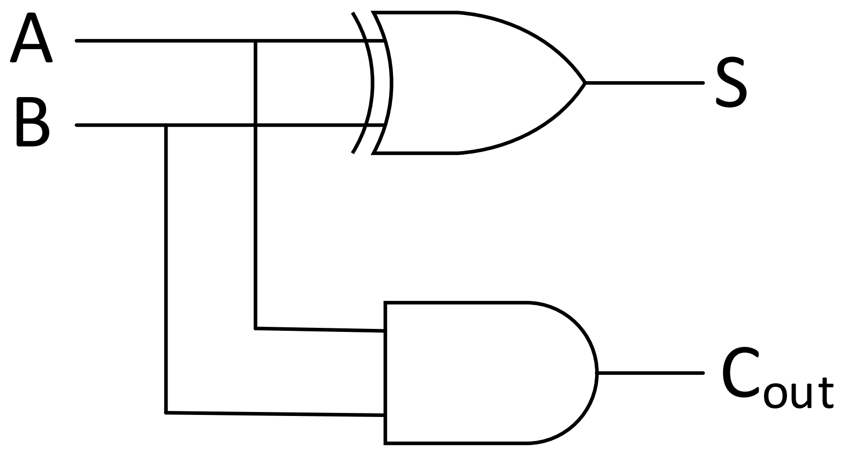


Truth Table

A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = A \oplus B$

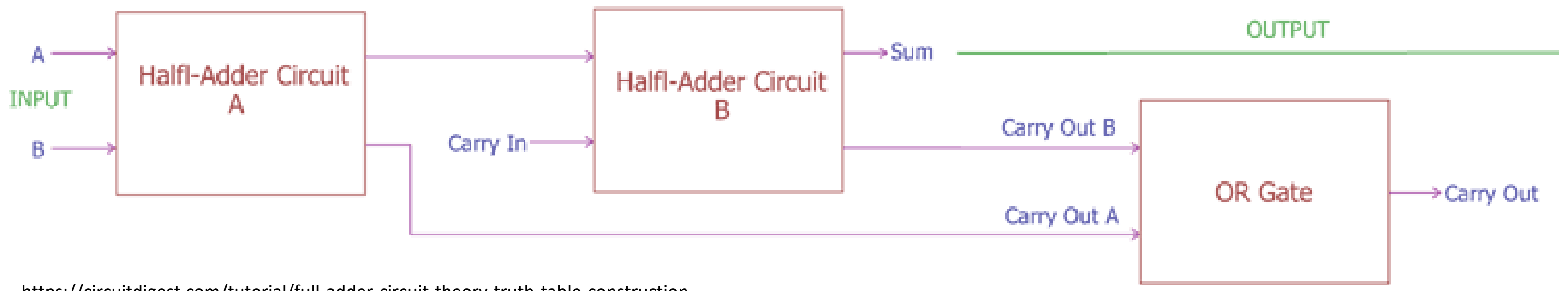
$C_{out} = AB$



Half-Adder Implementation

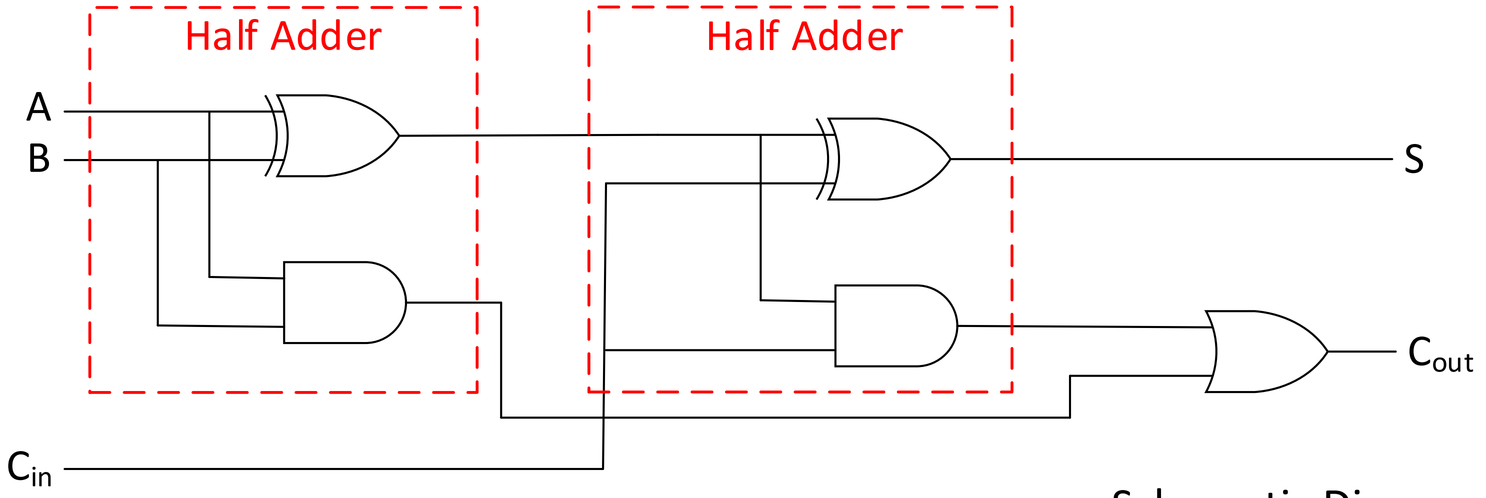
Full Adder Circuit Implementation

C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



https://circuitdigest.com/tutorial/full-adder-circuit-theory-truth-table-construction

Block Diagram



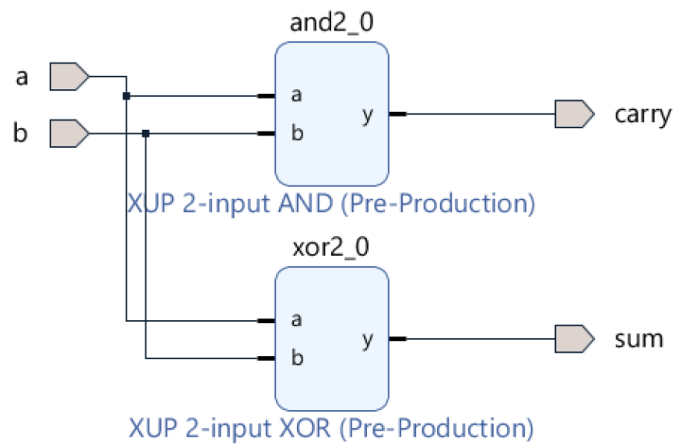
Schematic Diagram

Boolean Equations

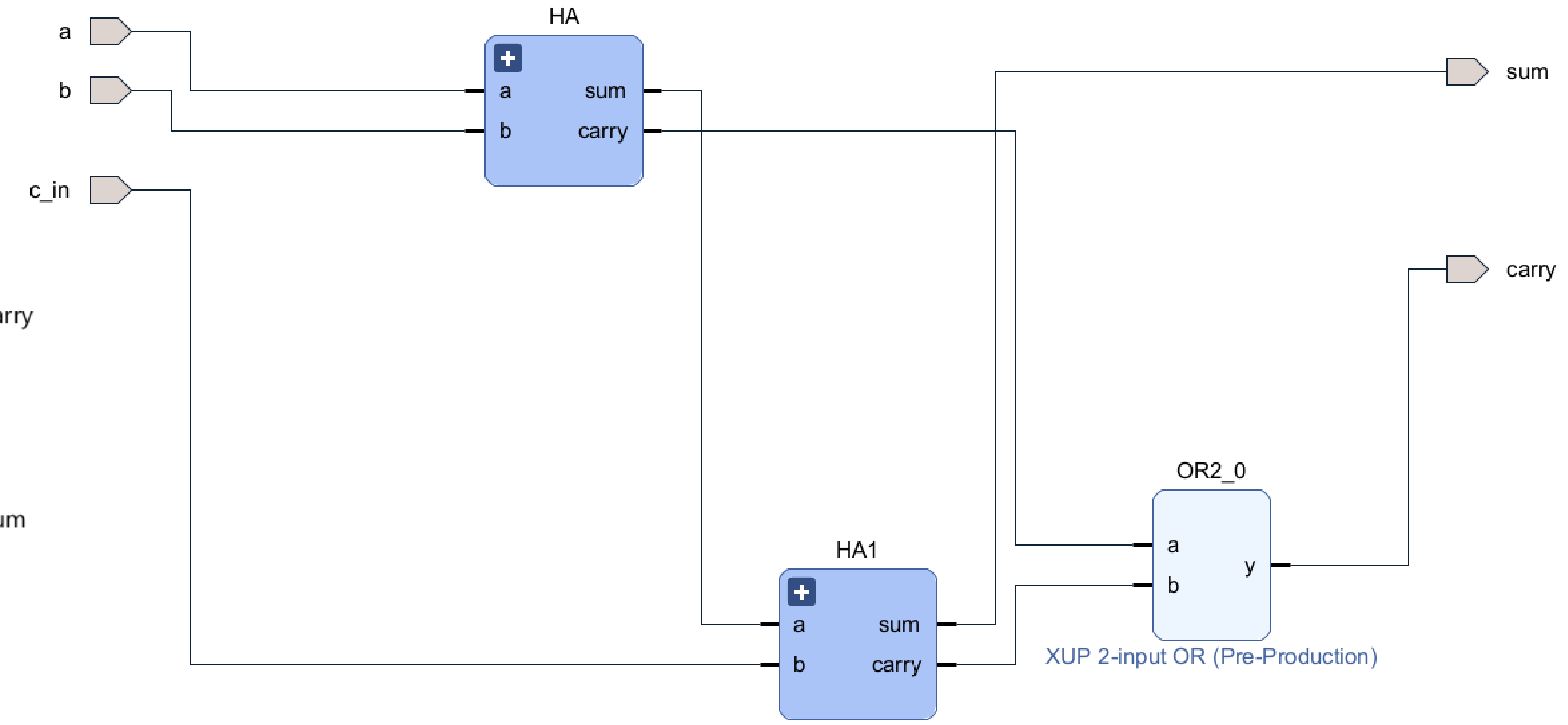
$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + C_{in}(A \oplus B)$$

Use Schematic Entry Method (not HDL)

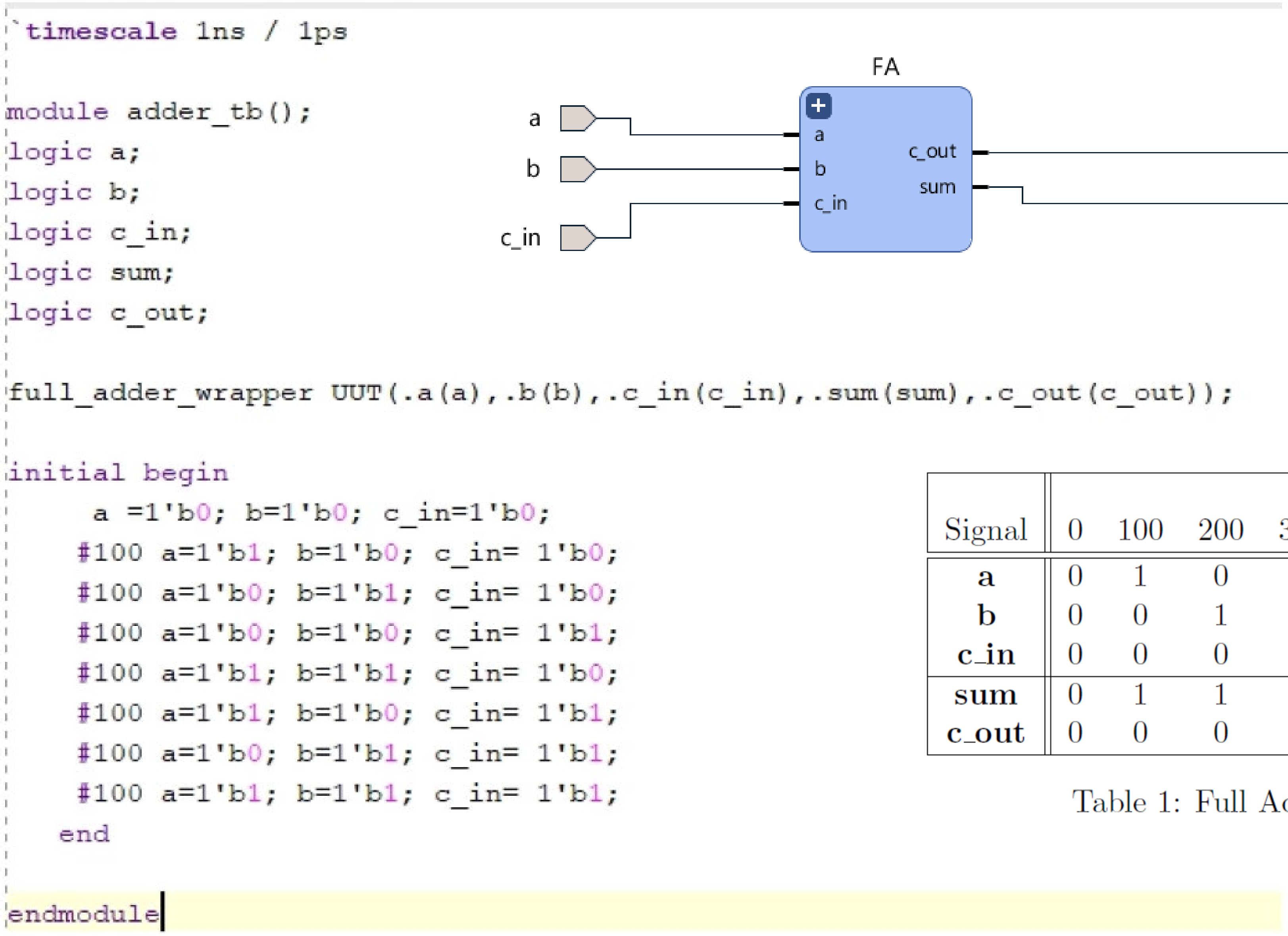

a
b
and2_0
a
b
y
carry
XUP 2-input AND (Pre-Production)
xor2_0
a
b
y
sum
XUP 2-input XOR (Pre-Production)

Half Adder

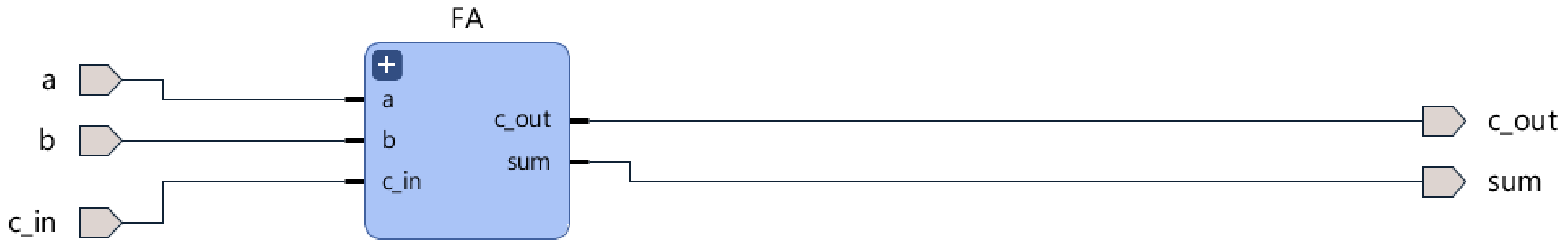


Full Adder

Simulate Circuit: Test Bench (System Verilog) and Test Vectors



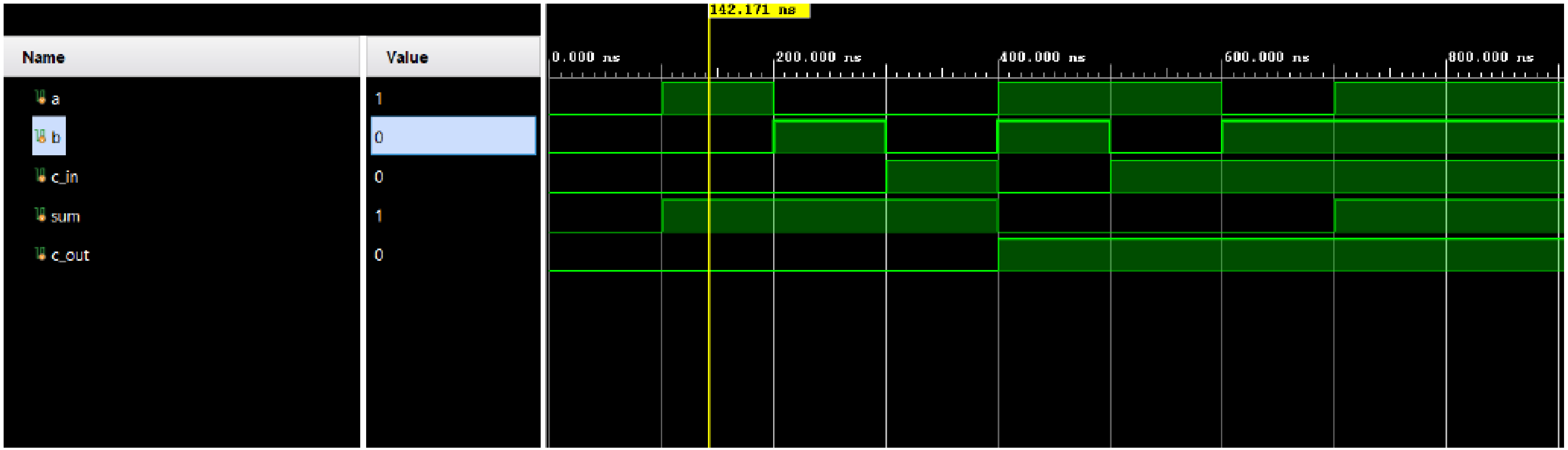
```
`timescale 1ns / 1ps

module adder_tb();
logic a;
logic b;
logic c_in;
logic sum;
logic c_out;

full_adder_wrapper UUT(.a(a),.b(b),.c_in(c_in),.sum(sum),.c_out(c_out));

initial begin
     a =1'b0; b=1'b0; c_in=1'b0;
    #100 a=1'b1; b=1'b0; c_in= 1'b0;
    #100 a=1'b0; b=1'b1; c_in= 1'b0;
    #100 a=1'b0; b=1'b0; c_in= 1'b1;
    #100 a=1'b1; b=1'b1; c_in= 1'b0;
    #100 a=1'b1; b=1'b0; c_in= 1'b1;
    #100 a=1'b0; b=1'b1; c_in= 1'b1;
    #100 a=1'b1; b=1'b1; c_in= 1'b1;
   end

endmodule
```

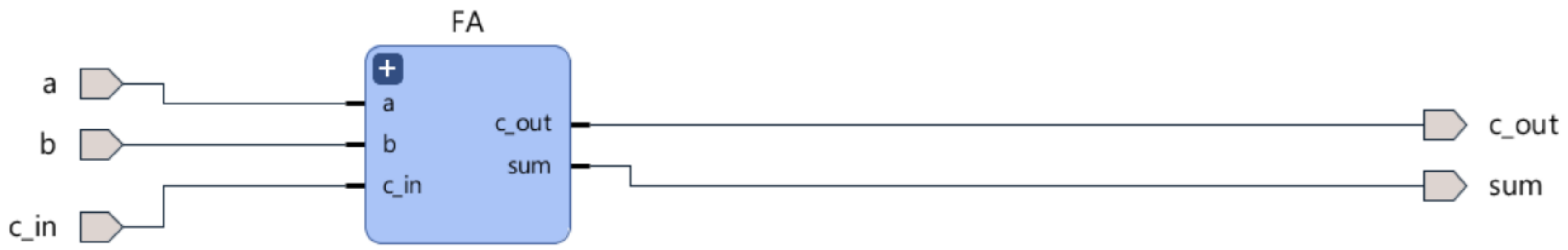


	Time (ns)							
Signal	0	100	200	300	400	500	600	700
a	0	1	0	0	1	1	0	1
b	0	0	1	0	1	0	1	1
c_in	0	0	0	1	0	1	1	1
sum	0	1	1	1	0	0	0	1
c_out	0	0	0	0	1	1	1	1

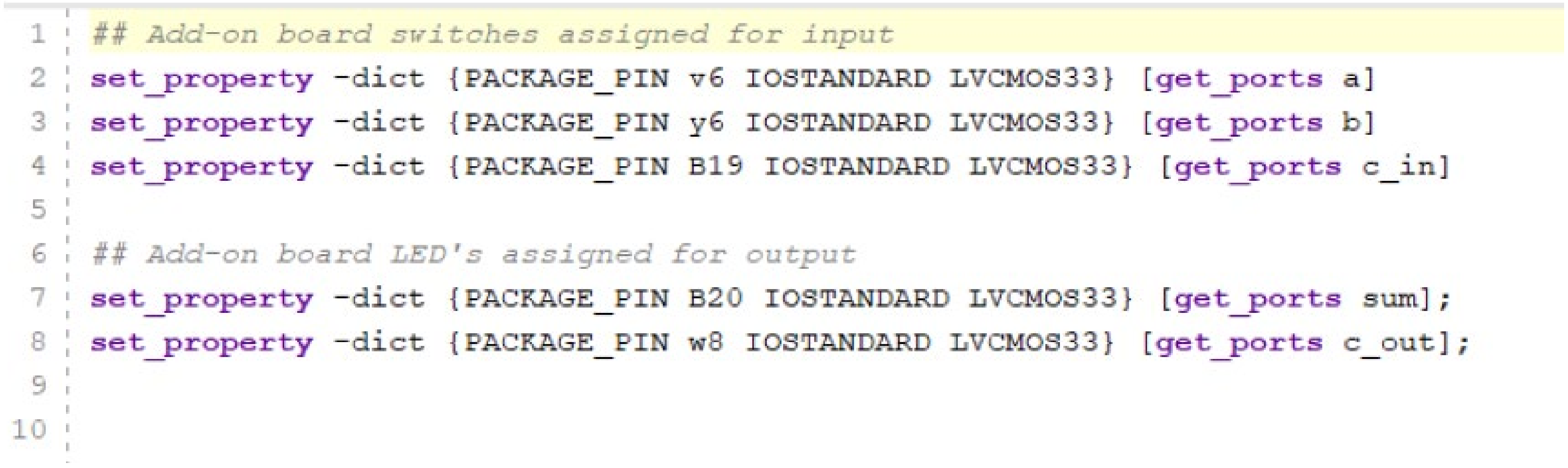
Table 1: Full Adder Test Vectors

Simulation Output

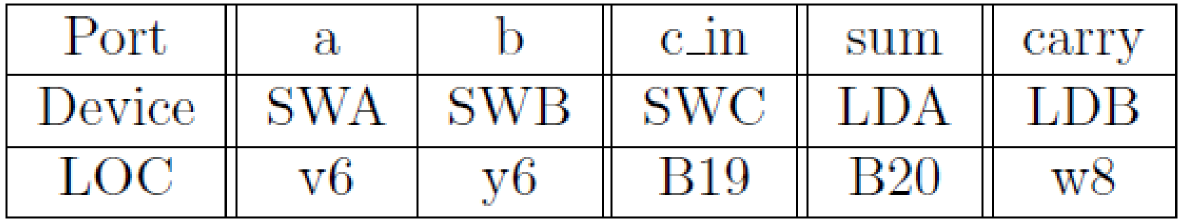




Implement Circuit in FPGA on Pynq Board
Constraint File



```
## Add-on board switches assigned for input
set_property -dict {PACKAGE_PIN v6 IOSTANDARD LVCMOS33} [get_ports a]
set_property -dict {PACKAGE_PIN y6 IOSTANDARD LVCMOS33} [get_ports b]
set_property -dict {PACKAGE_PIN B19 IOSTANDARD LVCMOS33} [get_ports c_in]

## Add-on board LED's assigned for output
set_property -dict {PACKAGE_PIN B20 IOSTANDARD LVCMOS33} [get_ports sum];
set_property -dict {PACKAGE_PIN w8 IOSTANDARD LVCMOS33} [get_ports c_out];
```

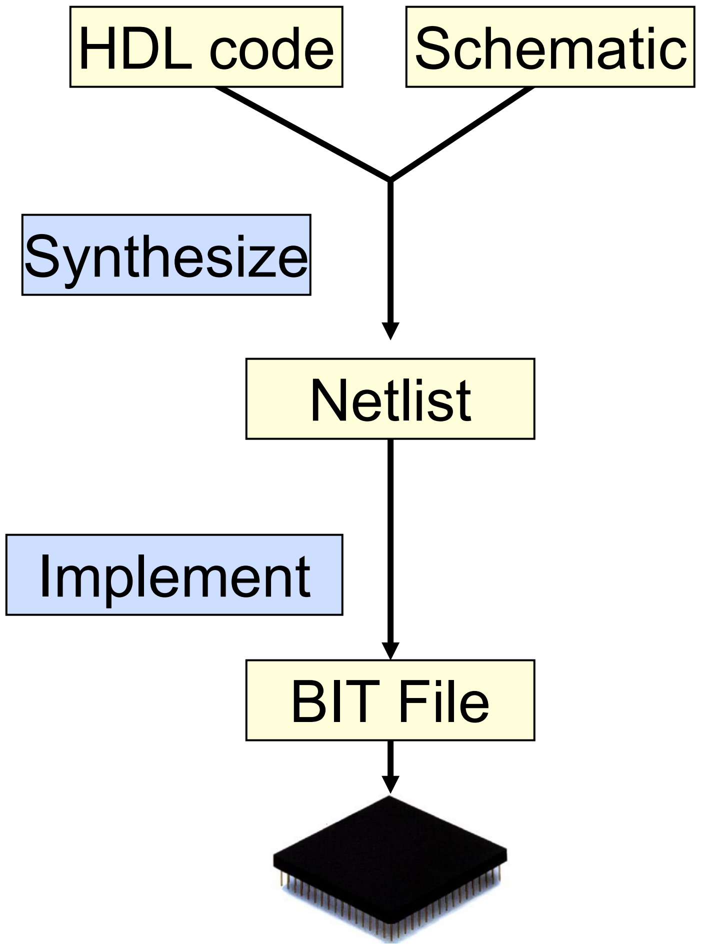


Port	a	b	c_in	sum	carry
Device	SWA	SWB	SWC	LDA	LDB
LOC	v6	y6	B19	B20	w8

Table 2: LOC Attributes for Full Adder I/Os

FPGA Design Process

- **Step1:** Create the Design
  - Two design entry methods: HDL(Verilog or VHDL) or schematic drawings
  - Check for syntax errors
- **Step 2:** Simulate the Design
  - Check for behavioral errors
- **Step 3:** Synthesize to create Netlist
  - Translates V, VHD, SCH files into an industry standard format EDIF file
- **Step 4:** Implement design (netlist)
  - Translate, Map, Place & Route Interconnects on Chip
- **Step 5:** Configure FPGA
  - Download BIT file into FPGA
- **Step 6:** Test the design in hardware
  - FPGA & associated circuits



- Netlist: “a list of the electronic components in a circuit and a list of the nodes they are connected to.” (Wikipedia)
- For FPGA’s the components are: Logic elements, Input/output elements, pecial function elements (e.g. RAM, multiplier)
- EDIF: Electronic Design Interchange Format, standard format for netlists.