VHDL Short Course – Module 1
Introduction

Jim Duckworth

ECE Department, WPI

Topics

- Background to VHDL
- Introduction to language
- Programmable Logic Devices
  - CPLDs and FPGAs
  - FPGA architecture
  - Spartan 3 Starter Board
- Using VHDL to synthesize and implement a design
- Verilog overview

Hardware Description Languages

- Example HDL's : ABEL, VERILOG, VHDL
- Advantages:
  - Documentation
  - Flexibility (easier to make design changes or mods)
  - Portability (if HDL is standard)
  - One language for modeling, simulation (test benches), and synthesis
  - Let synthesis worry about gate generation
    - Engineer productivity
- However: A different way of approaching design
  - engineers are used to thinking and designing using graphics (schematics) instead of text.

VHDL

- **V**HSIC **H**ardware **D**escription **L**anguage
  - **V**ery **H**igh **S**peed **I**ntegrated **C**ircuit
- Standard language used to describe digital hardware devices, systems and components
  - Developed initially for documentation
- VHDL program was an offshoot of the US Government's VHSIC Program
- Approved as an IEEE Standard in December 1987 (IEEE standard 1076-1987)
  - Revised - now 1076-1993 (supported by all tools)
  - Work under way on VHDL-200X
    - Integration of 1164 std
    - General improvements, etc

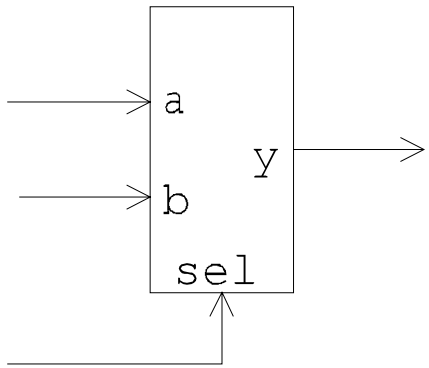
VHDL References

- IEEE Standard VHDL Language Reference Manual (1076 - 1993)
- "Introductory VHDL *From Simulation to Synthesis* by Sudhakar Yalamanchilli, 2002, Xilinx Design Series, Prentice Hall
- "VHDL Modeling for Digital Design Synthesis" by Hsu, Tsai, Liu, and Lin, 1995, Kluwer Academic Press
- "VHDL Made Easy" by David Pellerin and Douglas Taylor, 1997, Prentice Hall
- "VHDL for Programmable Logic" by Kevin Skahill, 1996, Addison Wesley
- "Logic Synthesis using Synopsys", second edition, by Kurup and Abbasi, 1997, Kluwer Academic Press
- "A VHDL Primer" by J. Bkasker, Third Edition, 1999, Prentice Hall
- "Digital Systems Design Using VHDL" by Charles Roth, 1998, PWS Publishing

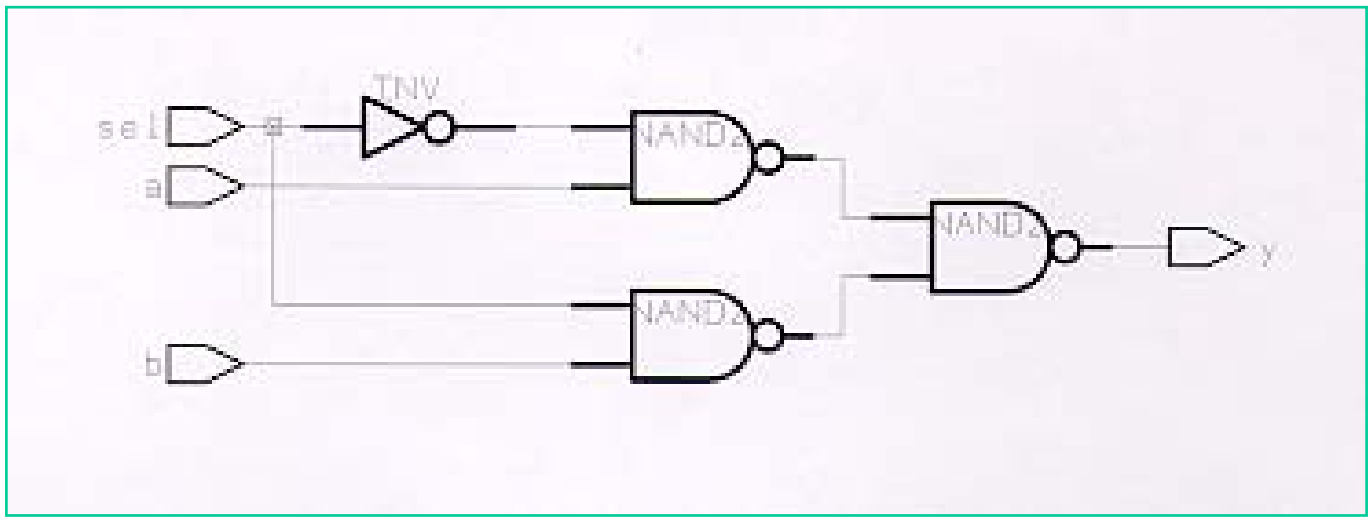
What exactly is VHDL ?

- A way of ***describing*** the operation of a system.
  - Example: a 2-input multiplexer

```
ENTITY mux IS
  PORT (a, b, sel :  IN std_logic;
        y :  OUT std_logic);
END mux;

ARCHITECTURE behavior OF mux IS
BEGIN

  y <= a WHEN sel = '0' ELSE
       b;

END behavior;
```



Example Synthesis Results (not Xilinx)



Basic Terminology

- Note: VHDL is case insensitive, free format.
- Comments are preceded by two consecutive dashes.
  - comment ends at end of current line
- A digital component is described using an
  - ENTITY DECLARATION and a corresponding
  - ARCHITECTURE BODY.
- Std_logic is an enumeration type defined in an IEEE package
  - has the values '0' and ‘1’ and ‘Z’ (and others)
- Ports are like IC pins, connected by wires called SIGNALS

IEEE STANDARD 1164

- Provides a standard data type (std_logic) - nine values
  - U unitialized
  - X forcing unknown
  - 0 forcing logic 0
  - 1 forcing logic 1
  - Z high impedance
  - W weak unknown
  - L weak logic 0
  - H weak logic 1
  - - don't care
- To use standard logic data types place at top of source file
  - LIBRARY ieee; -- library
  - USE ieee.std_logic_1164.ALL; -- package

Entity

- The ENTITY defines the external view of the component
- PORTS are the communication links between entities or connections to the device pins
- Note the use of libraries before entity description

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux IS
  PORT (a, b, sel :  IN std_logic;
        y :  OUT std_logic);
END mux;
```

Architecture

- The ARCHITECTURE defines the function or behavior or structure of the ENTITY
- Consists of concurrent statements, e.g.
  - Process statements
  - Concurrent Signal Assignment statements
  - Conditional Signal Assignment statements
- An entity may have several architectures

```
ARCHITECTURE behavior OF mux IS
BEGIN

  y <= a WHEN sel = '0' ELSE
       b;

END behavior;
```

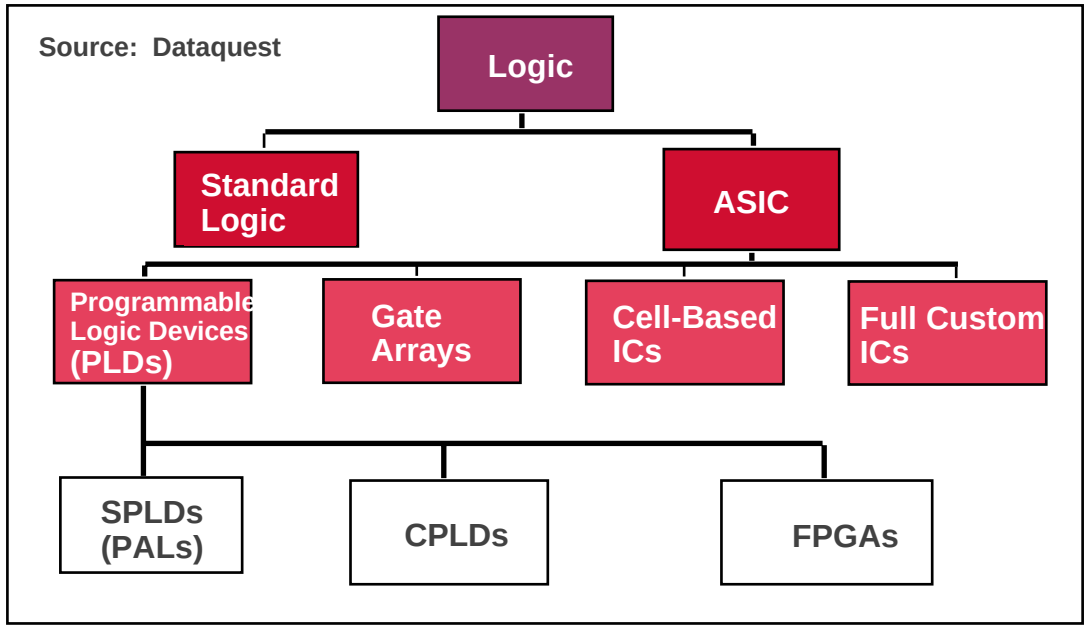
VHDL Notes

- There is no explicit reference to actual hardware components
  - There are no D-type flip-flops, mux, etc
  - Required logic is inferred from the VHDL description
  - Same VHDL can target many different devices
- There are many alternative ways to describe the required behavior of the final system
  - Exactly the same hardware will be produced
  - Some ways are more intuitive and easier to read
- Remember that the synthesis tools must be able to deduce your intent and system requirements
  - For sequential circuits it is usually necessary to follow recommended templates and style

Programmable Logic Devices

- Xilinx user programmable devices
  - FPGAs – Field Programmable Gate Array
    - Virtex 4, Virtex-II, Virtex-II PRO
    - Spartan 3
    - Consist of configurable logic blocks
      - Provides look-up tables to implement logic
      - Storage devices to implement flip-flops and latches
  - CPLDs – Complex Programmable Logic Devices
    - CoolRunner-II CPLDS (1.8 and 3.3 volt devices)
    - XC9500 Series (3.3 and 5 volt devices)
    - Consist of macrocells that contain programmable and-or matrix with flip-flops

Electronic Components (Xilinx)



Acronyms

SPLD = Simple Prog. Logic Device
PAL = Prog. Array of Logic
CPLD = Complex PLD
FPGA = Field Prog. Gate Array

Common Resources

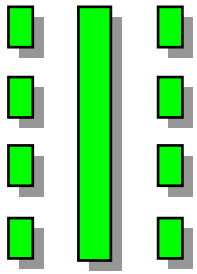
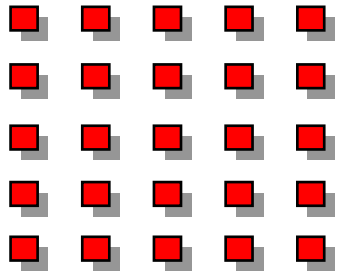
Configurable Logic Blocks (CLB)
- Memory Look-Up Table
- AND-OR planes
- Simple gates

Input / Output Blocks (IOB)
- Bidirectional, latches, inverters, pullup/pulldowns

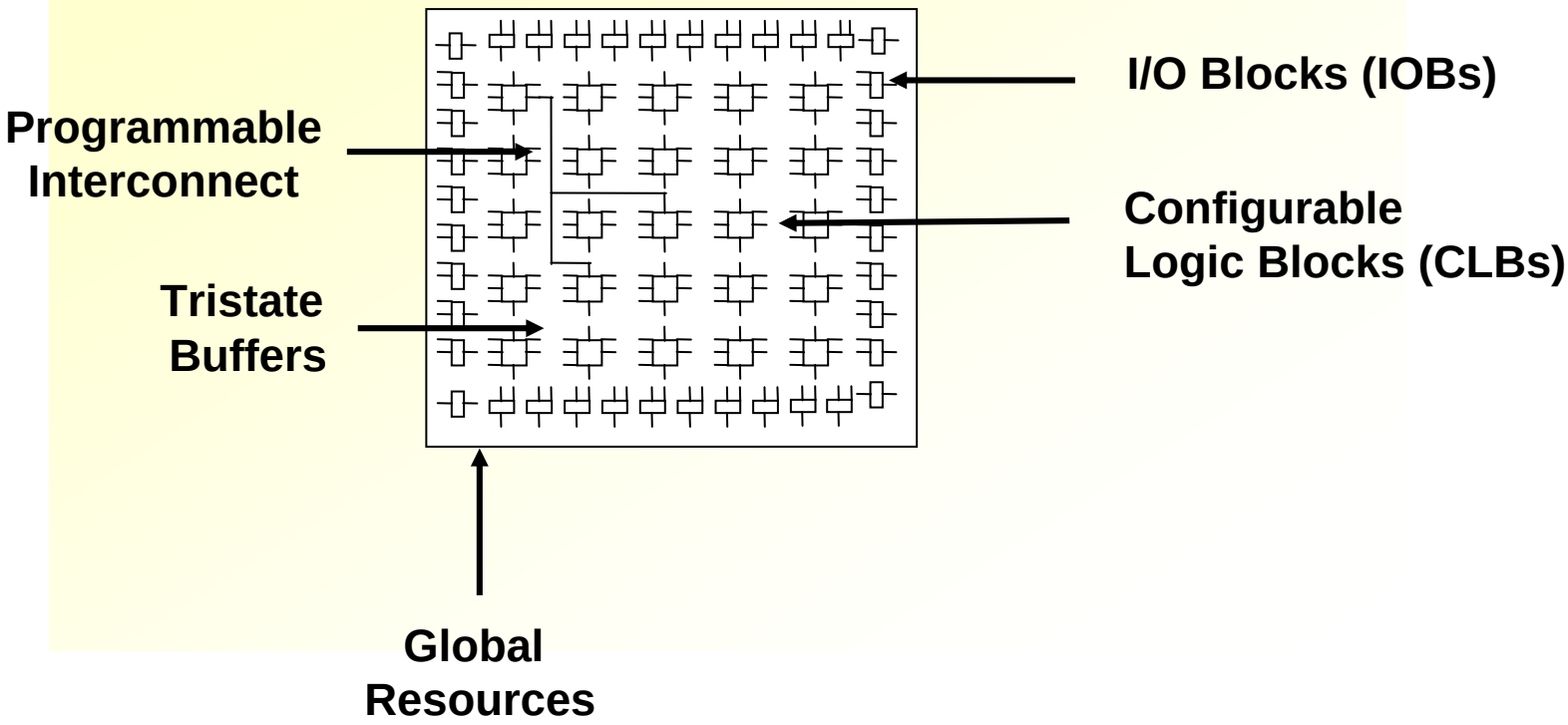
Interconnect or Routing
- Local, internal feedback, and global

Xilinx Products (Xilinx)

CPLDs and FPGAs

	Complex Programmable Logic Device (CPLD)	Field-Programmable Gate Array (FPGA)
		
Architecture	PAL/22V10-like More Combinational	Gate array-like More Registers + RAM
Density	Low-to-medium 0.5-10K logic gates	Medium-to-high 1K to 3.2M system gates
Performance	Predictable timing Up to 250 MHz today	Application dependent Up to 200 MHz today
Interconnect	“Crossbar Switch”	Incremental

Overview of Xilinx FPGA Architecture (Xilinx)



Spartan-3 FPGA Family

- "Designed to meet the needs of high-volume, cost-sensitive consumer electronic applications"
- 326 MHz system clock rate
- Programmed by loading configuration data into static memory cells – place serial PROM on board

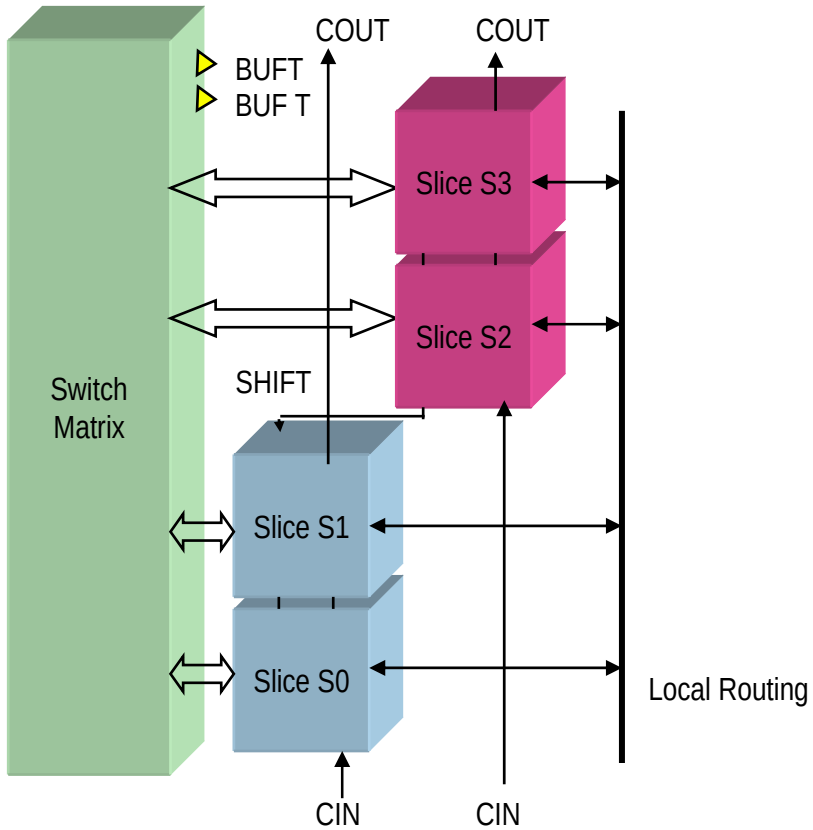
Device	System Gates	CLBs (4 slices)	CLB flip-flops	Block Ram (bits)	User IO	Price (250K)
XC3S200	200K	480	3,840	216K	173	$2.95
XC3S1000	1M	1,920	15,360	432K	391	$12
XC3S4000	4M	6,912	55,296	1,728K	712	$100

Programmable Functional Elements

- Configurable Logic Blocks (CLBs)
  - RAM-based look-up tables to implement logic
  - Storage elements for flip-flops or latches
- Input/Output Blocks
  - Supports bidirectional data flow and 3-state operation
  - Supports different signal standards including LVDS
  - Double-data rate registers included
  - Digitally controlled impedance provides on-chip terminations
- Block RAM provides data storage
  - 18-Kbit dual-port blocks
- Multiplier blocks (accepts two 18-bit binary numbers)
- Digital Clock Manager (DCM)
  - Provides distribution, delaying, mult, div, phase shift of clocks

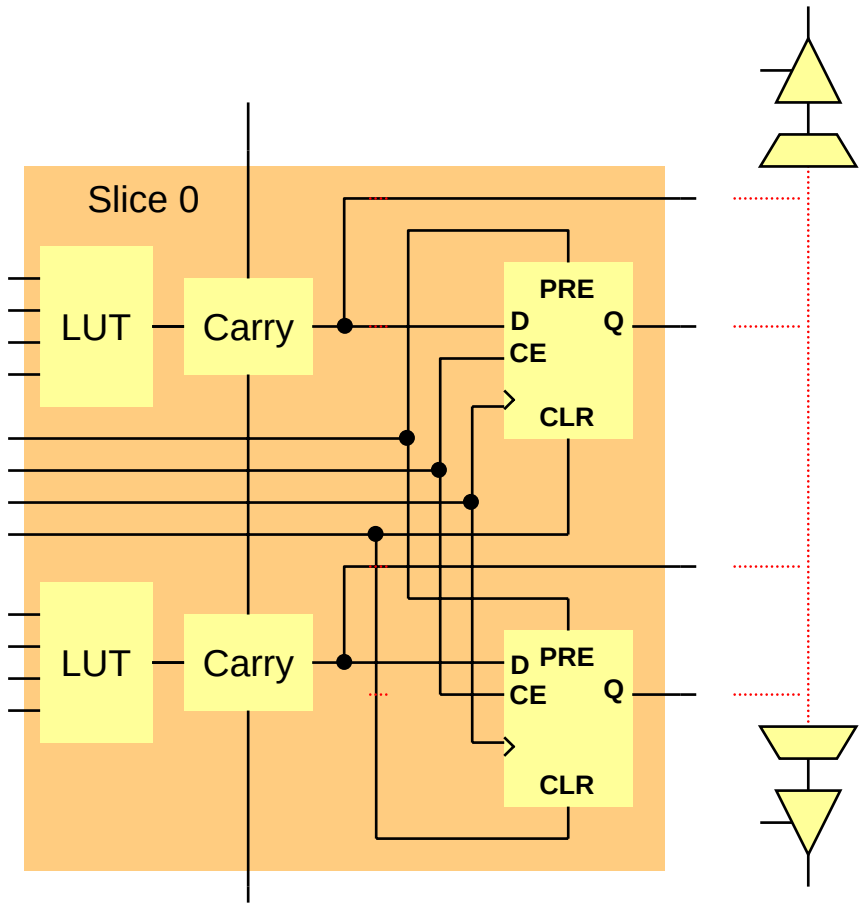
Slices and CLBs (Xilinx)

- Each Virtex™-II CLB contains four slices
  - Local routing provides feedback between slices in the same CLB, and it provides routing to neighboring CLBs
  - A switch matrix provides access to general routing resources



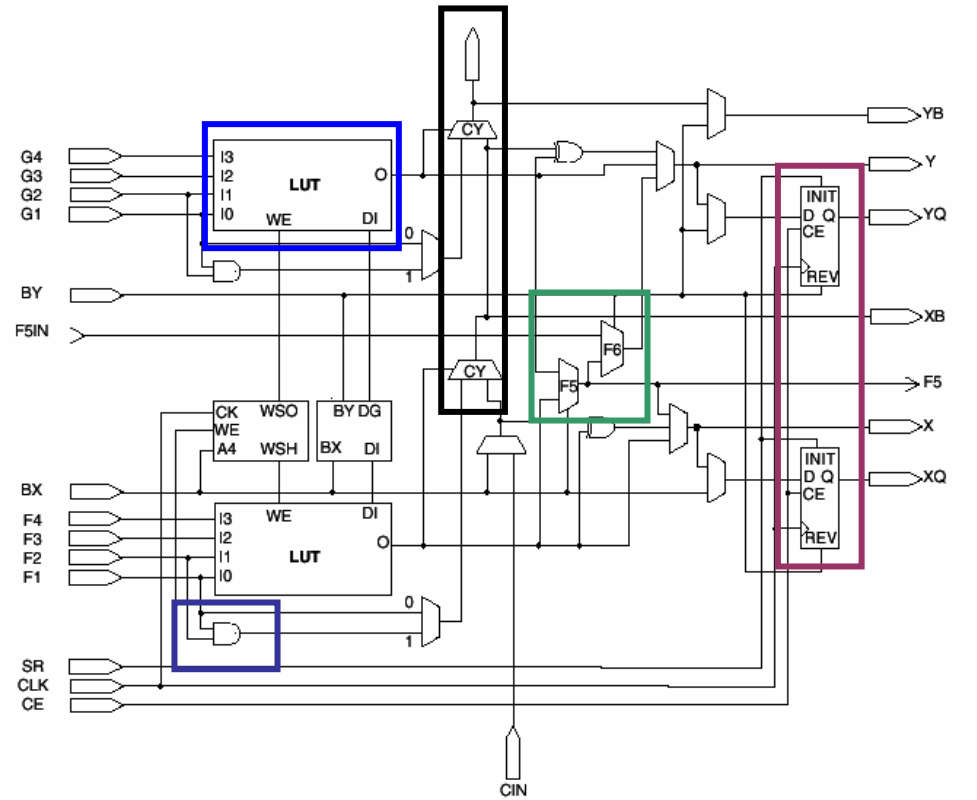
Simplified Slice Structure (Xilinx)

- Each slice has four outputs
  - Two registered outputs, two non-registered outputs
  - Two BUFTs associated with each CLB, accessible by all 16 CLB outputs
- Carry logic runs vertically, up only
  - Two independent carry chains per CLB



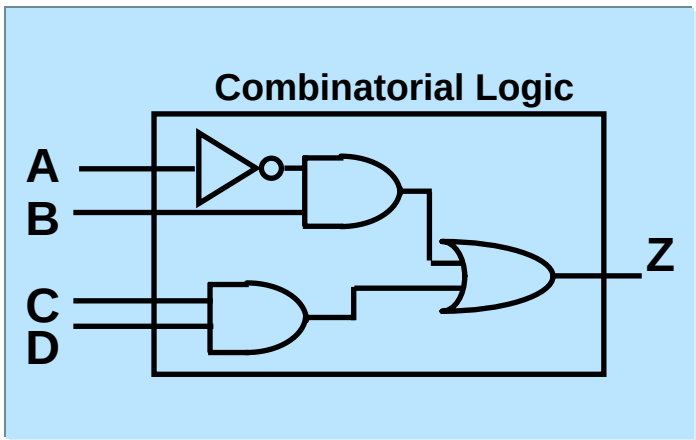
Detailed Slice Structure (Xilinx)

- The next slides will discuss the slice features
  - LUTs
  - MUXF5, MUXF6, MUXF7, MUXF8 (only the F5 and F6 MUX are shown in the diagram)
  - Carry Logic
  - MULT_ANDs
  - Sequential Elements



Look-Up Tables (Xilinx)

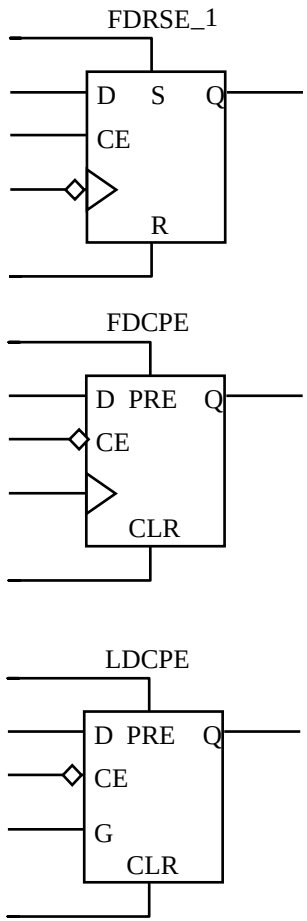
- Combinatorial logic is stored in Look-Up Tables (LUTs)
  - Also called Function Generators (FGs)
  - Capacity is limited by number of inputs, not complexity
- Delay through the LUT is constant



A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
	.	.	.	
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

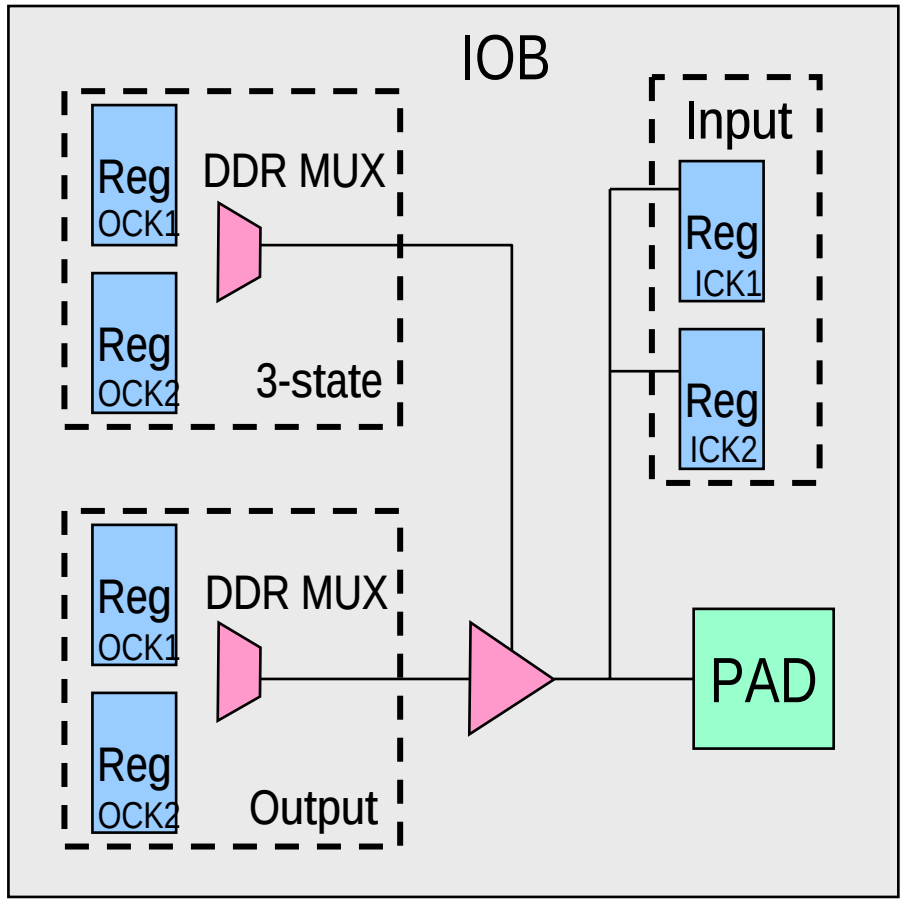
Flexible Sequential Elements (Xilinx)

- Can be flip-flops or latches
- Two in each slice; eight in each CLB
- Inputs can come from LUTs or from an independent CLB input
- Separate set and reset controls
  - Can be synchronous or asynchronous
- All controls are shared within a slice
  - Control signals can be inverted locally within a slice



IOB Element (Xilinx)

- Input path
  - Two DDR registers
- Output path
  - Two DDR registers
  - Two 3-state enable DDR registers
- Separate clocks and clock enables for I and O
- Set and reset signals are shared



SelectIO Standard (Xilinx)

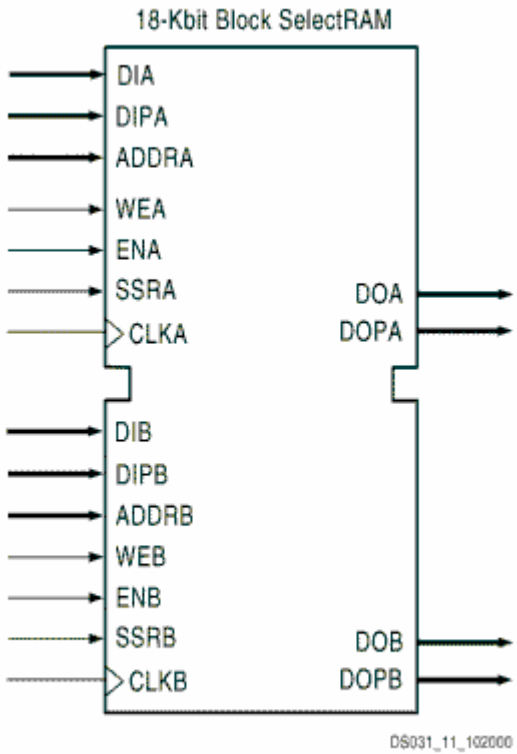
- Allows direct connections to external signals of varied voltages and thresholds
  - Optimizes the speed/noise tradeoff
  - Saves having to place interface components onto your board
- Differential signaling standards
  - LVDS, BLVDS, ULVDS
  - LDT
  - LVPECL
- Single-ended I/O standards
  - LVTTL, LVCMOS (3.3V, 2.5V, 1.8V, and 1.5V)
  - PCI-X at 133 MHz, PCI (3.3V at 33 MHz and 66 MHz)
  - GTL, GTLP
  - and more!

Digital Controlled Impedance (DCI)

- DCI provides
  - Output drivers that match the impedance of the traces
  - On-chip termination for receivers and transmitters
- DCI advantages
  - Improves signal integrity by eliminating stub reflections
  - Reduces board routing complexity and component count by eliminating external resistors
  - Internal feedback circuit eliminates the effects of temperature, voltage, and process variations

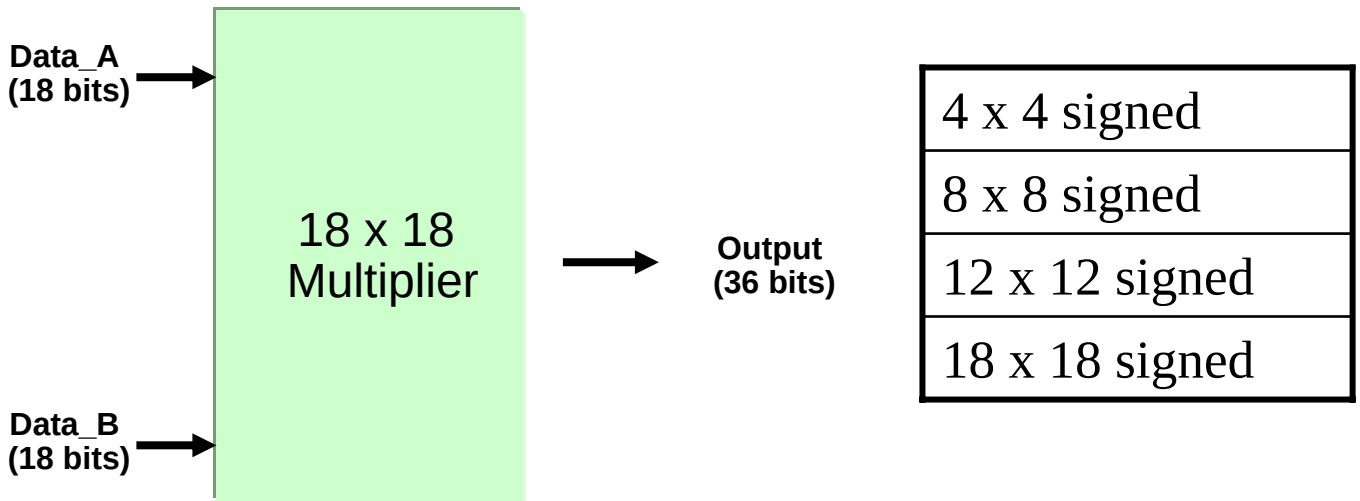
Block SelectRAM Resources (Xilinx)

- Up to 3.5 Mb of RAM in 18-kb blocks
  - Synchronous read and write
- True dual-port memory
  - Each port has synchronous read and write capability
  - Different clocks for each port
- Supports initial values
- Synchronous reset on output latches
- Supports parity bits
  - One parity bit per eight data bits

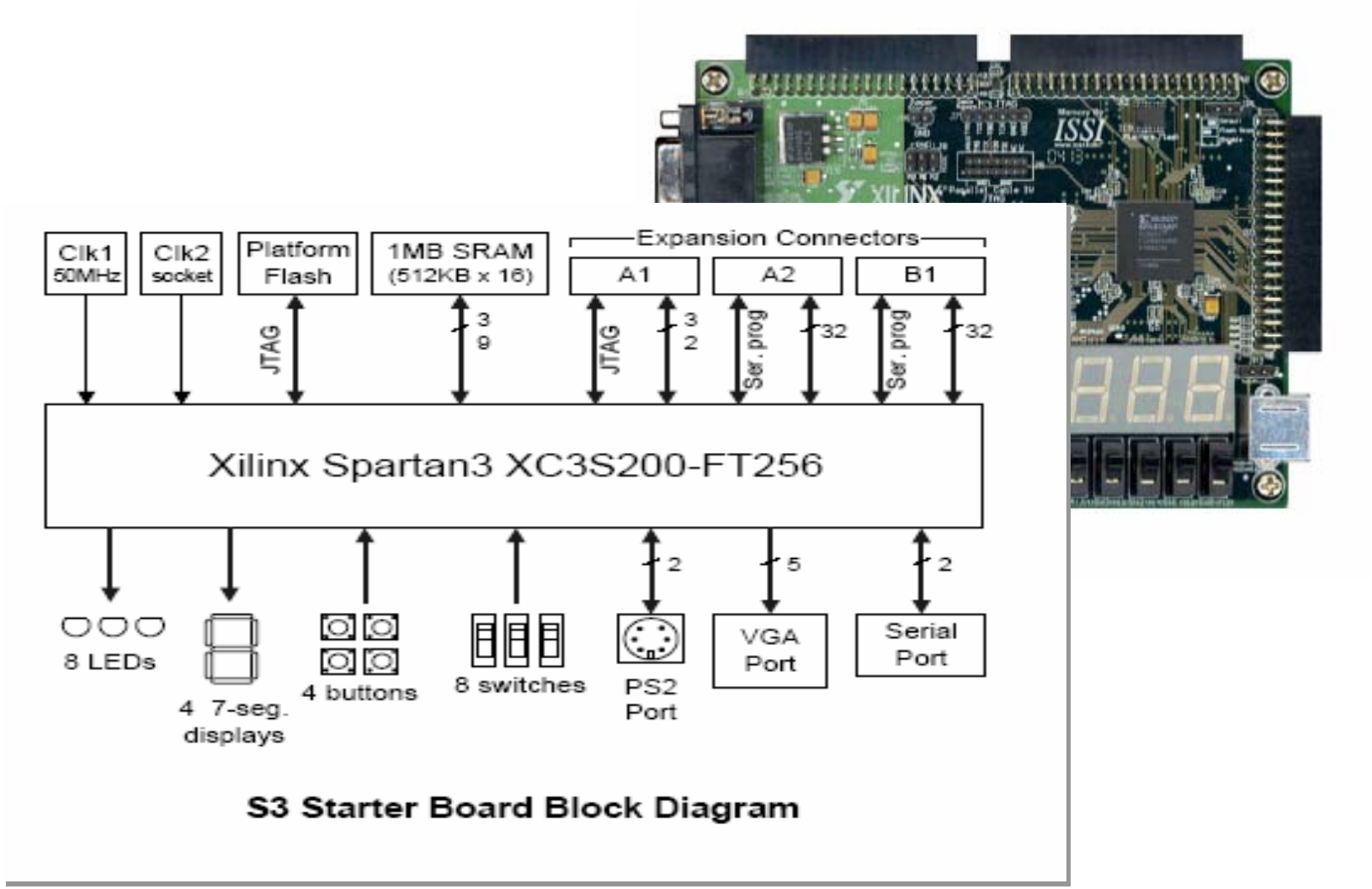


Dedicated Multiplier Blocks (Xilinx)

- 18-bit twos complement signed operation
- Optimized to implement multiply and accumulate functions
- Multipliers are physically located next to block SelectRAM™ memory

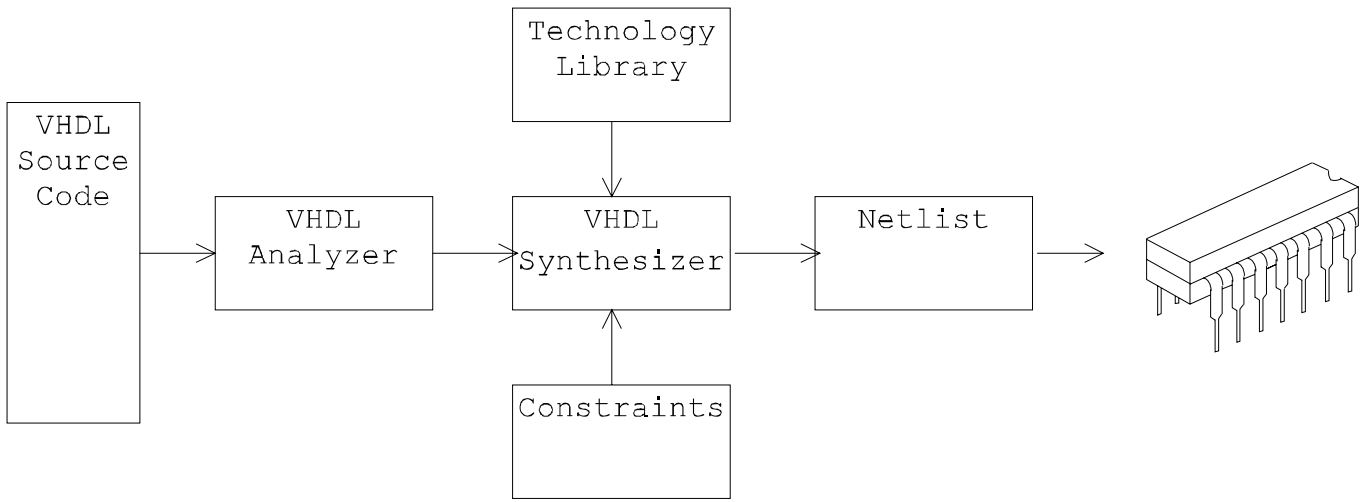


Spartan-3 Starter Board



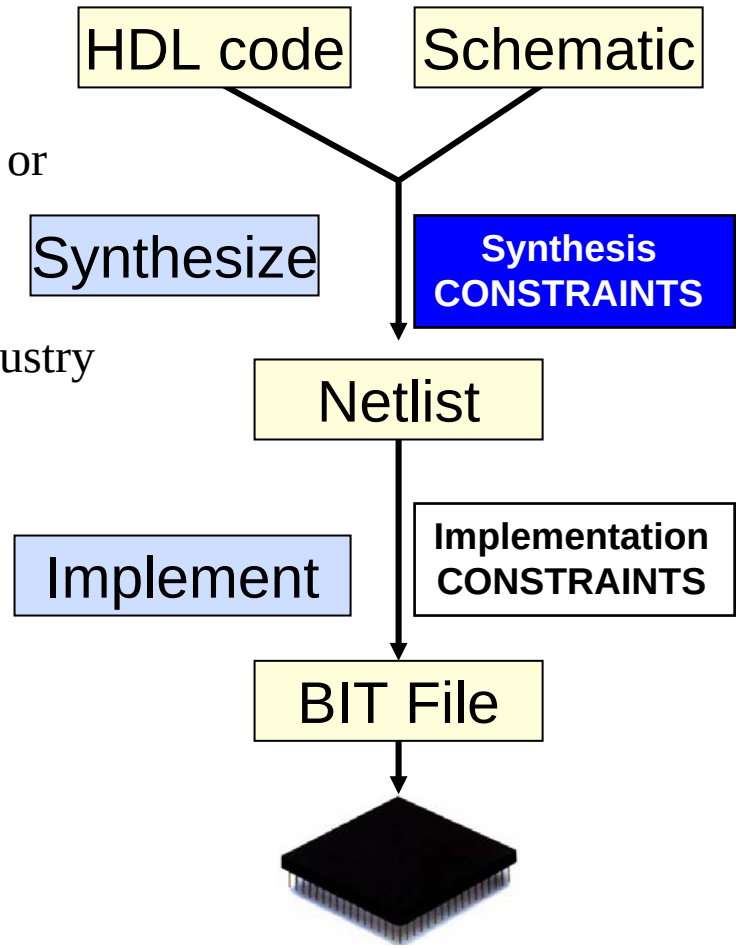
Logic Synthesis

- A process which takes a digital circuit description and translates it into a gate level design, optimized for a particular implementation technology.

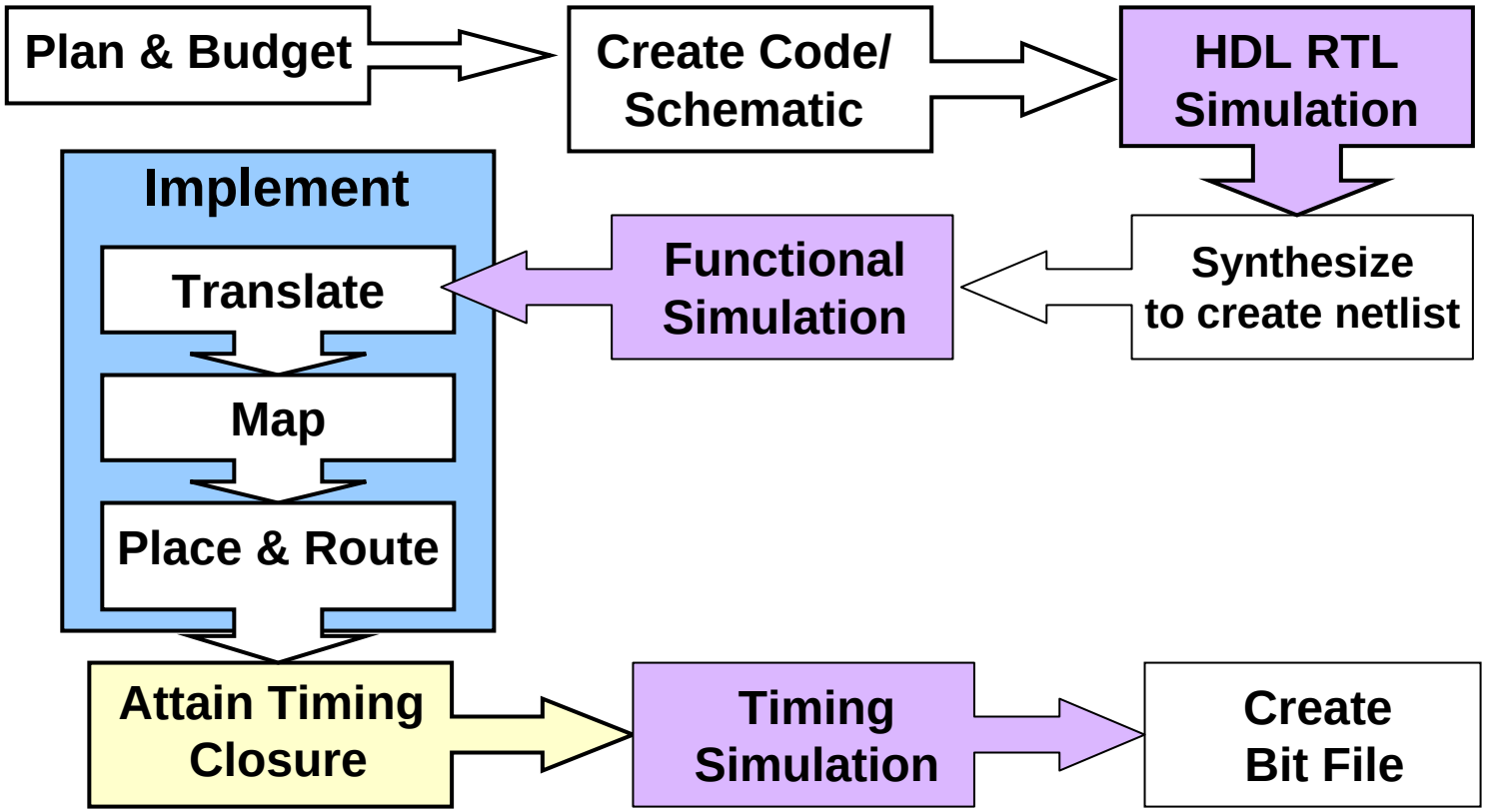


Xilinx Design Process (Xilinx)

- Step1: Design
  - Two design entry methods: HDL(Verilog or VHDL) or schematic drawings
- Step 2: Synthesize to create Netlist
  - Translates V, VHD, SCH files into an industry standard format EDIF file
- Step 3: Implement design (netlist)
  - Translate, Map, Place & Route
- Step 4: Configure FPGA
  - Download BIT file into FPGA

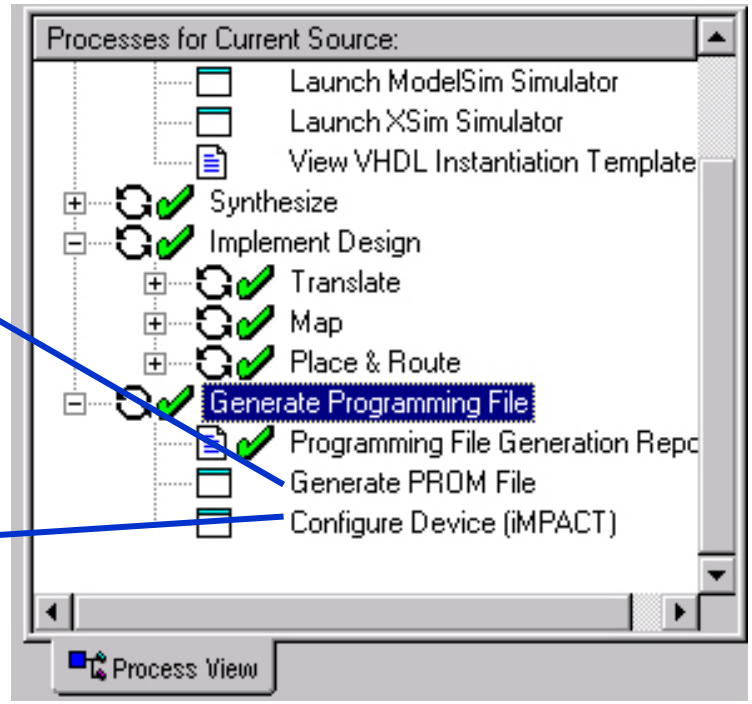


Xilinx Design Flow (Xilinx)

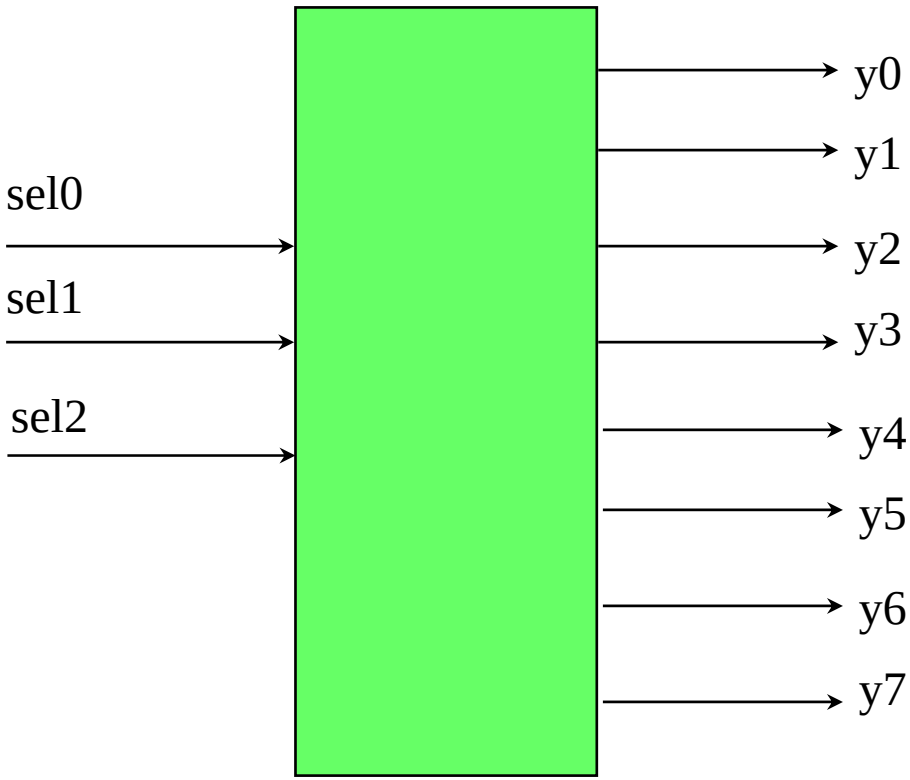


Program the FPGA (Xilinx)

- There are two ways to program an FPGA
  - Through a PROM device
    - You will need to generate a file that the PROM programmer will understand
  - Directly from the computer
    - Use the iMPACT configuration tool



Decoder Tutorial Demo Example



VHDL Source Code

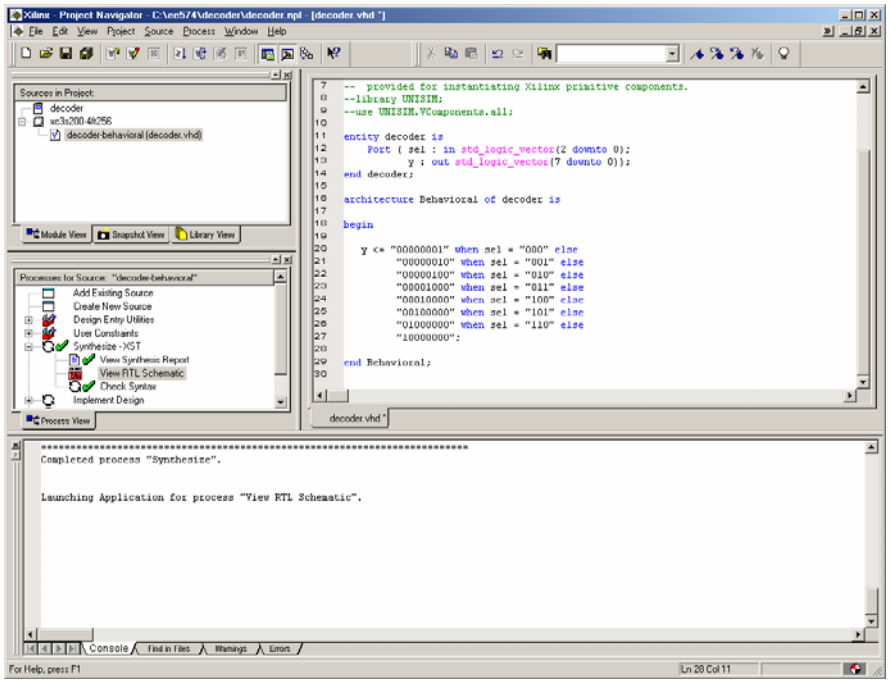
decoder.vhd * - ISE Text Editor

File Edit

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   --  Uncomment the following lines to use the declarations that are
7   --  provided for instantiating Xilinx primitive components.
8   --library UNISIM;
9   --use UNISIM.VComponents.all;
10
11  entity decoder is
12      Port ( sel : in std_logic_vector(2 downto 0);
13             y : out std_logic_vector(7 downto 0));
14  end decoder;
15
16  architecture Behavioral of decoder is
17
18  begin
19
20     y <= "00000001" when sel = "000" else
21          "00000010" when sel = "001" else
22          "00000100" when sel = "010" else
23          "00001000" when sel = "011" else
24          "00010000" when sel = "100" else
25          "00100000" when sel = "101" else
26          "01000000" when sel = "110" else
27          "10000000";
28
29  end Behavioral;
30
```

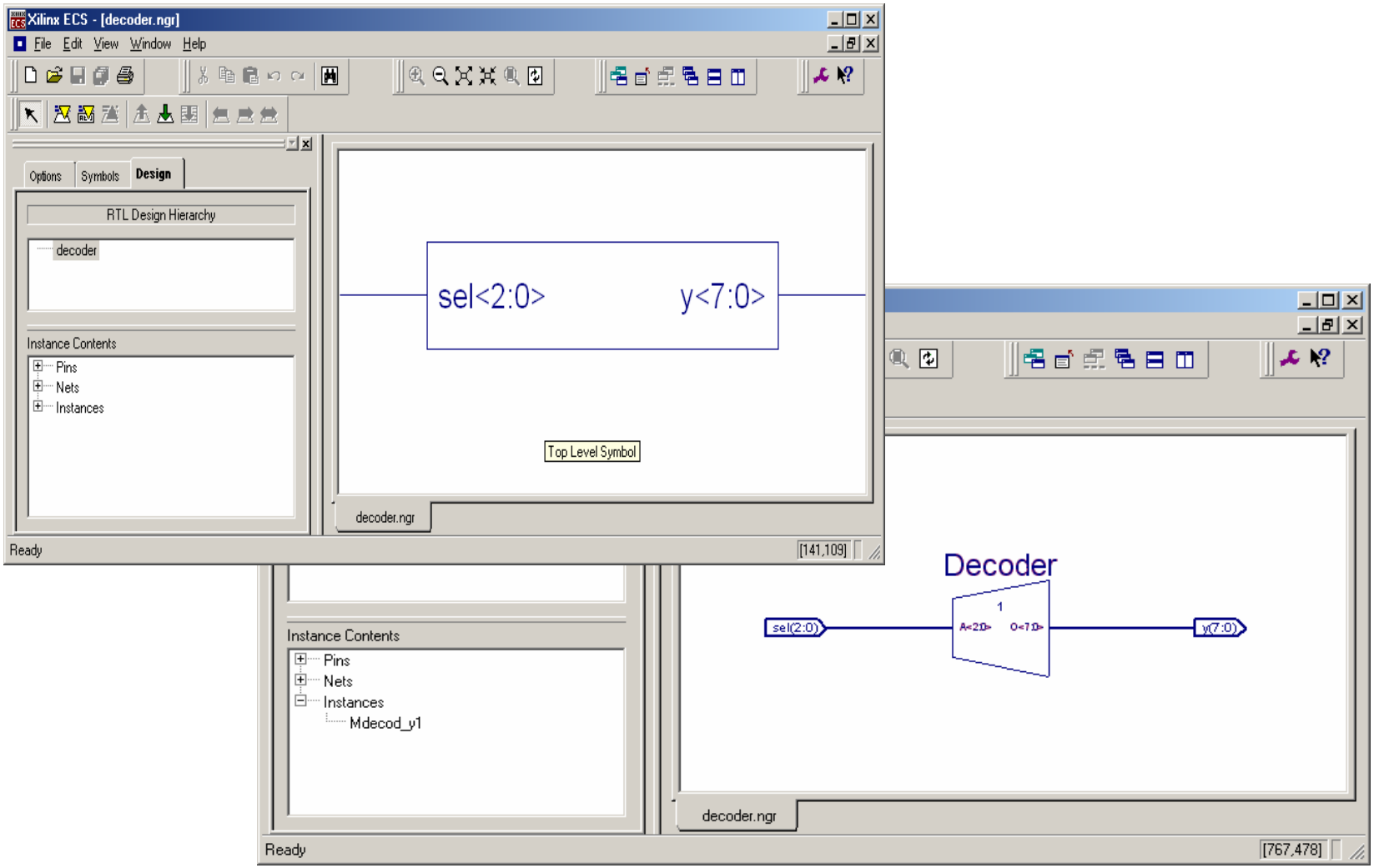
For Help, press F1

Synthesizing the Design

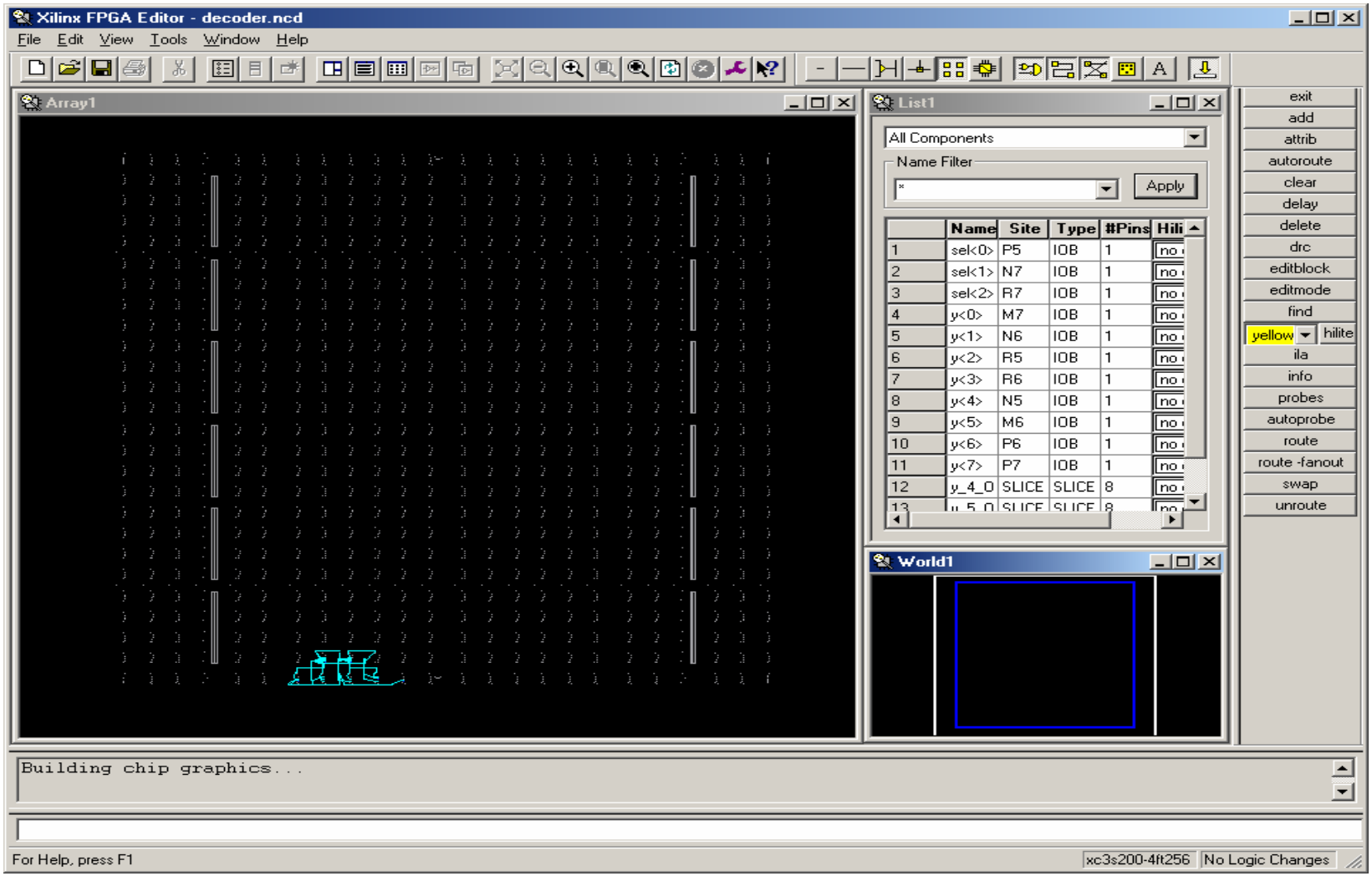



```
-  =========================================================================*
   HDL Synthesis
   *=========================================================================
-  Synthesizing Unit <decoder>.
-  Related source file is c:/ee574/decoder/decoder.vhd.
-  Found 1-of-8 decoder for signal <y>.
-  Summary:       inferred   1 Decoder(s).
-  Unit <decoder> synthesized.
```

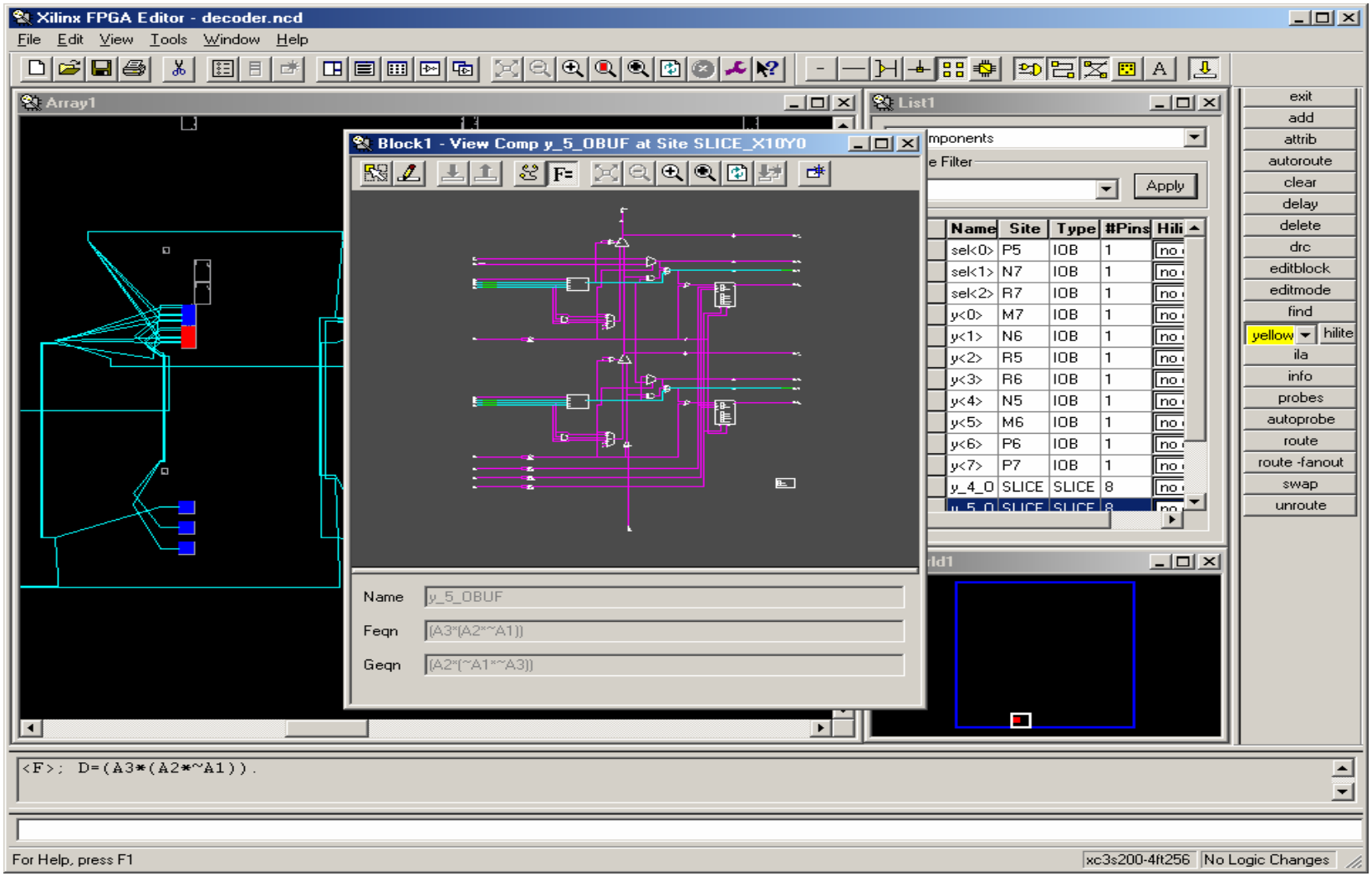
View the Schematic Representation



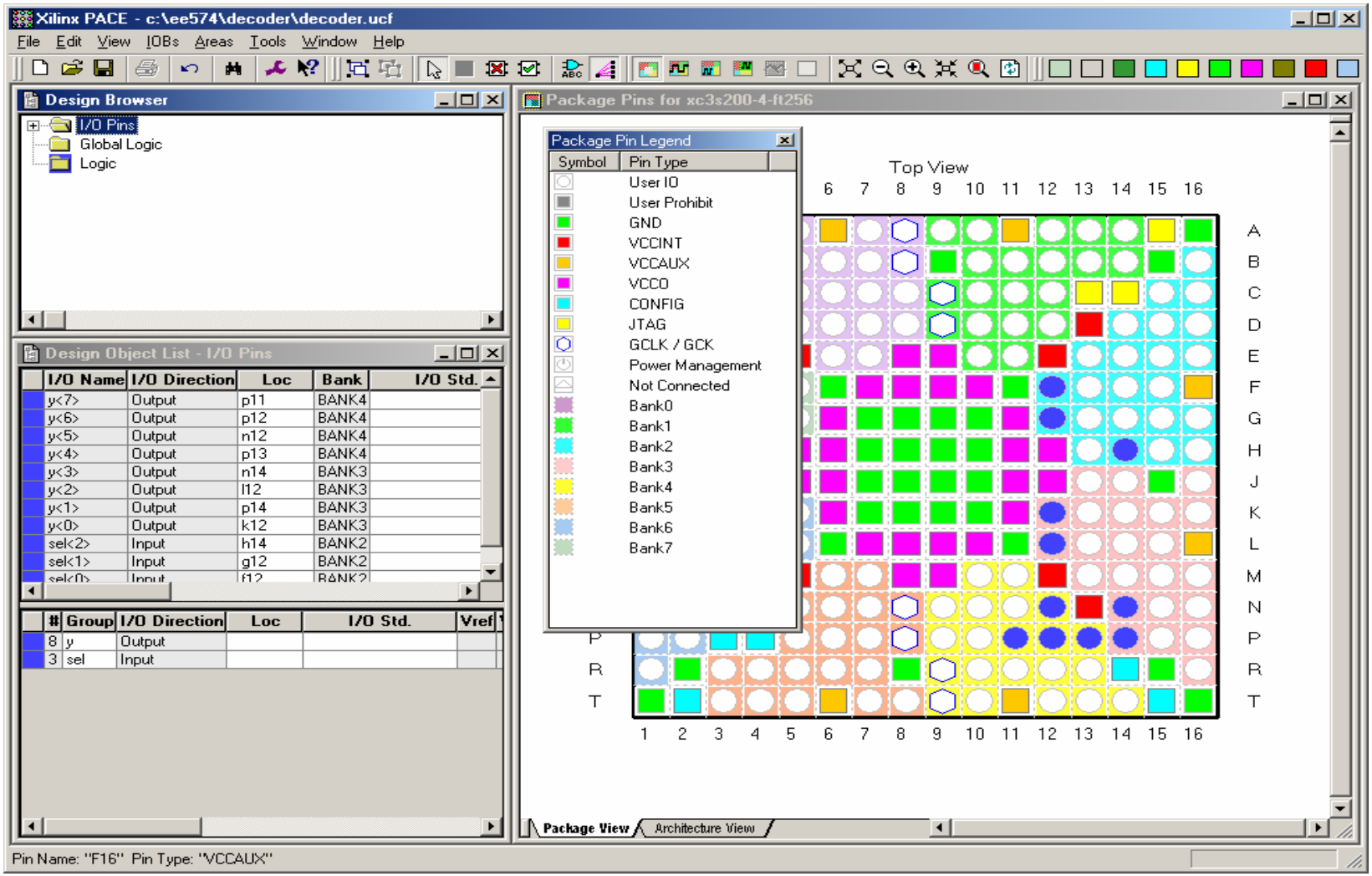
Decoder Implemented on FPGA



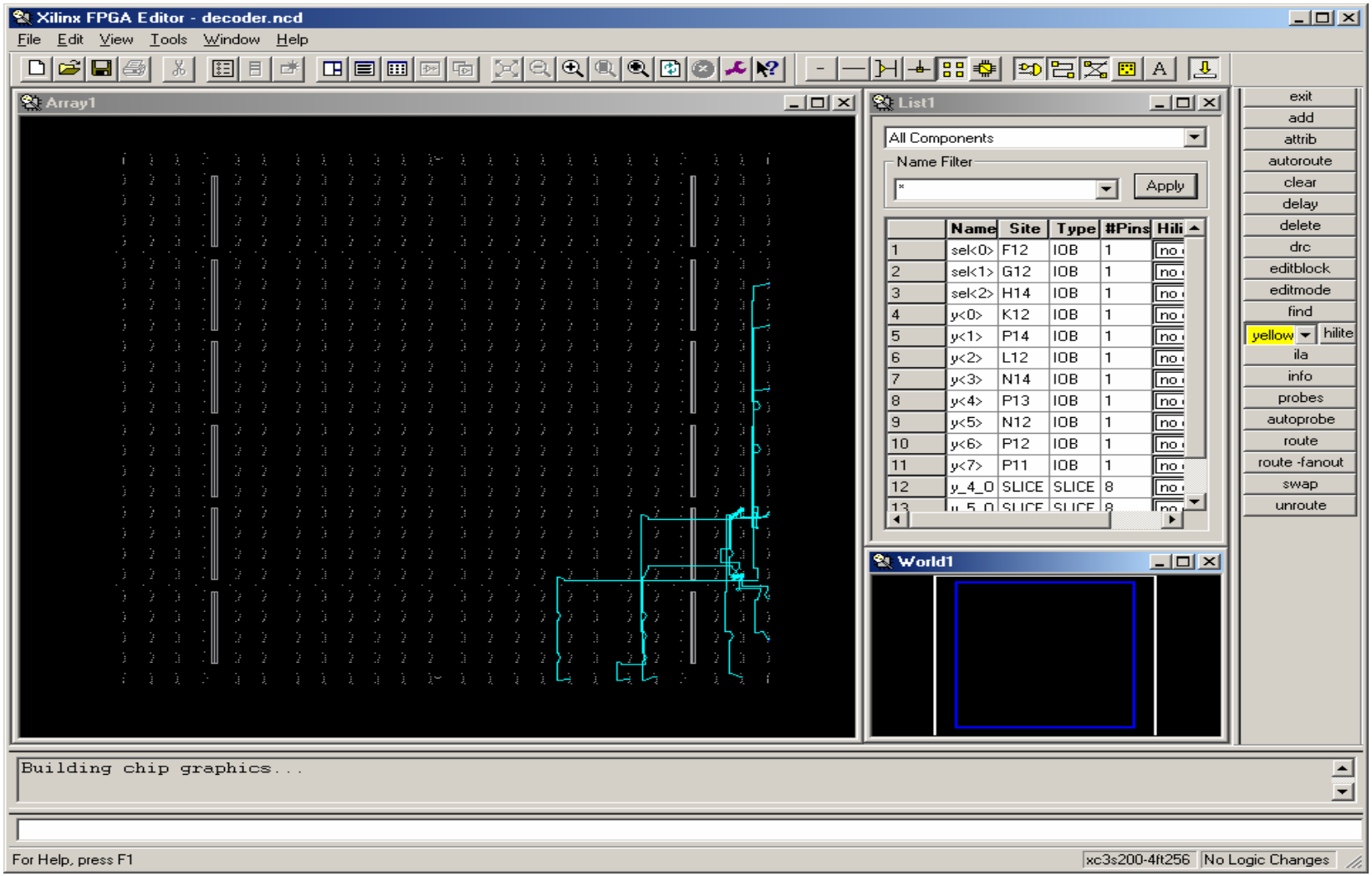
Zooming in on Logic Slice



Assigning Package Pins



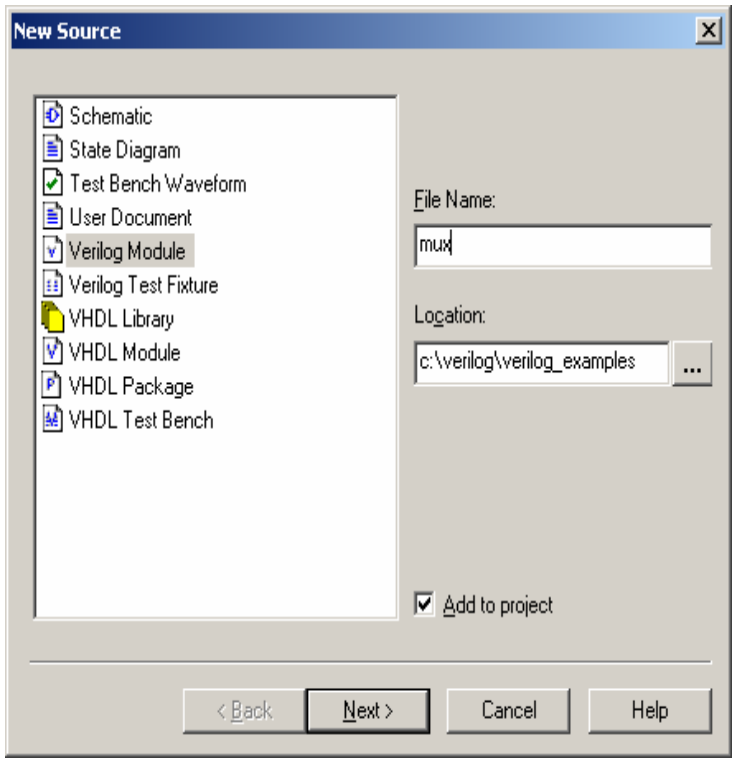
New Implementation to Match Target

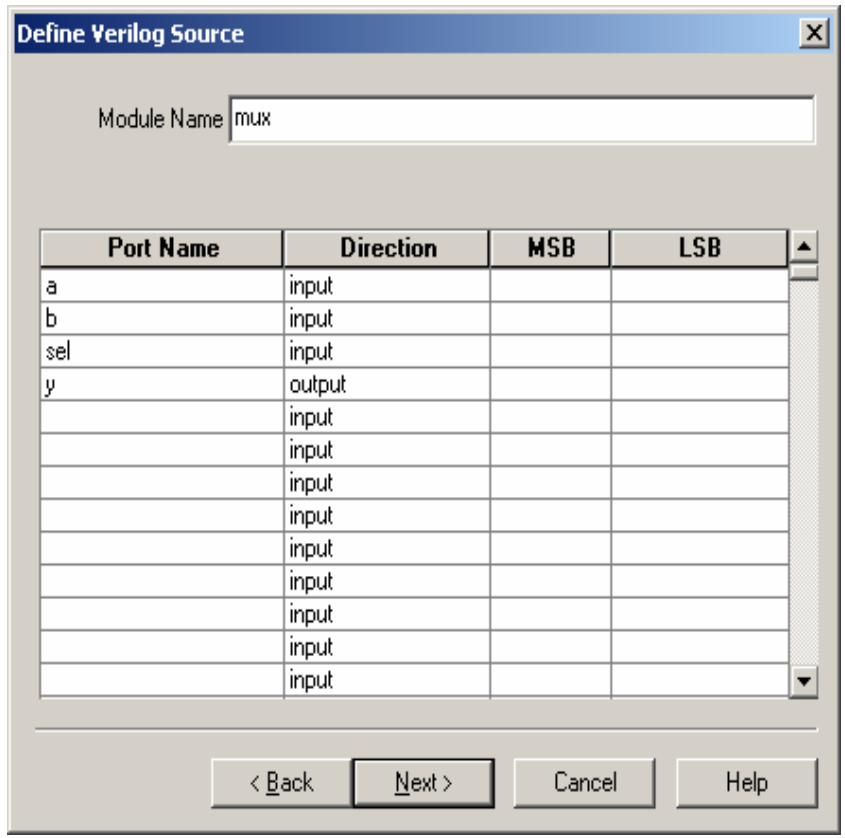


Verilog Background

- 1983: Gateway Design Automation released Verilog HDL "Verilog" and simulator
- 1985: Verilog enhanced version – "Verilog-XL"
- 1987: Verilog-XL becoming more popular (same year VHDL released as IEEE standard)
- 1989: Cadence bought Gateway
- 1995: Verilog adopted by IEEE as standard 1364
  - Verilog HDL, Verilog 1995
- 2001: First major revision (cleanup and enhancements)
  - Standard 1364-2001 (or Verilog 2001)
- System Verilog under development

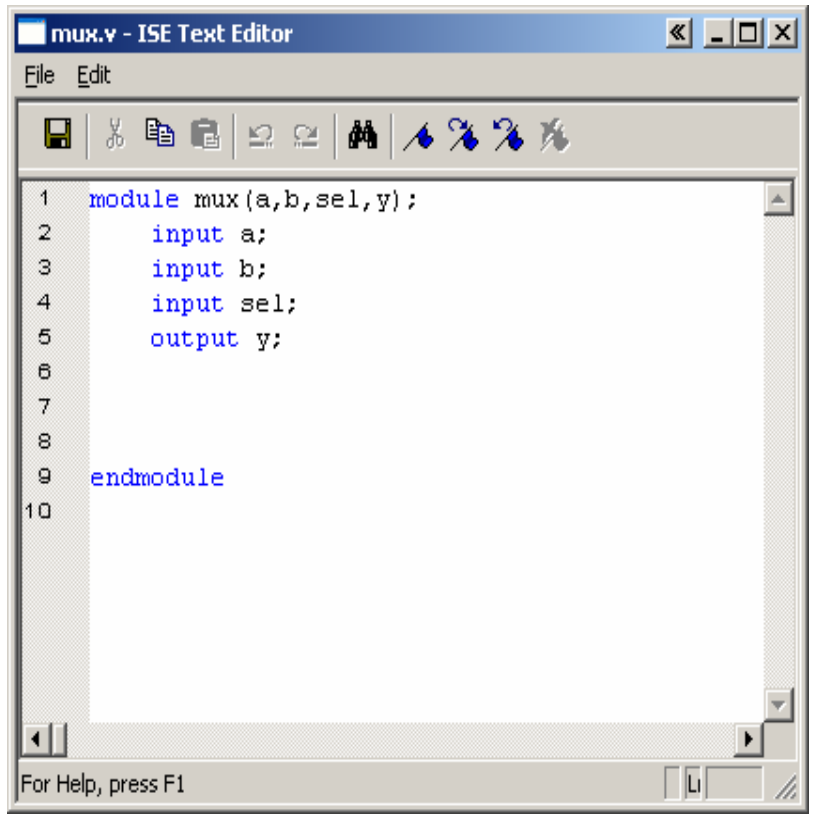
Create Verilog Module





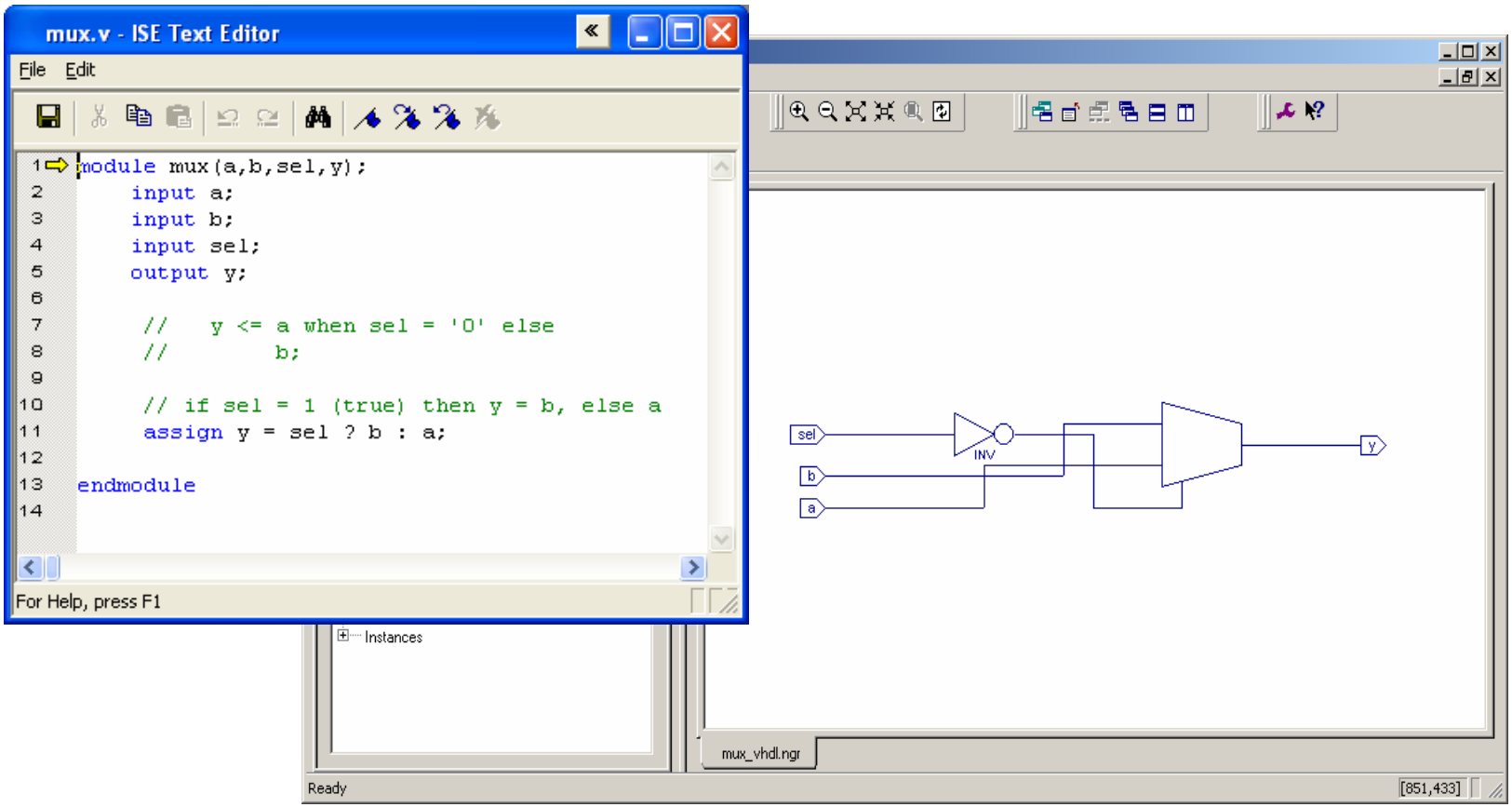
Module Created

- No separate entity and arch – just module
- Ports can be input, output, or inout
- Note: Verilog 2001 has alternative port style:
  - `(input a, b, sel, output y);`



Add Assign Statement

- Similar to VHDL conditional signal assignment – continuous assignment
- Exactly same hardware produced



Verilog - General Comments

- VHDL is like ADA and Pascal in style
  - Strongly typed – more robust
- Verilog is more like the ‘C’ language
- Verilog is case sensitive
- White space is OK (tabs, new lines, etc)
- Statements terminated with semicolon (;)
- Verilog statements between
  - module and endmodule
- Comments // single line and /* and */

Verilog Logic

- Four-value logic system
  - 0 – logic zero, or false condition
  - 1 – logic 1, or true condition
  - x, X – unknown logic value
  - z, Z - high-impedance state
- Number formats
  - b, B binary
  - d, D decimal (default)
  - h, H hexadecimal
  - o, O octal
- 16'H789A – 16-bit number in hex format
- 1'b0 – 1-bit

Verilog and VHDL – Reminder

- VHDL - like Pascal and Ada programming languages
- Verilog - more like ‘C’ programming language
- But remember they are Hardware Description Languages - They are NOT programming languages
  - FPGAs do NOT contain an hidden microprocessor or interpreter or memory that executes the VHDL or Verilog code
  - Synthesis tools prepare a hardware design that is *inferred* from the behavior described by the HDL
  - A bit stream is transferred to the programmable device to configure the device
  - No shortcuts! Need to understand combinational/sequential logic
- Uses subset of language for synthesis
- Check - could you design circuit from description?