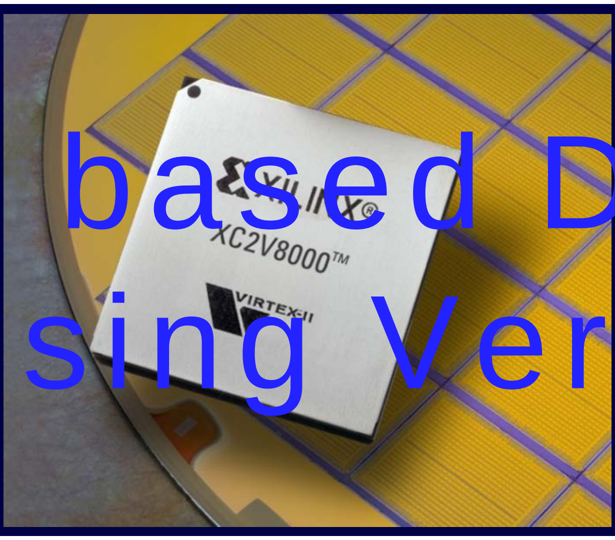
Training Course
on

FPGA based Digital Design using Verilog HDL



By

Nauman Mir
(HDL Designer)

**** Organized by*** **Skill Development Council,**
(*Ministry of Labour, Manpower and overseas Pakistani)*
Govt. of Pakistan.

Agenda – Course Introduction

- Introduction
- Why do we choose FPGA based solutions ?
- FPGA based Design Flow
- Efficient HDL Implementation Techniques
- Embedded Controllers in FPGAs
- Real-Time Debugging on FPGAs
- FPGA in Real World
- Motivation
- Course Strategy
- Q & A

FPGA based Digital Design using Verilog HDL

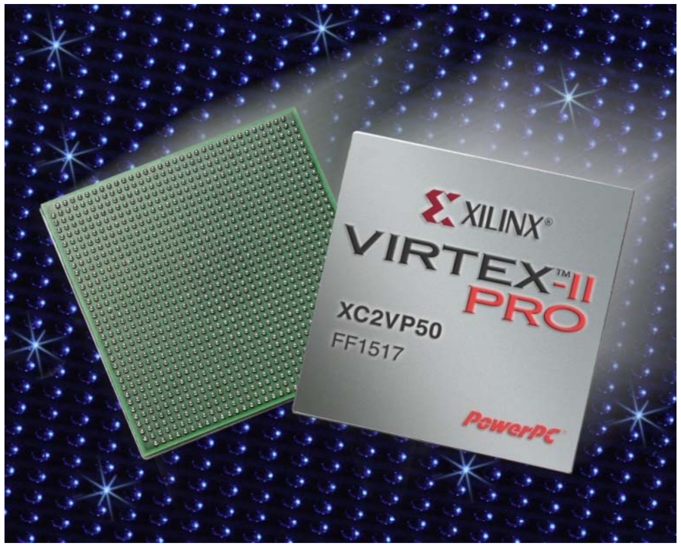
Introduction

Integrated Circuits

- Uses for digital IC technology today:
  - *Standard Microprocessors*
    - Used in desktop PC and embedded applications
  - *Memory Chips (RAMs)*
  - *Application Specific ICs (ASICs)*
    - Custom design to match particular applications
    - Can be optimized for low power, low cost, high performance

➢ *Field Programmable Logic Devices (FPGAs, CPLDs)*

- Customized to particular applications
- Reconfigure device
- Short time to market

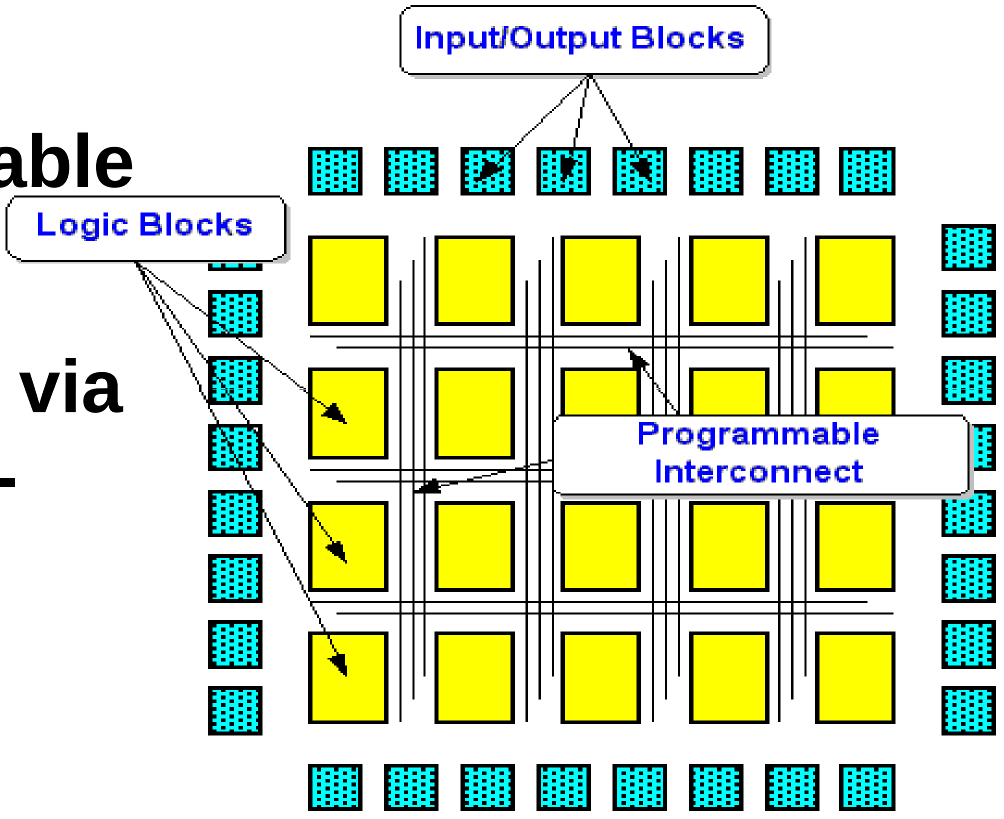


Xilinx FPGA

FPGA Basics

- **FPGA Basics**
  - FPGA stands for "Field Programmable Gate Array".
  - FPGA can be categorized as:
    - → Fuse/Anti-fuse based (ex: Actel)
    - → SRAM Based (ex: Xilinx, Altera)
  - SRAM Based FPGAs are most popular due to its reconfigurable feature and speed
  - World Popular Xilinx FPGA is typically an SRAM-based device
  - Number of System Gates, Speed Grade, Frequency, No of I/Os and Build in features etc are basic identity of any FPGA

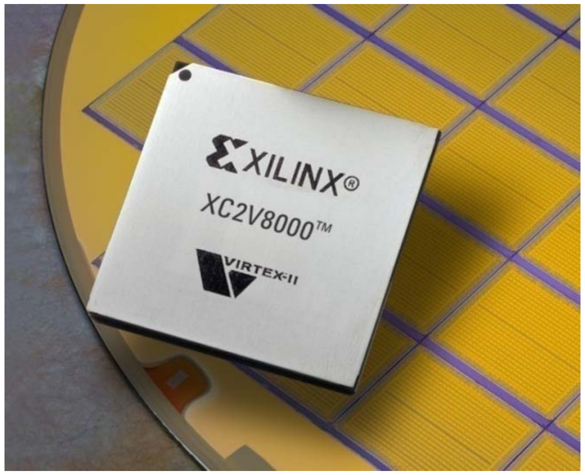
- **FPGA is a Programmable Integrated Circuit consisting of:**
  - **An internal array of Configurable Logic Blocks (CLBs).**
  - **A ring of programmable input/output blocks.**
  - **Connected together via programmable inter-connection.**



Basic FPGA Internal Architecture

➢ *Some Popular Xilinx FPGAs:*

- Spartan II/IIE/III, Virtex^{TM}, Virtex-II/IV and the Monster Virtex^{TM}-6 devices.
- Range of Xilinx FPGAs from Few 10K Gates to Multi-Million Gates with Embedded Cores.



Verilog HDL Basics

- **Verilog HDL Basics**
  - **Verilog HDL is a Hardware Description Language. It describes the hardware.**
  - **It is not a programming & procedural language.**
  - **Verilog HDL is a concurrent language**
  - **It enables specification of a digital system at a range of levels of abstraction:**
    Switches, Gates, RTL, and higher
  - **Open Verilog International (OVI) IEEE 1364**

- ***Why use an HDL ?***
  - **Describe complex designs (millions of gates)**
  - **Input to synthesis tools (synthesizable subset)**
  - **Design exploration with simulation**

- ***Why not use a general purpose language***
  - **Support for structure and instantiation**
  - **Support for describing bit-level behavior**
  - **Support for timing**
  - **Support for concurrency**

➢ *Verilog vs. VHDL*

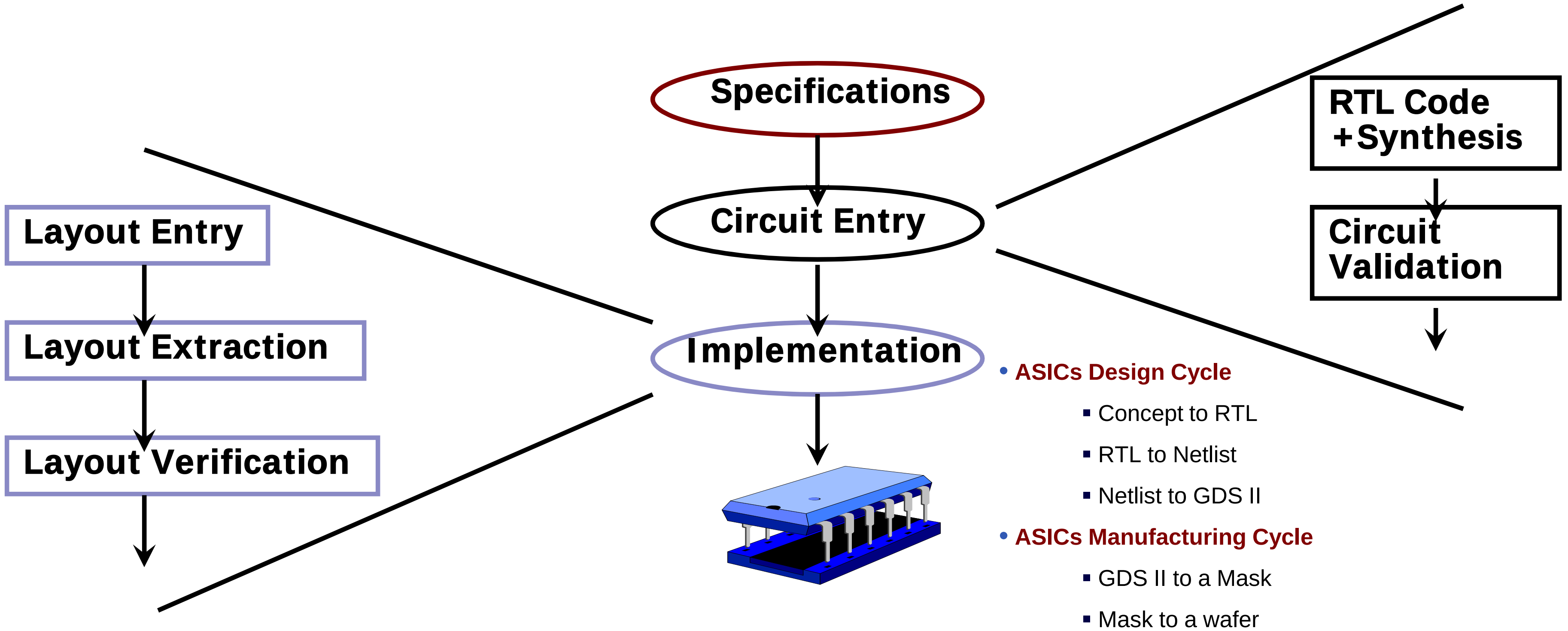
- **Verilog is relatively simple and close to C**
- **VHDL is complex**
- **Verilog has 60% of the world digital design market (larger share in US)**

FPGA based Digital Design using Verilog HDL

Why do we choose FPGA based solutions?

Why do we choose FPGA based solutions ?

- **We choose FPGA based solution due to following reasons :**
  - ***Prefer FPGA design flow over ASIC design flow***
    - **Traditional ASIC Design Flow**

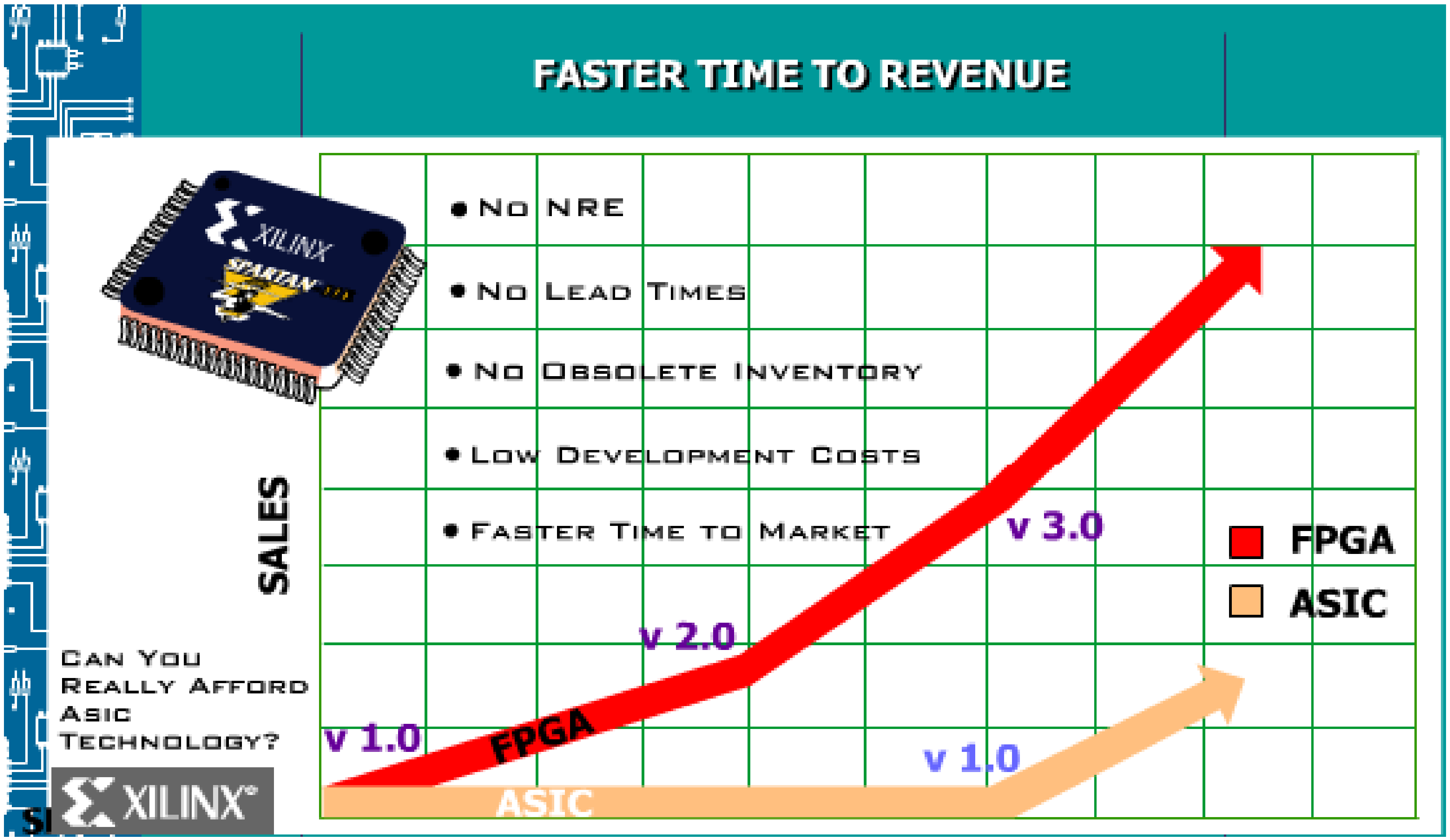


- ASICs Design Cycle
  - Concept to RTL
  - RTL to Netlist
  - Netlist to GDS II
- ASICs Manufacturing Cycle
  - GDS II to a Mask
  - Mask to a wafer
  - Wafer to a Packaged part

➢ Why do we prefer FPGA over ASIC ?

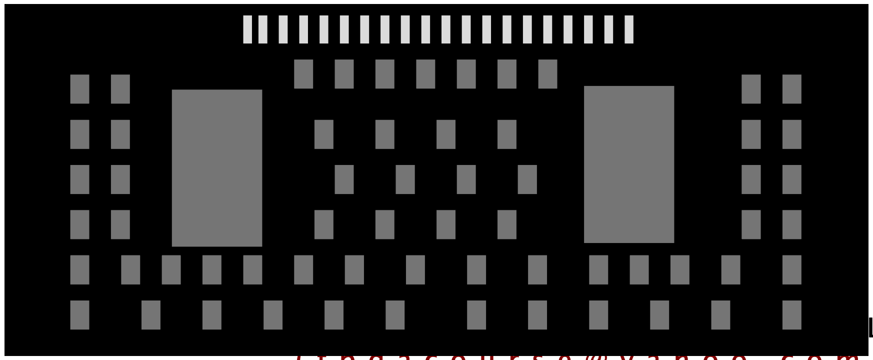
- Customized to particular applications
- Reconfigure hardware
- Short time to market
- When product has hundreds or thousands components, FPGA is preferred. Mostly in Pakistan, FPGA based solution is preferred
- Setup cost of ASIC is too High. On the other hand min cost of FPGA is about $5
- There are FPGA based training boards for rapid prototyping

FPGA Vs ASIC

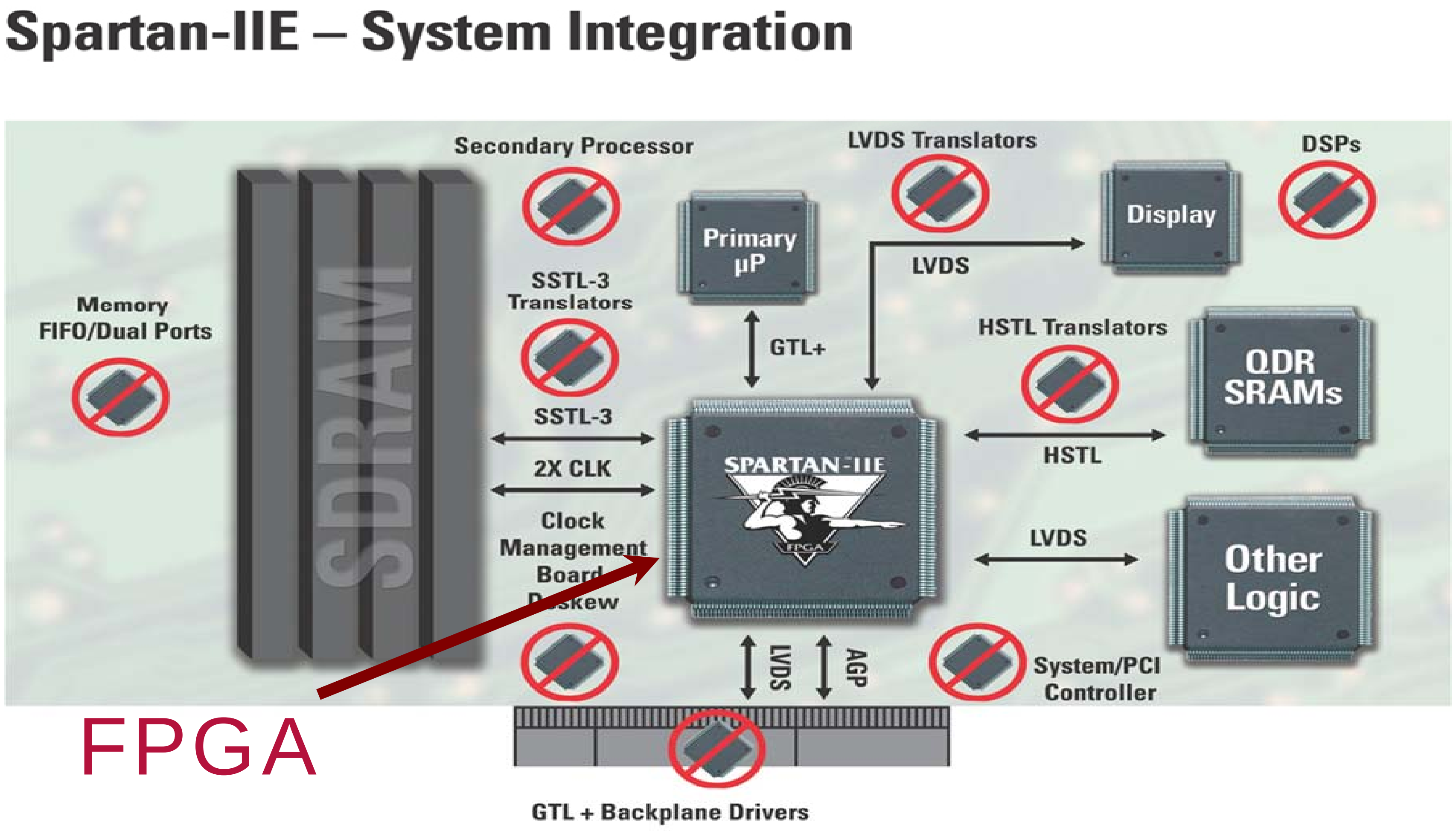

FASTER TIME TO REVENUE
XILINX
SPARTAN
• No NRE
• No Lead Times
• No Obsolete Inventory
• Low Development Costs
• Faster Time to Market
SALES
Can You Really Afford ASIC Technology?
XILINX
v 1.0
FPGA
v 2.0
v 3.0
v 1.0
ASIC
🟥 FPGA
🟧 ASIC

➢ *Design Optimization and Specific Chip Dependency Removal*

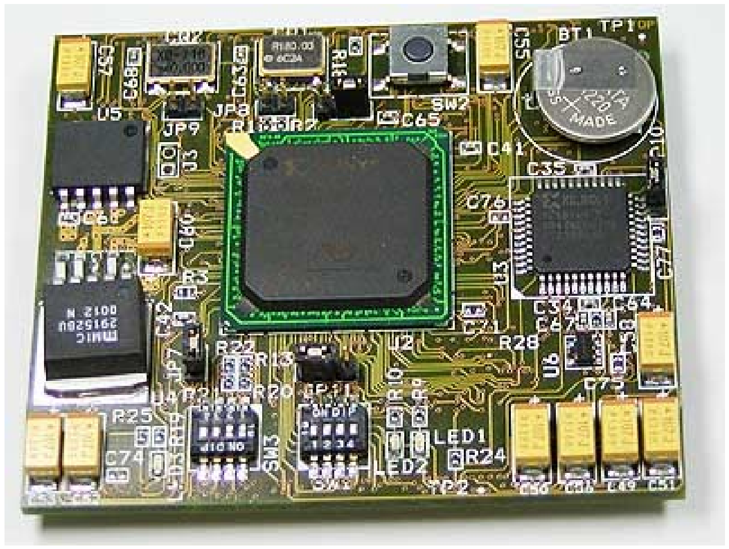
- In latest FPGAs, there are multi-million system gates
- FPGA has great resources having RAM Blocks, Arithmetic Operations, Signaling Compatibility and many more…
- FPGAs are introduced as an alternative to custom ICs for implementing glue logic:
  - improved density relative to discrete SSI/MSI components (within around 100x of custom ICs)

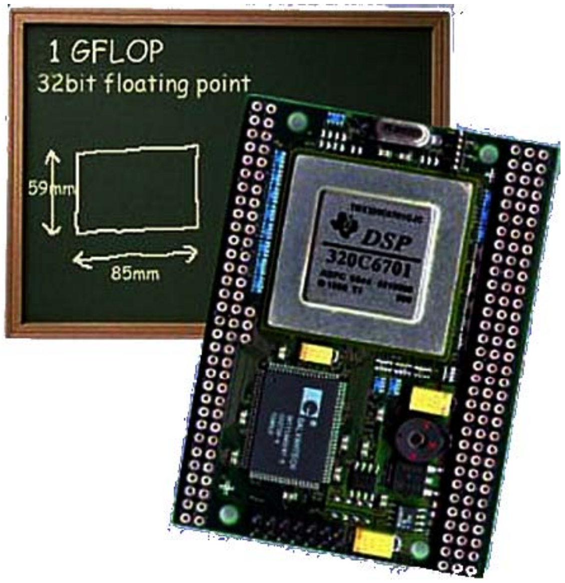


System Integration



FPGA versus Processor Technology



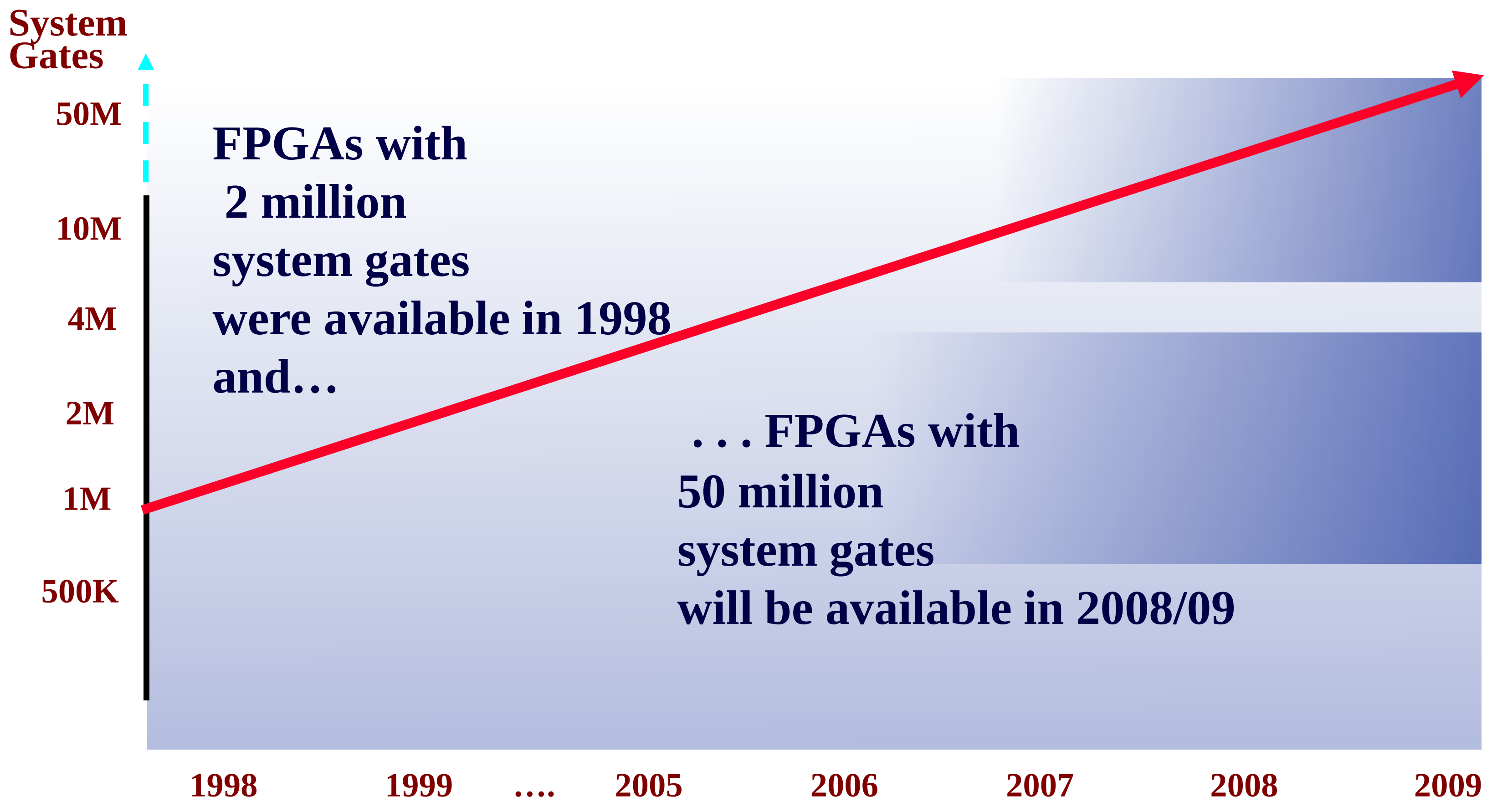


+ Configurable Hardware
+ Parallel processing in Hardware
+ Huge Hardware size having Million of Gates
- no Floating-point Arithmetic
- long turn-around times (Synthesis, Simulation, Place and Route)
- Memory managment

+ Floating-point Arithmetic
+ easy C programming tools
+ fast Debug- and Test options
+ Memory managment
- Serial Codeprocessing
- Limited Hardware size as compare to FPGAs

FPGA Technology

Moore's Law: In 1965, Gordon Moore noted that the number of transistors on a chip doubled every 18 to 24 months. He made a prediction that semiconductor technology will double its effectiveness every 18 months.



Virtex-6 Family FPGAs

VIRTEX 6

		Virtex-6 LXT FPGAs: Optimized for High-performance Logic and DSP with Low-power Serial Connectivity (1.0 Volt, 0.9 Volt)							Virtex-6 SXT FPGAs: Optimized for Ultra High-performance DSP with Low-power Serial Connectivity (1.0 Volt, 0.9 Volt)		Virtex-6 HXT FPGAs: Optimized for Communications Systems Requiring Highest-bandwidth Serial Connectivity (1.0 Volt)			
	Part Number	XC6VLX75T	XC6VLX130T	XC6VLX195T	XC6VLX240T	XC6VLX365T	XC6VLX550T	XC6VLX760	XC6VSX315T	XC6VSX475T	XC6VHX250T	XC6VHX255T	XC6VHX380T	XC6VHX565T
	EasyPath™ FPGA Cost Reduction Solutions [1]	XCE6VLX75T	XCE6VLX130T	XCE6VLX195T	XCE6VLX240T	XCE6VLX365T	XCE6VLX550T	XCE6VLX760	XCE6VSX315T	XCE6VSX475T	XC6VHX250T	XC6VHX255T	XC6VHX380T	XC6VHX565T
Logic Resources	Slices [2]	11,640	20,000	31,200	37,680	56,880	85,920	118,560	49,200	74,400	39,360	39,600	59,760	88,560
	Logic Cells [3]	74,496	128,000	199,680	241,152	364,032	549,888	758,784	314,880	476,160	251,904	253,440	382,464	566,784
	CLB Flip-Flops	93,120	160,000	249,600	301,440	455,040	687,360	948,480	393,600	595,200	314,880	316,800	478,080	708,480
Memory Resources	Maximum Distributed RAM (Kbits)	1,045	1,740	3,040	3,650	4,130	6,200	8,280	5,090	7,640	3,040	3,050	4,570	6,360
	Block RAM/FIFO w/ ECC (36Kbits each)	156	264	344	416	416	632	720	704	1,064	504	516	768	912
	Total Block RAM (Kbits)	5,616	9,504	12,384	14,976	14,976	22,752	25,920	25,344	38,304	18,144	18,567	27,648	32,832
Clock Resources	Mixed Mode Clock Managers (MMCM)	6	10	10	12	12	18	18	12	18	12	12	18	18
I/O Resources [4, 5]	Maximum Single-Ended I/O	360	600	600	720	720	1,200	1,200	720	840	320	480	720	720
	Maximum Differential I/O Pairs	180	300	300	360	360	600	600	360	420	160	240	360	360
Embedded Hard IP Resources [6]	DSP48E1 Slices	288	480	640	768	576	864	864	1,344	2,016	576	576	864	864
	PCI Express® Interface Blocks	1	2	2	2	2	2	–	2	2	4	2	4	4
	10/100/1000 Ethernet MAC Blocks	4	4	4	4	4	4	–	4	4	4	2	4	4
	GTX Low-Power Transceivers	12	20	20	24	24	36	–	24	36	48	24	48	48
	GTH High-Speed Transceivers	–	–	–	–	–	–	–	–	–	–	24	24	24
Speed Grades	Commercial	-L1, -1, -2, -3	-L1, -1, -2, -3	-L1, -1, -2, -3	-L1, -1, -2, -3	-L1, -1, -2, -3	-L1, -1, -2	-L1, -1, -2	-L1, -1, -2, -3	-L1, -1, -2	-1, -2, -3	-1, -2, -3	-1, -2, -3	-1, -2
	Industrial	-L1, -1, -2	-L1, -1, -2	-L1, -1, -2	-L1, -1, -2	-L1, -1, -2	-L1, -1	-L1, -1	-L1, -1, -2	-L1, -1	-1, -2	-1, -2	-1, -2	-1
Configuration	Configuration Memory (Mbits)	25.0	41.7	58.7	70.4	91.6	137.4	176.3	99.6	149.4	76.2	76.2	114.2	153.2

Package [7]	Area	Available User I/O: SelectIO Pins [4,5] (GTX Low-power Transceivers, GTH High-speed Transceivers)												
FFA Packages (FF): flip-chip fine-pitch BGA (1.0 mm ball spacing)														
FF484	23 x 23 mm	240 (8, 0)	240 (8, 0)											
FF784	29 x 29 mm	360 (12, 0)	400 (12, 0)	400 (12, 0)	400 (12, 0)									
FF1156	35 x 35 mm		600 (20, 0)	600 (20, 0)	600 (20, 0)	600 (20, 0)			600 (20, 0)	600 (20, 0)				
FF1759	42.5 x 42.5 mm				720 (24, 0)	720 (24, 0)	840 (36, 0)		720 (24, 0)	840 (36, 0)				
FF1760	42.5 x 42.5 mm						1,200 (0, 0)	1,200 (0, 0)						
FF1154	35 x 35 mm										320 (48, 0)		320 (48, 0)	
FF1155	35 x 35 mm											440 (24, 12)	440 (24, 12)	
FF1923	45 x 45 mm											480 (24, 24)	720 (40, 24)	720 (40, 24)
FF1924	45 x 45 mm												640 (48, 24)	640 (48, 24)

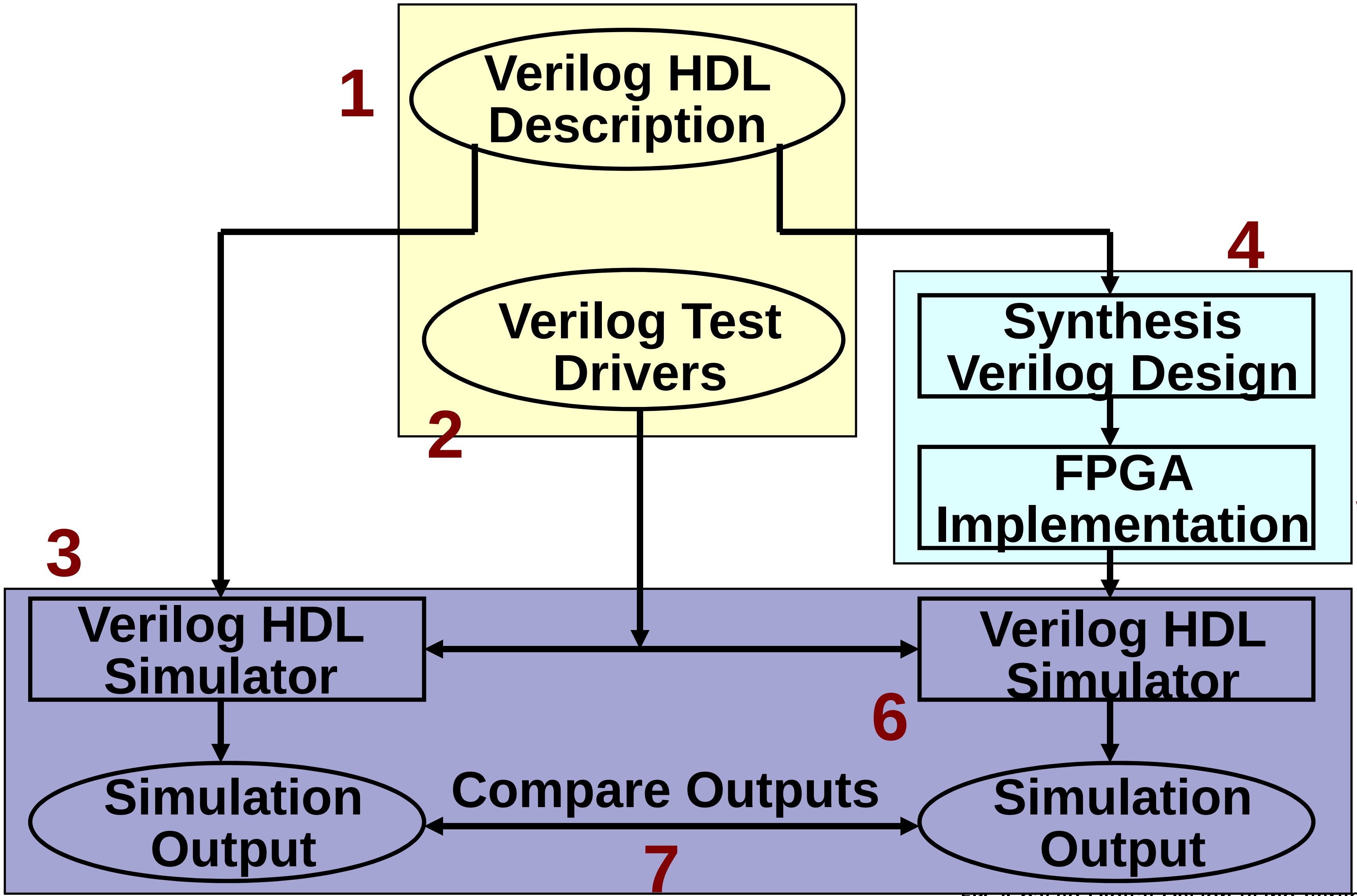
FPGA based Digital Design using Verilog HDL

FPGA based Design Flow

FPGA based Design Steps

- **Design Steps:**
  - **Verilog HDL is used for Hardware Designing**
  - **Synthesis tools typically accept only a subset of the full Verilog language constructs**
  - **Synthesis tools convert the actual design (Verilog code) to gate level netlist**
  - **Finally design is implemented into your target technology (Xilinx FPGAs)**

Design Methodology



Design Example

- **Example of 4-bit Ripple Carry Counter..**
  - **First make a design of system on paper and study its all aspects**
  - **Write a Behavioral Model in Verilog HDL of design for check its functionality**
  - **Write RTL in Verilog HDL of design**
  - **Synthesis tools convert the RTL design (Verilog code) to gate level netlist**
  - **Finally design is implemented into your target technology (Xilinx FPGAs)**
  - **Create a bit file of design that download in FPGA**

- **Software Required**
  - **Model*Sim* PE/SE/XE**
  - **Xilinx ISE 3.1i or Above**
- **Hardware Required**
  - **FPGA Boards (Xilinx Spartan/Virtex)**
  - **Xilinx Parallel / USB-Platform Cable**

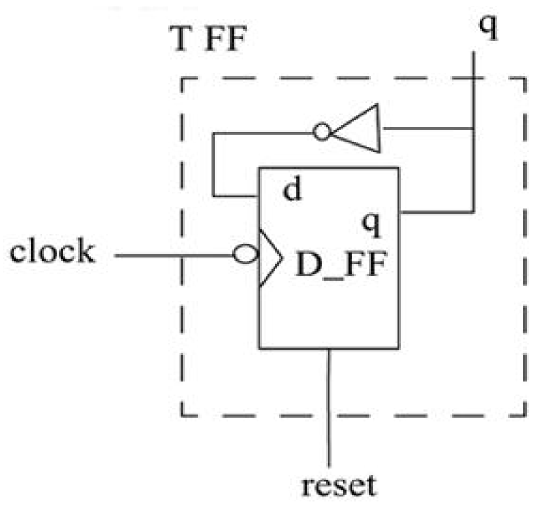
➢ Design *4-bit Ripple Carry Counter*

→ **First we define its input and outputs ports (in size, numbers…) and assign a unique name of each port. Here**

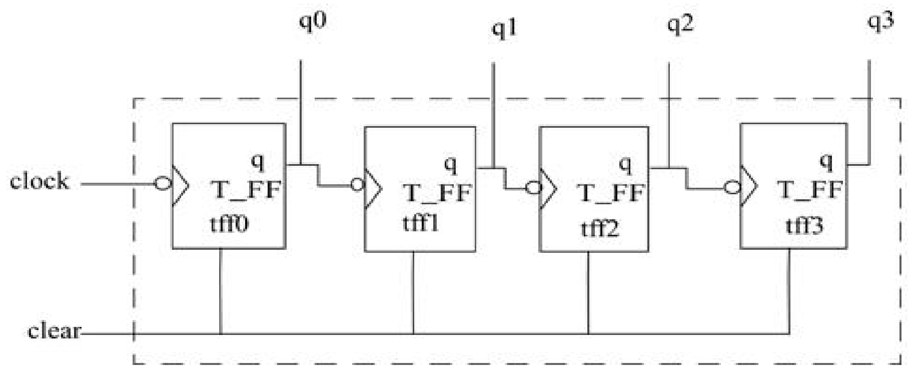
inputs: **clk, clear**

output: **q0, q1, q2, q3**

(* These are all single bit)

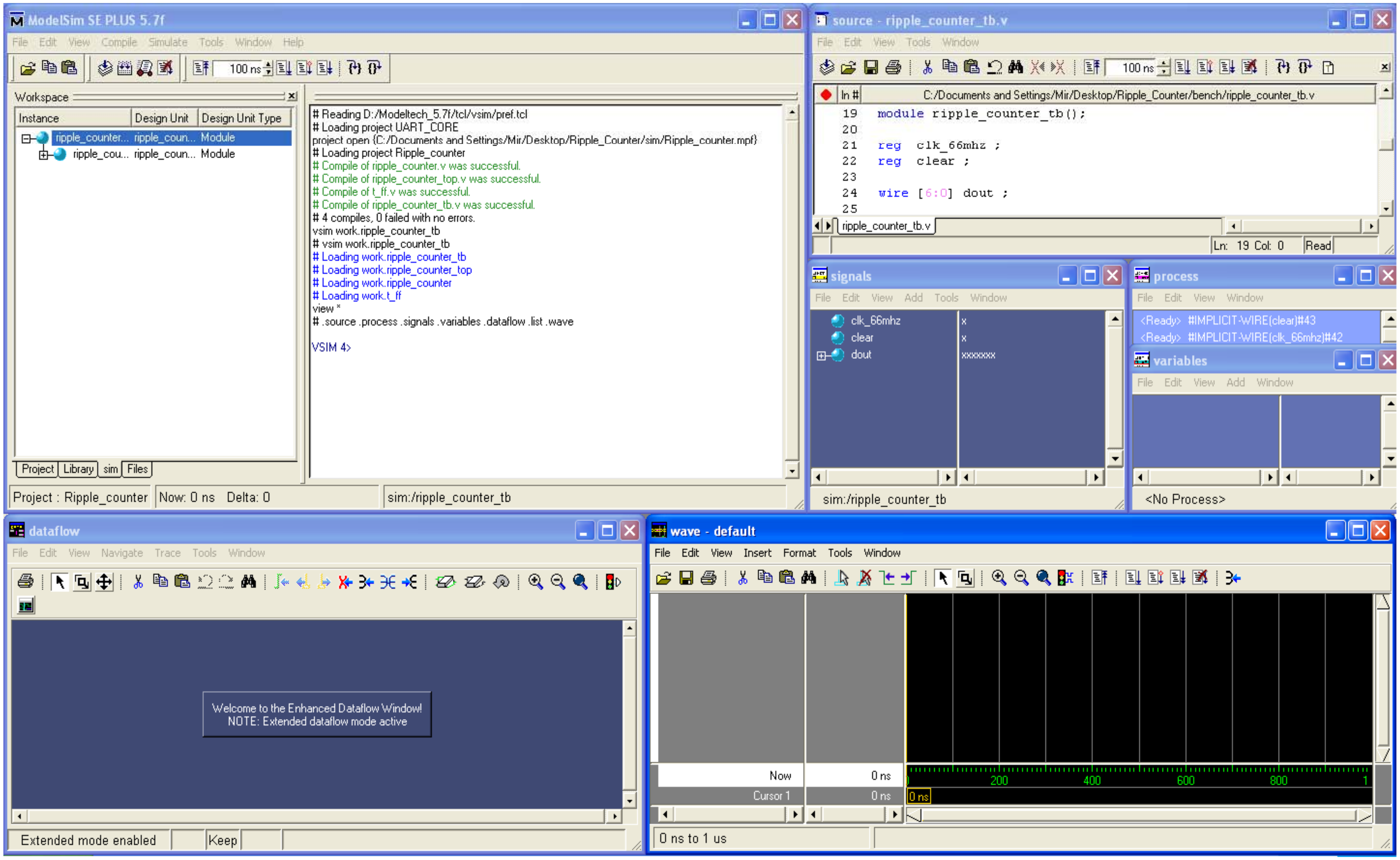


T-Flipflop



4-bit Ripple Carry Counter

➢ Model*Sim* main Window



Instance	Design Unit	Design Unit Type
ripple_counter...	ripple_coun...	Module
ripple_cou...	ripple_coun...	Module

```
# Reading D:/Modeltech_5.7f/tcl/vsim/pref.tcl
# Loading project UART_CORE
project open {C:/Documents and Settings/Mir/Desktop/Ripple_Counter/sim/Ripple_counter.mpf}
# Loading project Ripple_counter
# Compile of ripple_counter.v was successful.
# Compile of ripple_counter_top.v was successful.
# Compile of t_ff.v was successful.
# Compile of ripple_counter_tb.v was successful.
# 4 compiles, 0 failed with no errors.
vsim work.ripple_counter_tb
# vsim work.ripple_counter_tb
# Loading work.ripple_counter_tb
# Loading work.ripple_counter_top
# Loading work.ripple_counter
# Loading work.t_ff
view *
# .source .process .signals .variables .dataflow .list .wave

VSIM 4>
```

Project : Ripple_counter Now: 0 ns Delta: 0 sim:/ripple_counter_tb

C:/Documents and Settings/Mir/Desktop/Ripple_Counter/bench/ripple_counter_tb.v

```
module ripple_counter_tb();

reg  clk_66mhz ;
reg  clear ;

wire [6:0] dout ;
```

ripple_counter_tb.v Ln: 19 Col: 0 Read

clk_66mhz	x
clear	x
dout	xxxxxxx

sim:/ripple_counter_tb

```
<Ready>  #IMPLICIT-WIRE(clear)#43
<Ready>  #IMPLICIT-WIRE(clk_66mhz)#42
```

<No Process>

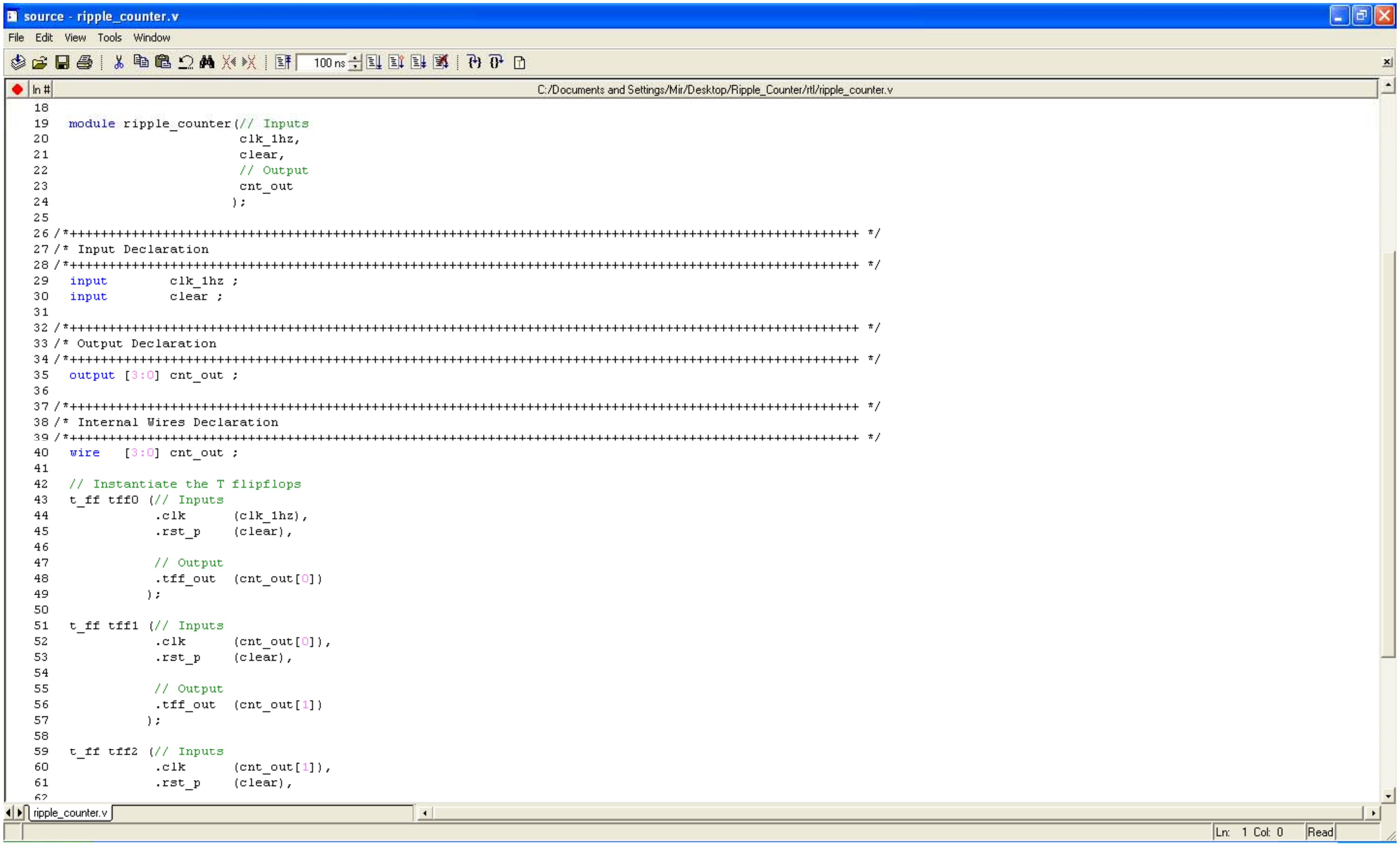
Welcome to the Enhanced Dataflow Window!
NOTE: Extended dataflow mode active

Extended mode enabled Keep

0 ns to 1 us

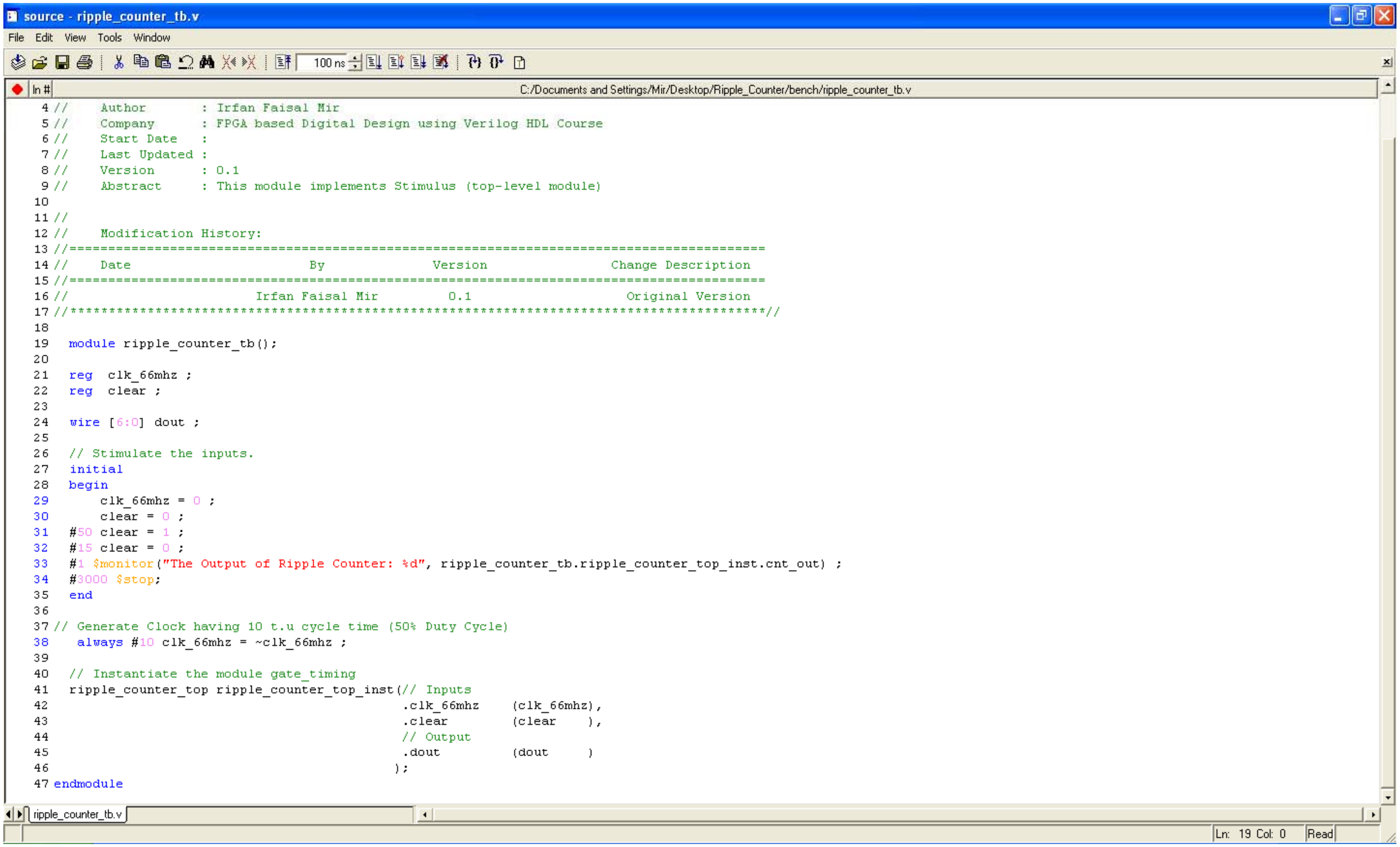
Introductory Class

➢ Design *4-bit Ripple Carry Counter in Verilog HDL*


source - ripple_counter.v
File Edit View Tools Window
100 ns
ln #
C:/Documents and Settings/Mir/Desktop/Ripple_Counter/rtl/ripple_counter.v
ripple_counter.v
Ln: 1 Col: 0
Read

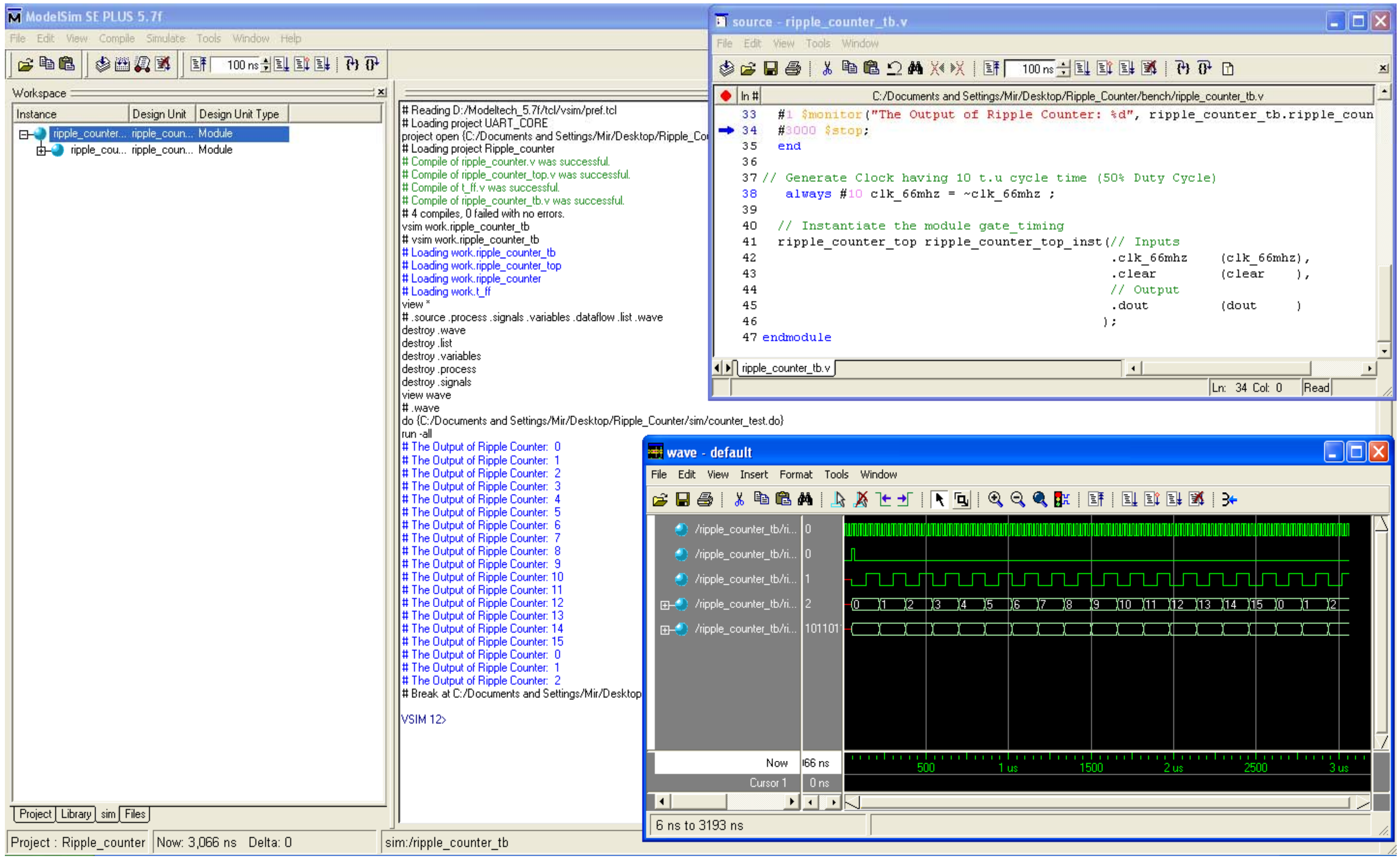
```verilog

module ripple_counter(// Inputs
                      clk_1hz,
                      clear,
                      // Output
                      cnt_out
                     );

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* Input Declaration
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
 input        clk_1hz ;
 input        clear ;

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* Output Declaration
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
 output [3:0] cnt_out ;

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* Internal Wires Declaration
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
 wire   [3:0] cnt_out ;

 // Instantiate the T flipflops
 t_ff tff0 (// Inputs
           .clk      (clk_1hz),
           .rst_p    (clear),

           // Output
           .tff_out  (cnt_out[0])
          );

 t_ff tff1 (// Inputs
           .clk      (cnt_out[0]),
           .rst_p    (clear),

           // Output
           .tff_out  (cnt_out[1])
          );

 t_ff tff2 (// Inputs
           .clk      (cnt_out[1]),
           .rst_p    (clear),
```

➢ Test Bench for Design


source - ripple_counter_tb.v
File Edit View Tools Window
100 ns
ln #
C:/Documents and Settings/Mir/Desktop/Ripple_Counter/bench/ripple_counter_tb.v
ripple_counter_tb.v
Ln: 19 Col: 0
Read

```verilog
//    Author        : Irfan Faisal Mir
//    Company       : FPGA based Digital Design using Verilog HDL Course
//    Start Date    :
//    Last Updated  :
//    Version       : 0.1
//    Abstract      : This module implements Stimulus (top-level module)

//
//    Modification History:
//==========================================================================================
//    Date                        By              Version                 Change Description
//==========================================================================================
//                      Irfan Faisal Mir         0.1                      Original Version
//******************************************************************************************//

  module ripple_counter_tb();

  reg  clk_66mhz ;
  reg  clear ;

  wire [6:0] dout ;

  // Stimulate the inputs.
  initial
  begin
      clk_66mhz = 0 ;
      clear = 0 ;
  #50 clear = 1 ;
  #15 clear = 0 ;
  #1 $monitor("The Output of Ripple Counter: %d", ripple_counter_tb.ripple_counter_top_inst.cnt_out) ;
  #3000 $stop;
  end

// Generate Clock having 10 t.u cycle time (50% Duty Cycle)
   always #10 clk_66mhz = ~clk_66mhz ;

  // Instantiate the module gate_timing
  ripple_counter_top ripple_counter_top_inst(// Inputs
                                           .clk_66mhz    (clk_66mhz),
                                           .clear        (clear    ),
                                           // Output
                                           .dout         (dout     )
                                          );
endmodule
```

➢ Model*Sim* main Window – Design results



```
# Reading D:/Modeltech_5.7f/tcl/vsim/pref.tcl
# Loading project UART_CORE
project open {C:/Documents and Settings/Mir/Desktop/Ripple_Co
# Loading project Ripple_counter
# Compile of ripple_counter.v was successful.
# Compile of ripple_counter_top.v was successful.
# Compile of t_ff.v was successful.
# Compile of ripple_counter_tb.v was successful.
# 4 compiles, 0 failed with no errors.
vsim work.ripple_counter_tb
# vsim work.ripple_counter_tb
# Loading work.ripple_counter_tb
# Loading work.ripple_counter_top
# Loading work.ripple_counter
# Loading work.t_ff
view *
# .source .process .signals .variables .dataflow .list .wave
destroy .wave
destroy .list
destroy .variables
destroy .process
destroy .signals
view wave
# .wave
do {C:/Documents and Settings/Mir/Desktop/Ripple_Counter/sim/counter_test.do}
run -all
# The Output of Ripple Counter:  0
# The Output of Ripple Counter:  1
# The Output of Ripple Counter:  2
# The Output of Ripple Counter:  3
# The Output of Ripple Counter:  4
# The Output of Ripple Counter:  5
# The Output of Ripple Counter:  6
# The Output of Ripple Counter:  7
# The Output of Ripple Counter:  8
# The Output of Ripple Counter:  9
# The Output of Ripple Counter: 10
# The Output of Ripple Counter: 11
# The Output of Ripple Counter: 12
# The Output of Ripple Counter: 13
# The Output of Ripple Counter: 14
# The Output of Ripple Counter: 15
# The Output of Ripple Counter:  0
# The Output of Ripple Counter:  1
# The Output of Ripple Counter:  2
# Break at C:/Documents and Settings/Mir/Desktop

VSIM 12>
```

C:/Documents and Settings/Mir/Desktop/Ripple_Counter/bench/ripple_counter_tb.v

```verilog
  #1 $monitor("The Output of Ripple Counter: %d", ripple_counter_tb.ripple_coun
  #3000 $stop;
  end

// Generate Clock having 10 t.u cycle time (50% Duty Cycle)
   always #10 clk_66mhz = ~clk_66mhz ;

  // Instantiate the module gate_timing
  ripple_counter_top ripple_counter_top_inst(// Inputs
                                             .clk_66mhz     (clk_66mhz),
                                             .clear         (clear    ),
                                             // Output
                                             .dout          (dout     )
                                            );
endmodule
```

Ln: 34 Col: 0 Read

Instance	Design Unit	Design Unit Type
ripple_counter...	ripple_coun...	Module
ripple_cou...	ripple_coun...	Module

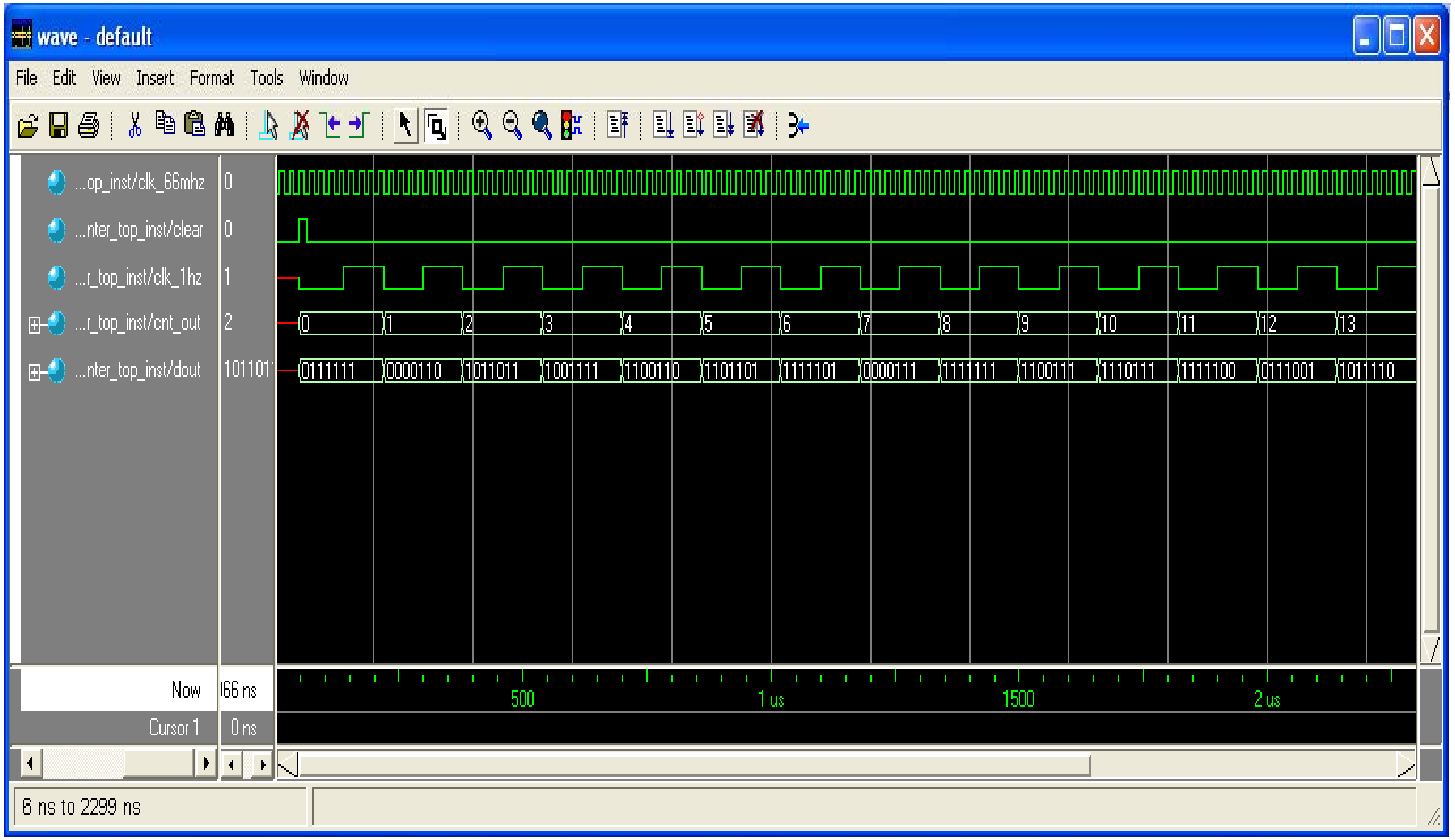
/ripple_counter_tb/ri...	0	
/ripple_counter_tb/ri...	0	
/ripple_counter_tb/ri...	1	
/ripple_counter_tb/ri...	2	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1 2
/ripple_counter_tb/ri...	101101	
Now	166 ns	
Cursor 1	0 ns	

6 ns to 3193 ns

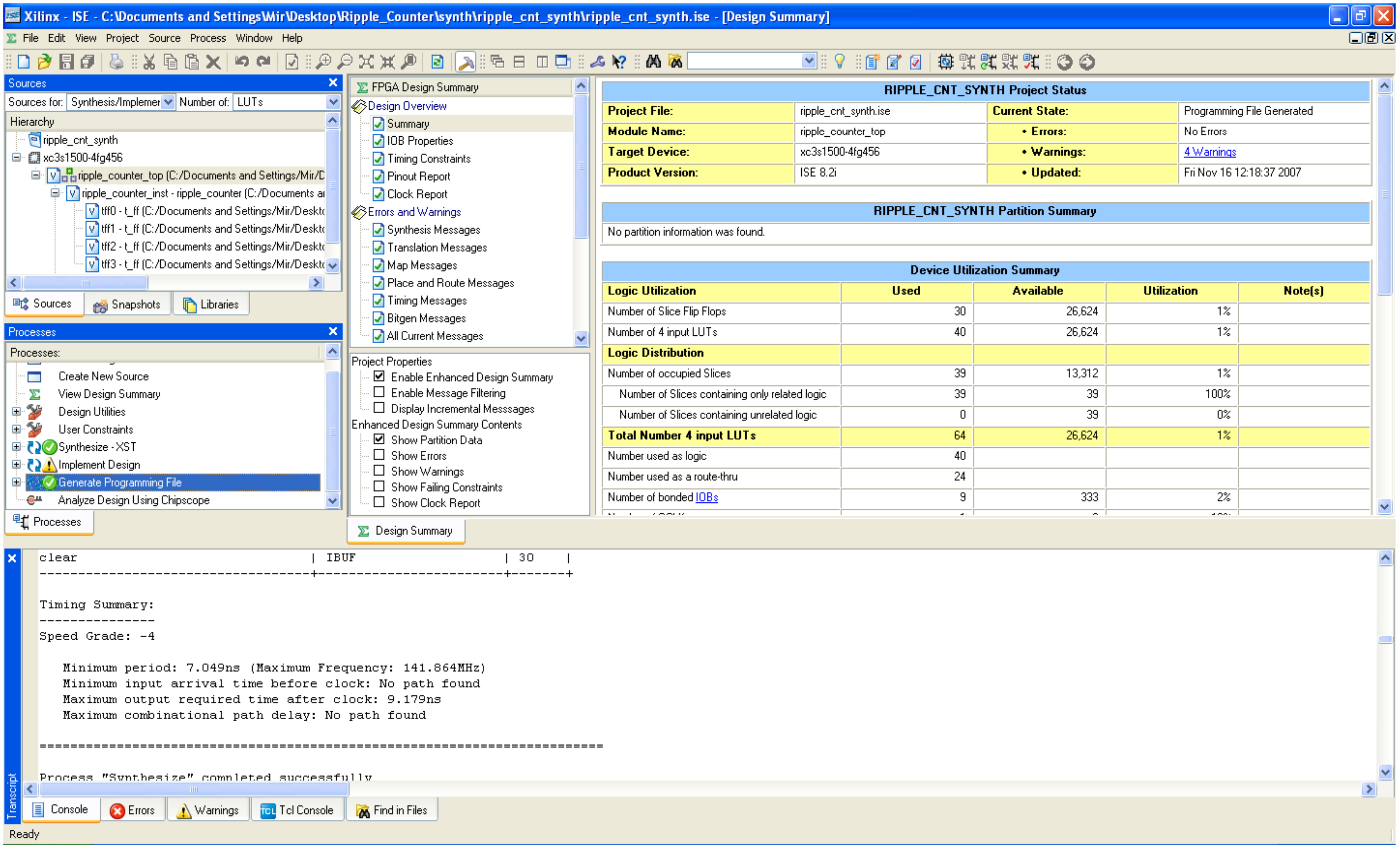
Project : Ripple_counter   Now: 3,066 ns   Delta: 0   sim:/ripple_counter_tb

ModelSim SE PLUS 5.7f
File Edit View Compile Simulate Tools Window Help
100 ns
Workspace
Project Library sim Files
source - ripple_counter_tb.v
File Edit View Tools Window
ripple_counter_tb.v
wave - default
File Edit View Insert Format Tools Window
500 1 us 1500 2 us 2500 3 us

➢ Waveform Viewer of Model*Sim – Design results*



➢ *Synthesis and Implement Design using XILINX tool*


Xilinx - ISE - C:\Documents and Settings\Mir\Desktop\Ripple_Counter\synth\ripple_cnt_synth\ripple_cnt_synth.ise - [Design Summary]

Project Properties
- [✓] Enable Enhanced Design Summary
- [ ] Enable Message Filtering
- [ ] Display Incremental Messages

Enhanced Design Summary Contents
- [✓] Show Partition Data
- [ ] Show Errors
- [ ] Show Warnings
- [ ] Show Failing Constraints
- [ ] Show Clock Report

RIPPLE_CNT_SYNTH Project Status

Project File:	ripple_cnt_synth.ise	**Current State:**	Programming File Generated
Module Name:	ripple_counter_top	• **Errors:**	No Errors
Target Device:	xc3s1500-4fg456	• **Warnings:**	4 Warnings
Product Version:	ISE 8.2i	• **Updated:**	Fri Nov 16 12:18:37 2007

RIPPLE_CNT_SYNTH Partition Summary

No partition information was found.

Device Utilization Summary

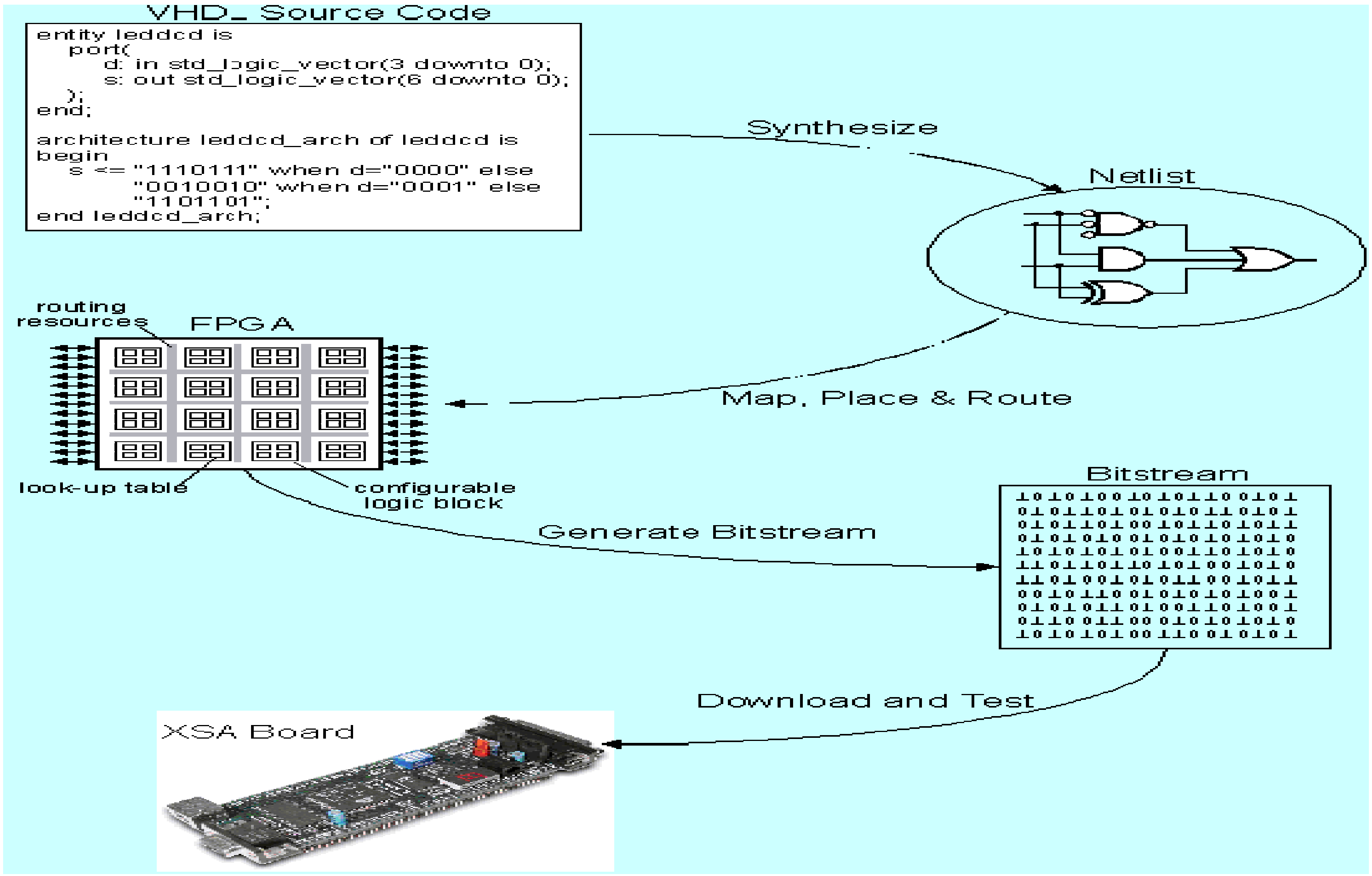
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	30	26,624	1%	
Number of 4 input LUTs	40	26,624	1%	
Logic Distribution				
Number of occupied Slices	39	13,312	1%	
Number of Slices containing only related logic	39	39	100%	
Number of Slices containing unrelated logic	0	39	0%	
Total Number 4 input LUTs	64	26,624	1%	
Number used as logic	40			
Number used as a route-thru	24			
Number of bonded IOBs	9	333	2%	

```
clear                              | IBUF                    | 30    |
-----------------------------------+------------------------+-------+

Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 7.049ns (Maximum Frequency: 141.864MHz)
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: 9.179ns
   Maximum combinational path delay: No path found

=========================================================================

Process "Synthesize" completed successfully
```

Introductory Class

FPGA based Digital Design using Verilog HDL
(f p g a c o u r s e @ y a h o o . c o m)

FPGA Based Design Flow

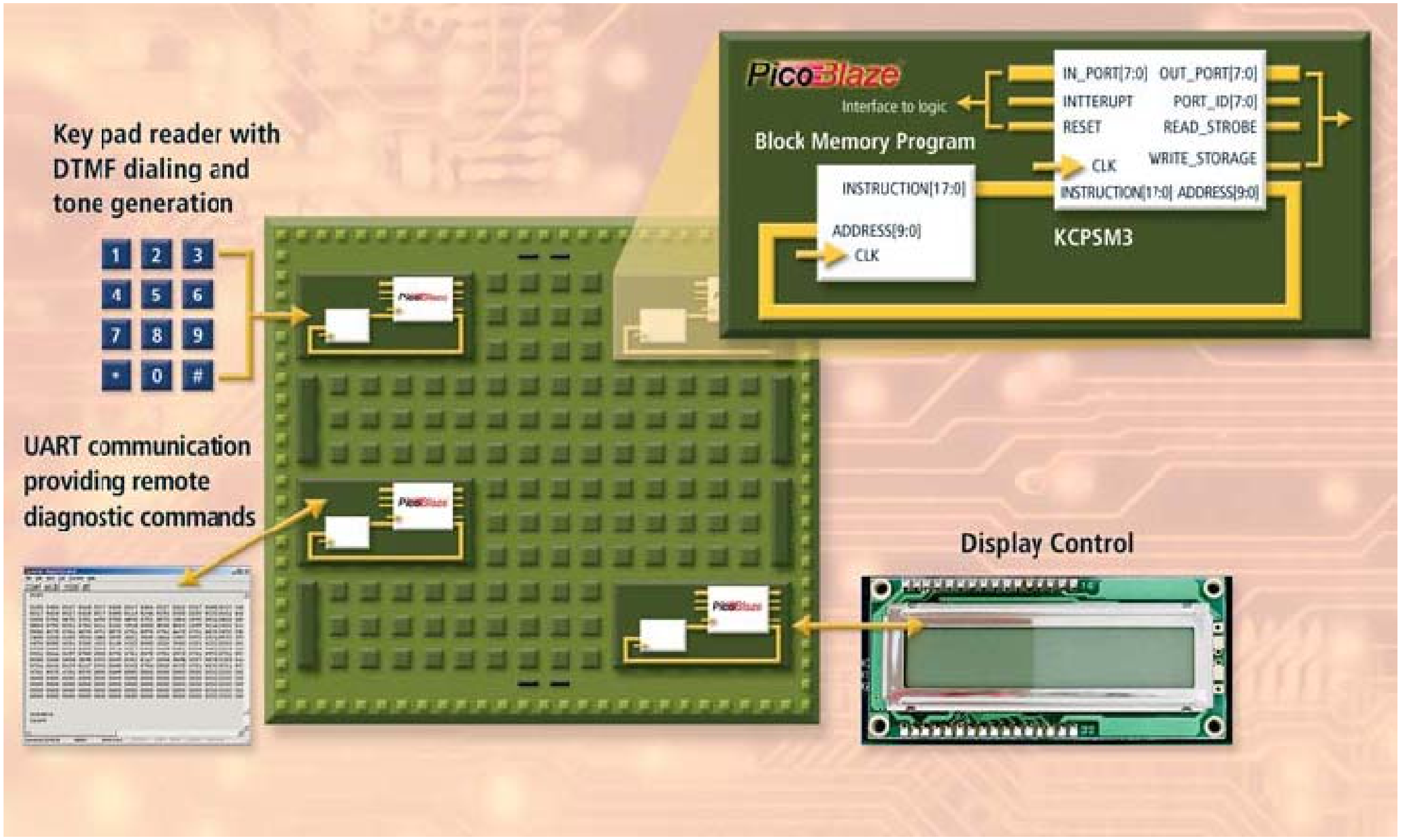


```
entity leddcd is
   port(
      d: in std_logic_vector(3 downto 0);
      s: out std_logic_vector(6 downto 0);
   );
end;

architecture leddcd_arch of leddcd is
begin
   s <= "1110111" when d="0000" else
        "0010010" when d="0001" else
        "1101101";
end leddcd_arch;
```

FPGA based Digital Design using Verilog HDL

Embedded Controllers in FPGAs

Microprocessor inside FPGA



FPGA based Digital Design using Verilog HDL

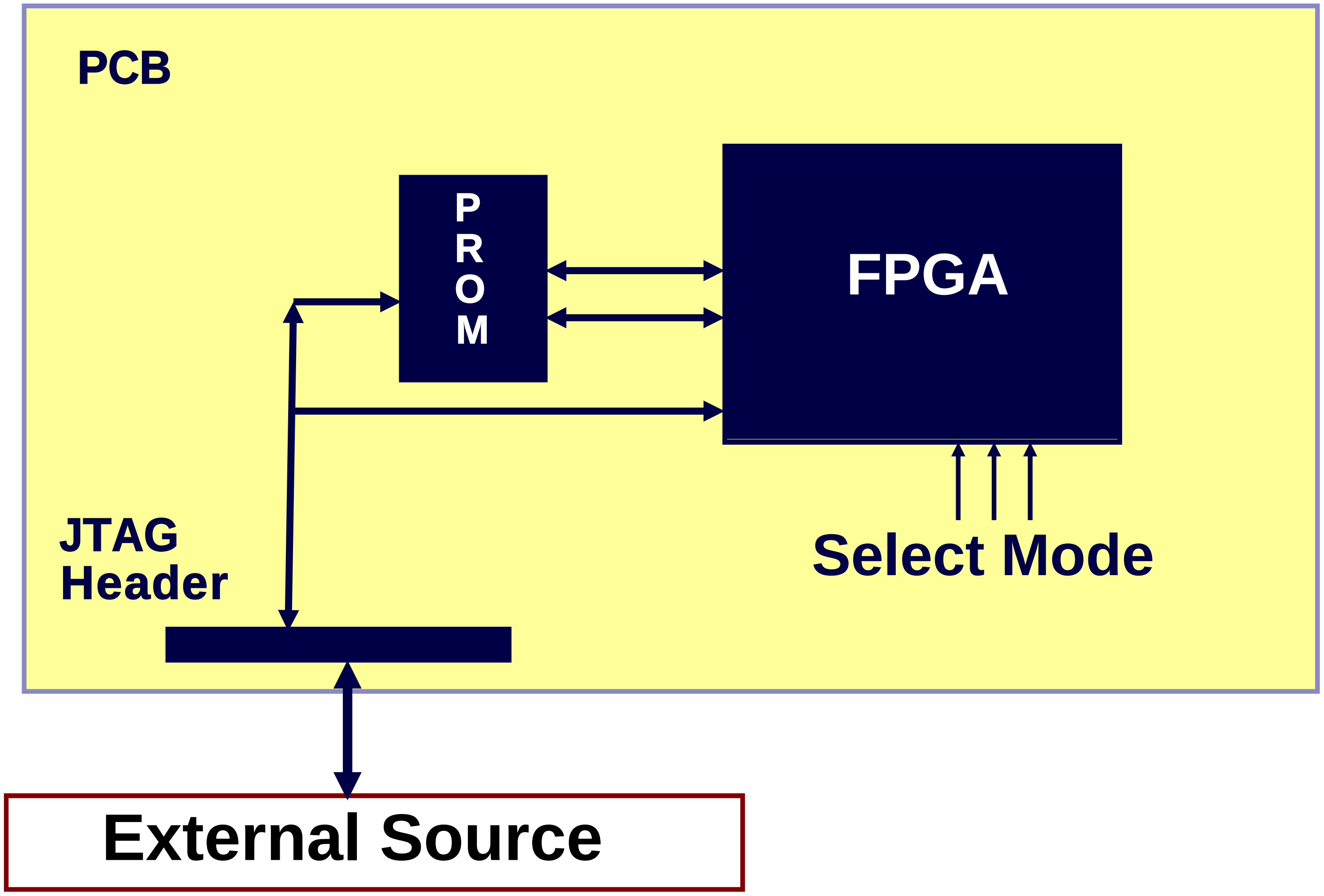
FPGA in Real World

FPGA in Real World

How to use in Real Environment

- **Xilinx FPGAs are volatile because they are based on SRAM technology.**
- **That is, the device loses its configuration if the power to the device is turned off.**
- **FPGAs typically utilize an external memory device, such as a PROM for production type environment to prevent the loss of configuration data in a power outage. Xilinx PROM are available in two different type:**
  - **→ One Time Programmable (OTP) [XC17x]**
  - **→ In System Programmable [XCFxxx/XC18x]**

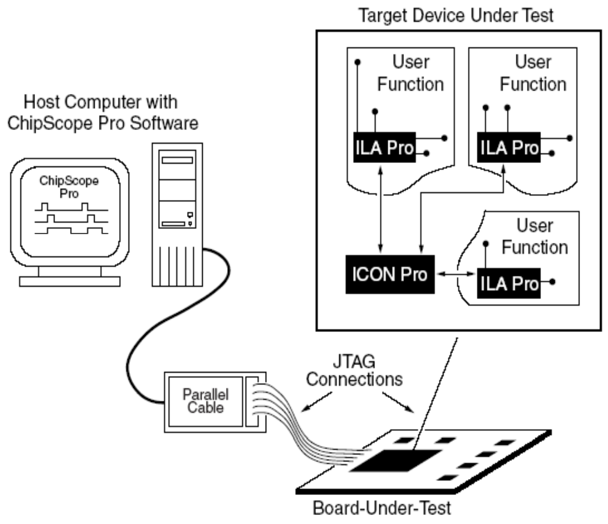
Xilinx FPGA with PROM



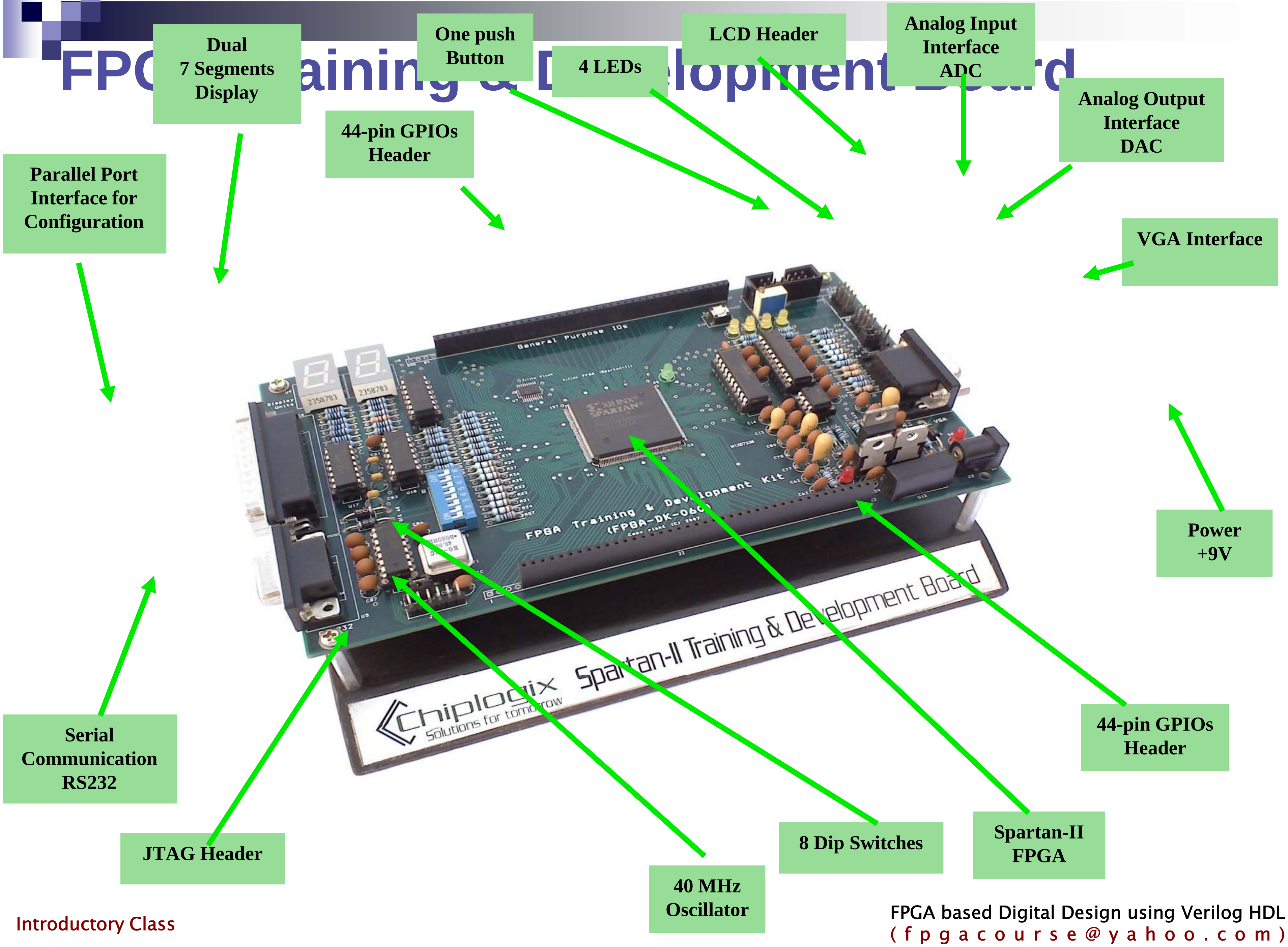
Real time Debugging for Xilinx FPGA

❑ The Solution: Xilinx ChipScope Pro tools

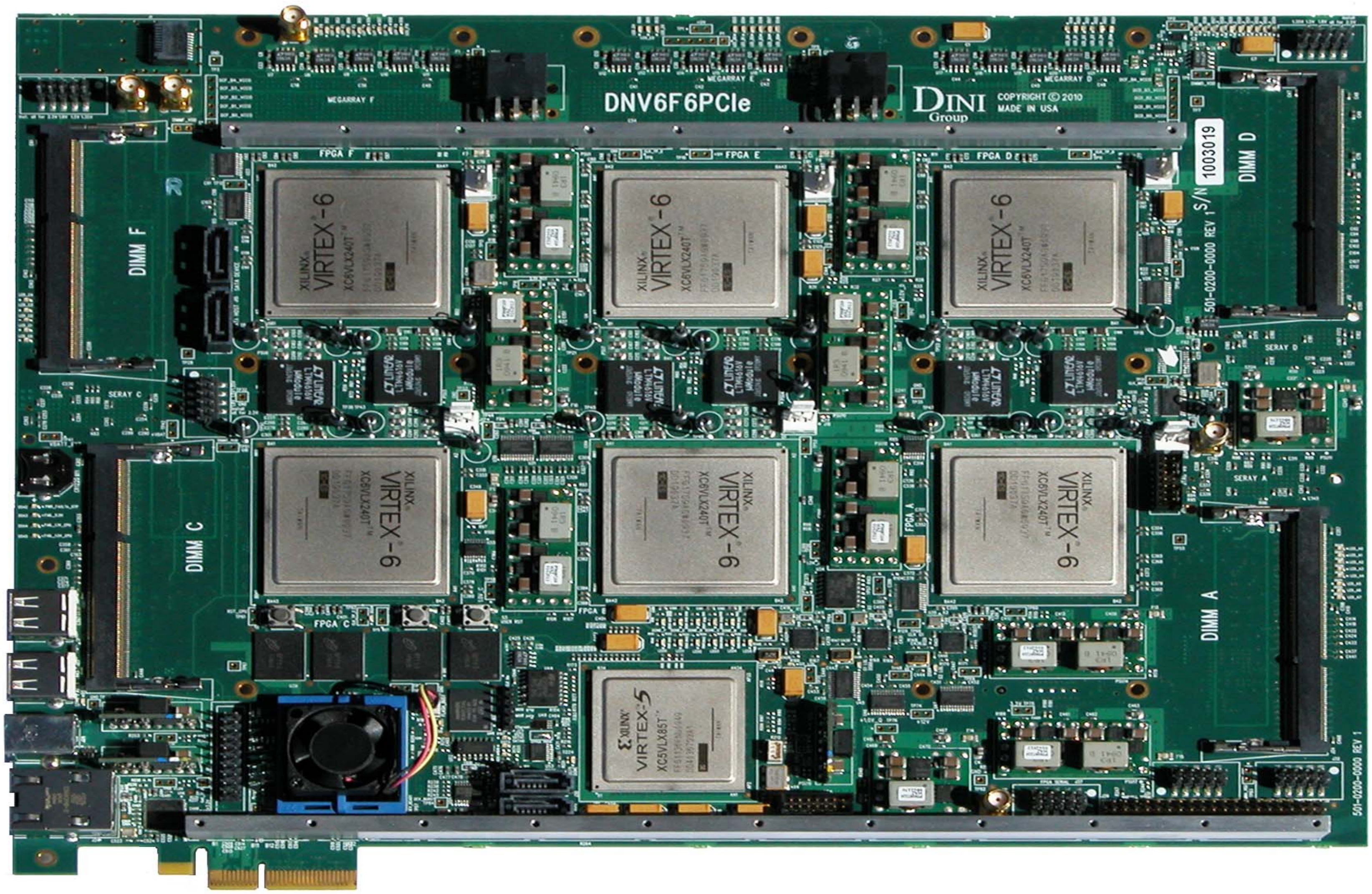
- As the density of FPGAs increase, so does the uselessness of test equipment probes to these devices under test.
- The ChipScope Pro tools integrate key logic analyzer hardware components with the target design inside Xilinx FPGAs devices.
- The ChipScope Pro tools communicate with these components and provide the designer with a complete logic analyzer.

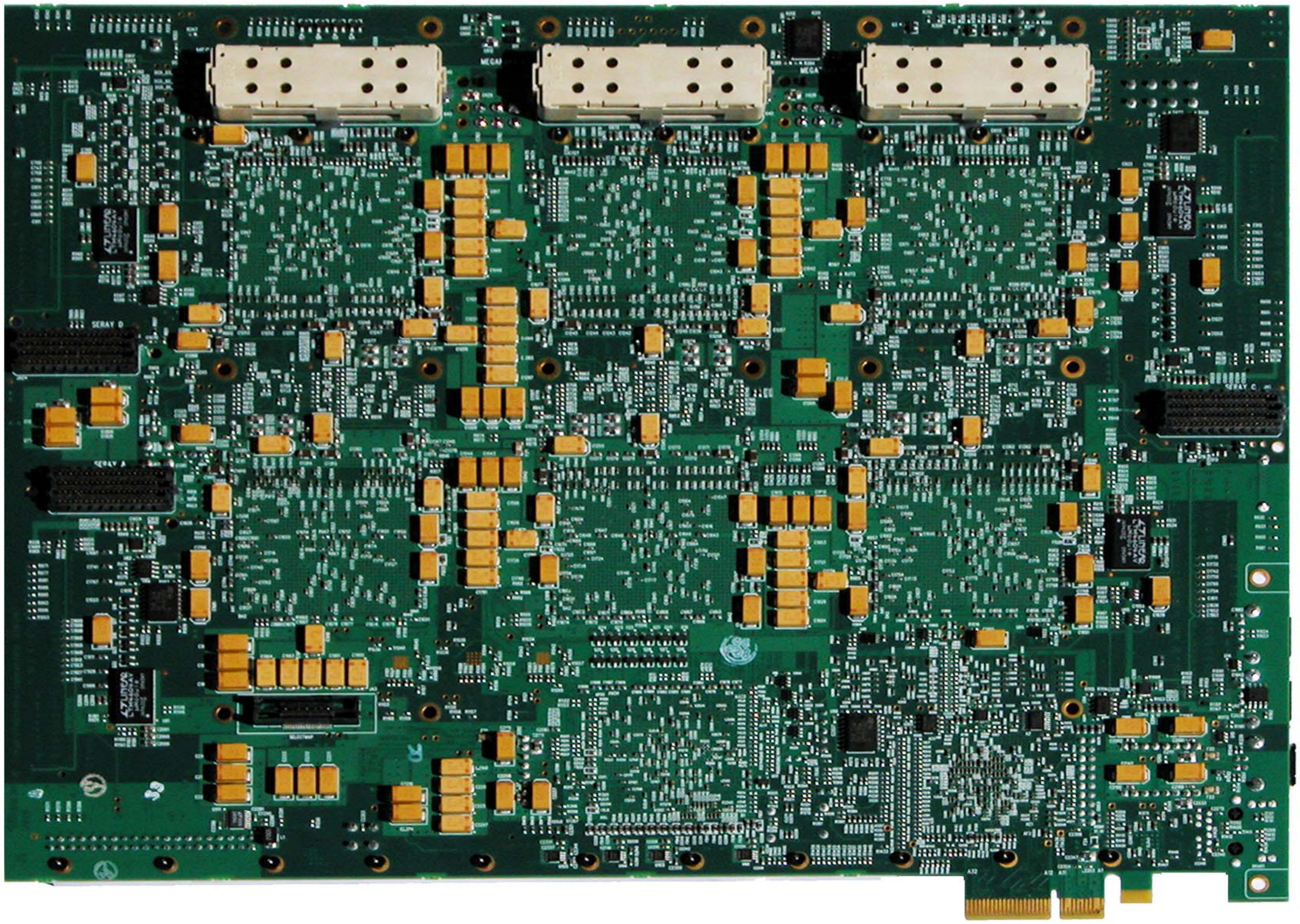


FPGA Training & Development Board


Dual 7 Segments Display
One push Button
4 LEDs
LCD Header
Analog Input Interface ADC
Analog Output Interface DAC
44-pin GPIOs Header
Parallel Port Interface for Configuration
VGA Interface
Power +9V
Serial Communication RS232
JTAG Header
40 MHz Oscillator
8 Dip Switches
Spartan-II FPGA
44-pin GPIOs Header
Chiplogix Solutions for tomorrow
Spartan-II Training & Development Board
FPGA Training & Development Kit (FPGA-DK-060)
General Purpose IOs
XILINX SPARTAN

DINI Group:


DNV6F6PCIe
DINI Group
COPYRIGHT © 2010 MADE IN USA
FPGA F
FPGA E
FPGA D
FPGA C
DIMM F
DIMM C
DIMM D
DIMM A
XILINX VIRTEX-6 XC6VLX240T
XILINX VIRTEX-5 XC5VLX85T
MEGARRAY F
MEGARRAY E
MEGARRAY D
S/N 1003019
501-0200-0000 REV 1

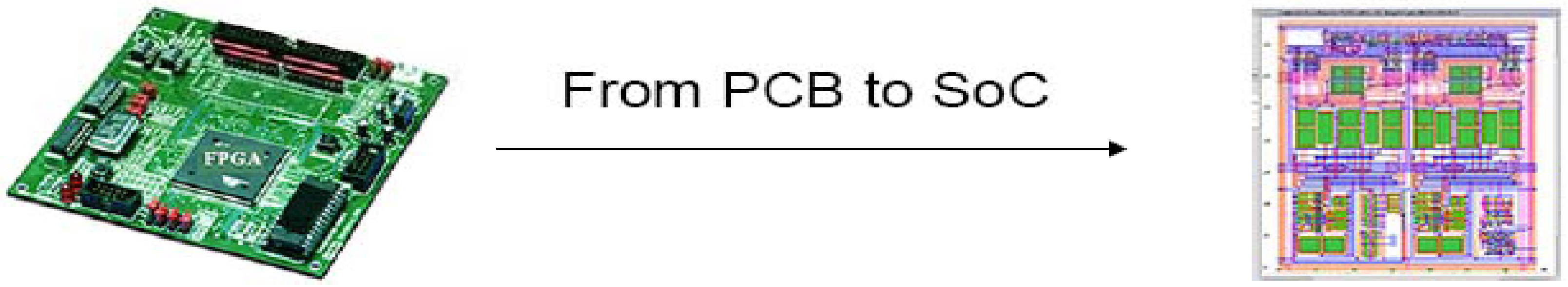


FPGA based Digital Design using Verilog HDL

Motivation

Motivation

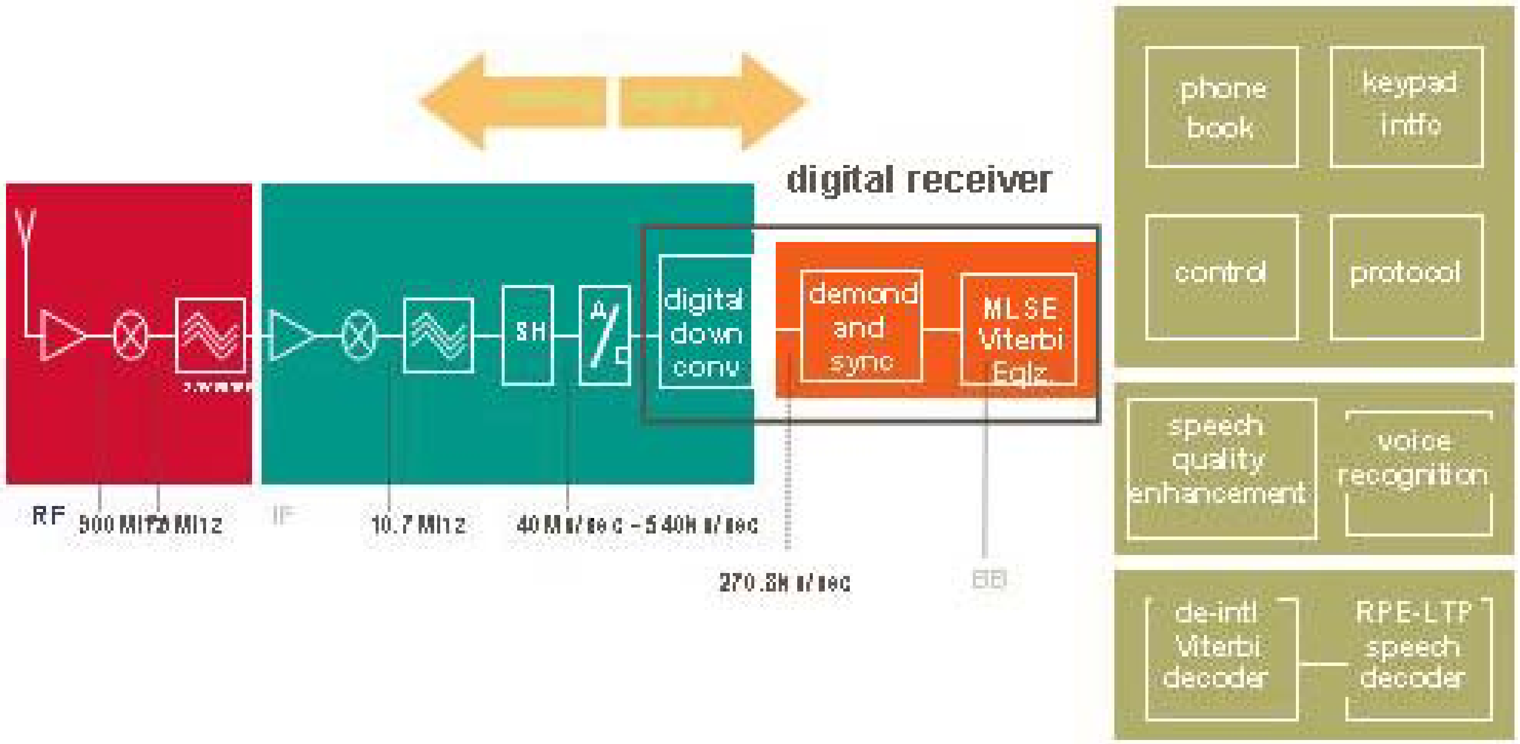
- **System On Chip (SoC)**
  - **SoC means you put the entire system on a single chip**
  - **Your Analog and Digital Design on a single chip**
  - **Example of any complex design that have Micro controller, Memories, Analog portion, DSP processor and algorithm on FPGA**
  - **Concept is that put all above in a single chip**



- New market of business in electronics
- For SoC you need the Core of every component i-e IP (Intellectual Property) of these core
- You can build your own IPs…A group of two or three engineers

FPGA based DSP Systems

❑ System Level Modeling: GSM System



Major SoC Applications

- Speech Signal Processing .
- Image and Video Signal Processing .
- Information Technologies
  - PC interface (USB, PCI,PCI-Express, IDE,..etc) Computer peripheries (printer control, LCD monitor controller, DVD controller,.etc) .
- Data Communication
  - Wireline Communication: 10/100 Based-T, xDSL, Gigabit Ethernet,.. Etc
  - Wireless communication: BlueTooth, WLAN, 2G/3G/4G, WiMax, UWB, …,etc

FPGA based Digital Design using Verilog HDL

Course Strategy

Course Outlines

- Digital Design Methodology
- Digital Chip Design using Verilog HDL with *ModelSim* Simulator
- Introduction to VHDL and comparison with Verilog HDL
- Controller based Designs – State Machines
- Embedded Controllers – Implementation of MicroBlaze
- Testing and Verification Methodology – Automated Test Benches
- Xilinx FPGA/CPLD architecture – Spartan-II & Spartan-III devices
- Xilinx FPGA Design Methodology
- Design synthesis and implementation using Xilinx *ISE 9.1i*
- Digital Circuits Designed with built-in resources for specific FPGA
- Timing Analyzer, Core Generator, Constraint Editor, Floor Planner and iMPACT tools
- Introduction to System-on-Chip (SoC)
- FPGA configuration modes and in-depth study of FPGA based PCB
- Real time on-chip debugging for Xilinx FPGA by using ChipScope Pro Tools
- Embedded Development Kit Overview (EDK 8.2i)
- Real time simulations with Xilinx FPGA based Training Boards
- Case Study : ***Study to build a Data Acquisitioning System FPGA

- **Course Duration:**
  Four Weeks (3 days per week)

- **Total Lectures:**
  Sixteen (18) (**Time:** ≈ 2 hrs per lecture)

- **Total LABS:**
  Twenty (18) (**Time:** ≈ 2 hrs per lab)

Group URL: www.groups.yahoo.com/group/chip_designing

Course Strategy

- **Information**
  - **Instructor**
    - **Nauman Mir**

      email : naumanmir@ChipLogix.com
      cell : +92 303 553 9956

  - **Prerequisite**
    - **Digital Logic Design**
    - **Computer Architecture Concept**
    - **Programming Basics (C language)**

Skills Gained..

- After completing this comprehensive training, you will have the necessary skills to:
  - Locate the design issues and solve them..
  - Implement digital designs using Verilog HDL
  - Create test and verification strategy for chip designing
  - Design an optimize state machine based designs
  - Xilinx FPGA architecture details
  - Synthesis, Implementation & configuration processes using ISE 9.1
  - Create design constraints and analyze synthesis & timing reports
  - Use built-in resources of Xilinx FPGA in your design
  - How to diagnose your Design Critical Path by using PlanAhead
  - Use Xilinx Chipscope Pro tools for real-time debugging (ILA)
  - Use Xilinx Chipscope Pro tools for real-time debugging (VIO)
  - Create Real-time testing environment with FPGA boards

Thanks