浙江大学
ZHEJIANG UNIVERSITY

Computer Architecture Experiment

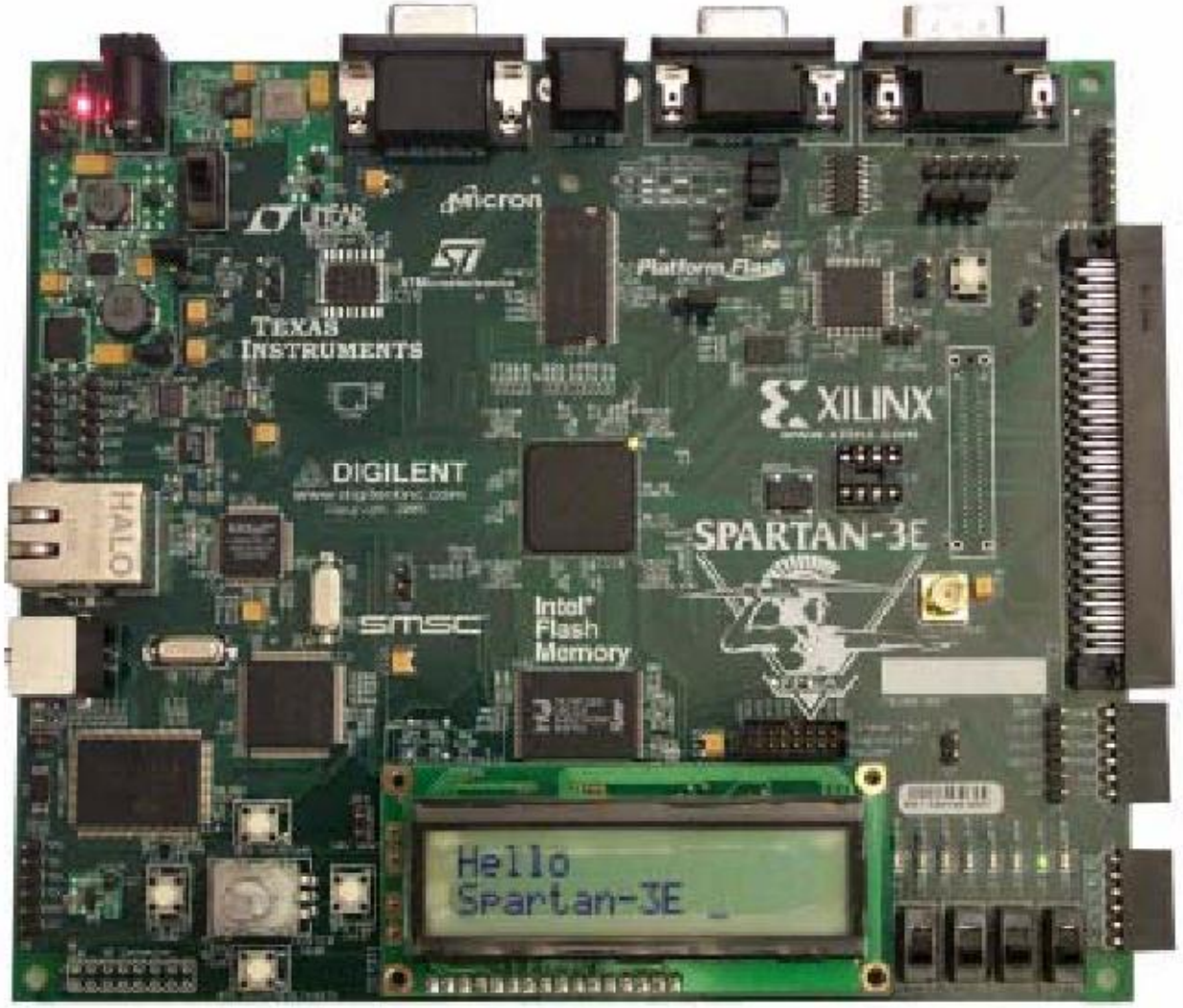
Topic 0. Basic Knowledge

浙江大学计算机学院
陈文智、王总辉
{chenwz, zhwang}@zju.edu.cn

Outline

- Xilinx Spartan-3E Start Kit Board
- Verilog HDL
- Xilinx ISE 10.1 i
- Report format

Xilinx Spartan-3E Start kit Board（1）



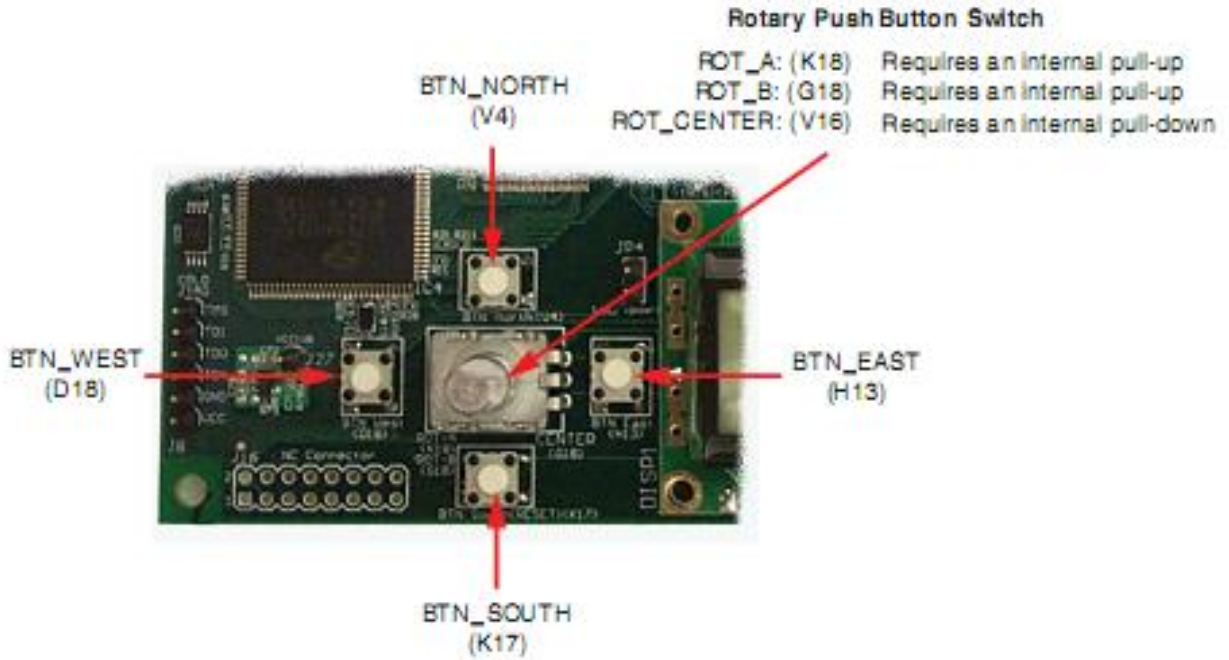
Key Features

- Xilinx XC3S500E Spartan-3E FPGA
- 2-line, 16-character LCD screen
- Eight discrete LEDs
- Four slide switches
- Four push-button switches
- 50 MHz clock oscillator
- USB Cable for programming

Four slide switches

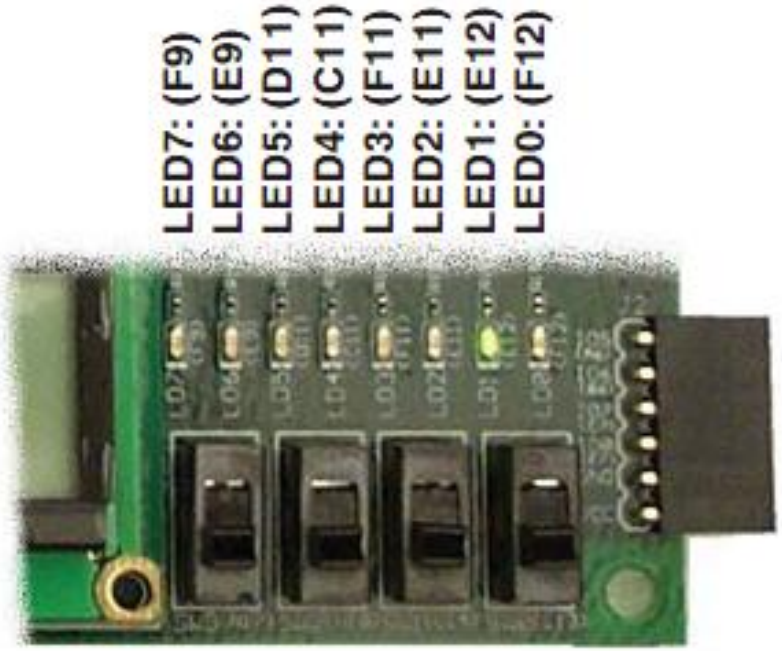

```
NET "SW<0>" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
NET "SW<1>" LOC = "L14" | IOSTANDARD = LVTTL | PULLUP ;
NET "SW<2>" LOC = "H18" | IOSTANDARD = LVTTL | PULLUP ;
NET "SW<3>" LOC = "N17" | IOSTANDARD = LVTTL | PULLUP ;
```

Four push-button switches




```
NET "BTN_EAST"  LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "BTN_NORTH" LOC = "V4"  | IOSTANDARD = LVTTL | PULLDOWN ;
NET "BTN_SOUTH" LOC = "K17" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "BTN_WEST"  LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN ;
```

Eight discrete LEDs




```
NET "LED<7>" LOC = "F9"  | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<6>" LOC = "E9"  | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<5>" LOC = "D11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<4>" LOC = "C11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<1>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
```

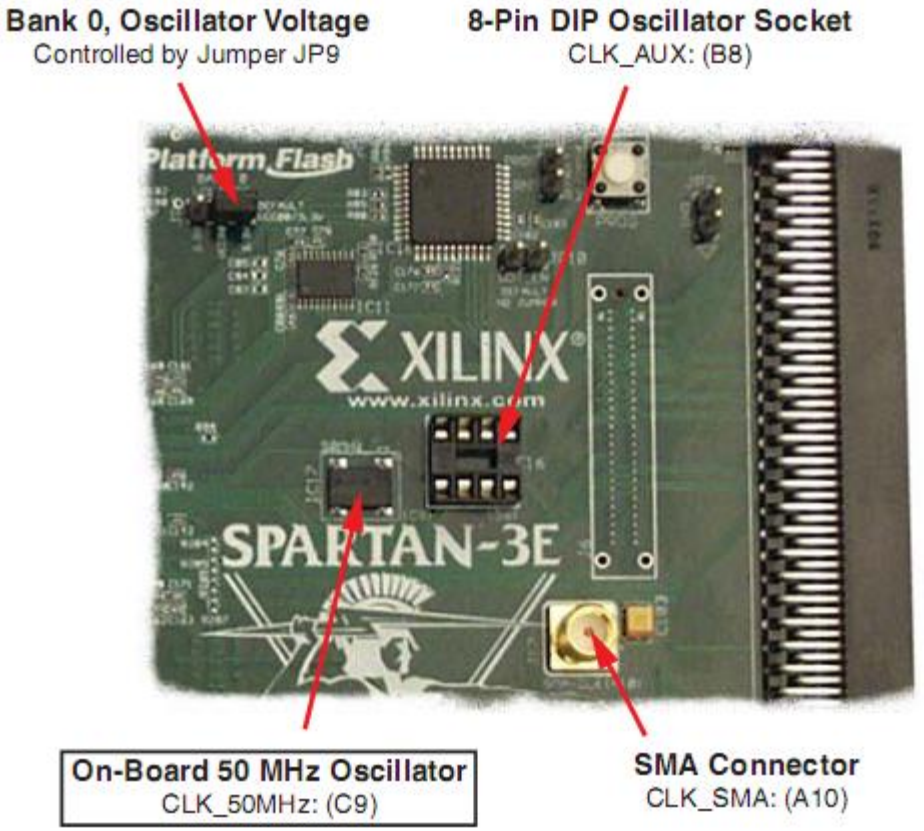
50MHz clock oscillator

板上支持3个主时钟输入源：

1）包括一个50MHz的时钟晶振

2）通过SMA连接器，时钟可以板外供应。反之，FPGA也可以提供时钟信号或其它高速信号给SMA连接器

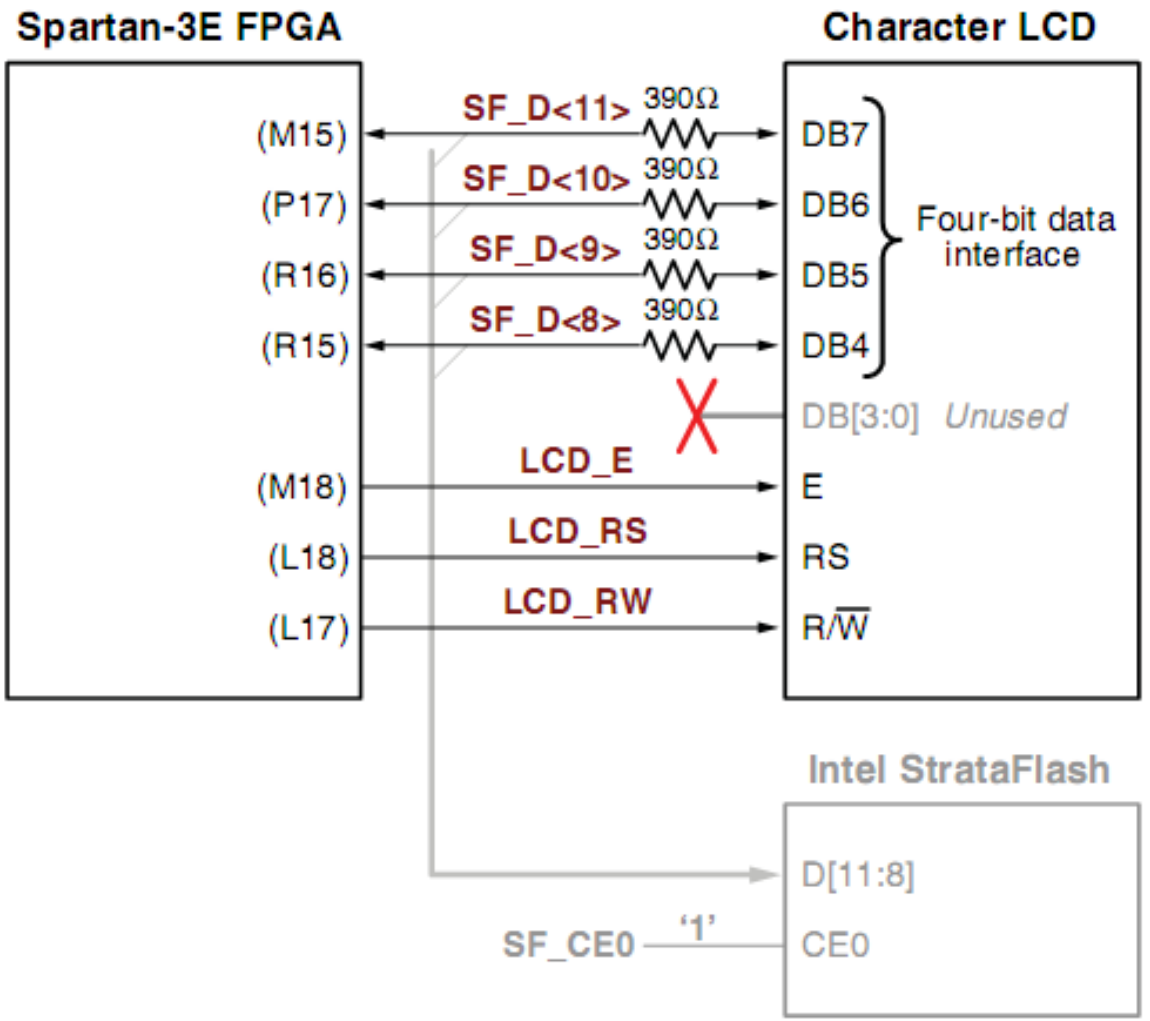
3）分列式8-DIP时钟晶振插槽




```
NET "CLK_50MHZ" LOC = "C9"  | IOSTANDARD = LVCMOS33 ;
NET "CLK_SMA"   LOC = "A10" | IOSTANDARD = LVCMOS33 ;
NET "CLK_AUX"   LOC = "B8"  | IOSTANDARD = LVCMOS33 ;
```

Character LCD Interface

4 根 LCD 数据线与 StrataFlash 数据线 SF_D<11:8> 复用。LCD/StrataFlash 存储器依赖于设计而交互使用。当存储器失能时（SF_CE0=1），FPGA用作全读/写通道给LCD。相反，当 LCD 读失能时（LCD_RW=0），FPGA用作全读/写通道给存储器。



Character LCD Interface Signal

Signal Name	FPGA Pin	Function	
SF_D<11>	M15	Data bit DB7	Shared with StrataFlash pins SF_D<11:8>
SF_D<10>	P17	Data bit DB6	
SF_D<9>	R16	Data bit DB5	
SF_D<8>	R15	Data bit DB4	
LCD_E	M18	Read/Write Enable Pulse 0: Disabled 1: Read/Write operation enabled	
LCD_RS	L18	Register Select 0: Instruction register during write operations. Busy Flash during read operations 1: Data for read or write operations	
LCD_RW	L17	Read/Write Control 0: WRITE, LCD accepts data 1: READ, LCD presents data	

```
NET "LCD_E"   LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RS"  LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RW"  LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

# The LCD four-bit data interface is shared with the StrataFlash.
NET "SF_D<8>"  LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<9>"  LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<11>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```

LCD Controller

- Graphics controller（Sitronix ST7066U）
  - Samsung S6A0069X or KS0066U
  - Hitachi HD44780
  - SMOS SED1278

- **DD RAM (Display Data RAM)**

	Character Display Addresses																Undisplayed Addresses		
1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	…	27
2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	…	67
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	…	40

LCD Character Set

- **CG ROM**

(Character Generator ROM)

浙江大學 ZheJiang University

Custom Checkerboard Character

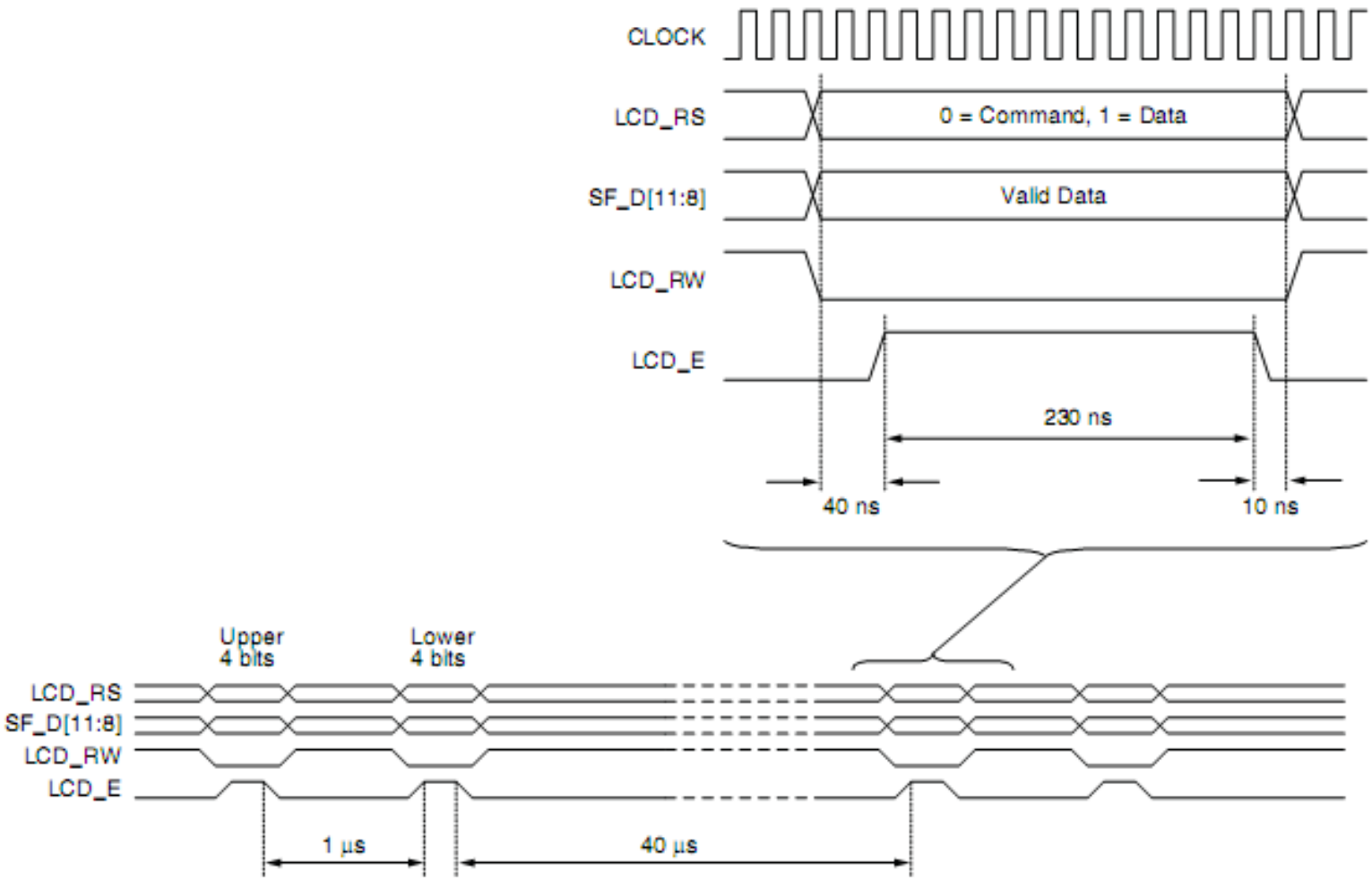
- **CG RAM(Character Generator RAM)**

						Upper Nibble				Lower Nibble			
						Write Data to CG RAM or DD RAM							
A5	A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
Character Address			Row Address			Don't Care			Character Bitmap				
0	1	1	0	0	0	-	-	-	0	■	0	■	0
0	1	1	0	0	1	-	-	-	■	0	■	0	■
0	1	1	0	1	0	-	-	-	0	■	0	■	0
0	1	1	0	1	1	-	-	-	■	0	■	0	■
0	1	1	1	0	0	-	-	-	0	■	0	■	0
0	1	1	1	0	1	-	-	-	■	0	■	0	■
0	1	1	1	1	0	-	-	-	0	■	0	■	0
0	1	1	1	1	1	-	-	-	0	0	0	0	0

LCD Character Display Command Set

Function	LCD_RS	LCD_RW	Upper Nibble				Lower Nibble			
			DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear Display	0	0	0	0	0	0	0	0	0	1
Return Cursor Home	0	0	0	0	0	0	0	0	1	-
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S
Display On/Off	0	0	0	0	0	0	1	D	C	B
Cursor and Display Shift	0	0	0	0	0	1	S/C	R/L	-	-
Function Set	0	0	0	0	1	0	1	0	-	-
Set CG RAM Address	0	0	0	1	A5	A4	A3	A2	A1	A0
Set DD RAM Address	0	0	1	A6	A5	A4	A3	A2	A1	A0
Read Busy Flag and Address	0	1	BF	A6	A5	A4	A3	A2	A1	A0
Write Data to CG RAM or DD RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0
Read Data from CG RAM or DD RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0

LCD Interface Operation Timing



Operation

- **Initializing the Display**
  - Power-On Initialization
  - Display Configuration
- **Writing Data to the Display**
- **Disabling the Unused LCD**

Verilog HDL

- Logic Value
- Bit-widths
- Continuous Assignment Statement
- Procedural Assignment Statement
- Operators
- If & Case Statements
- Loop Statements
- Module

Logic Value

0 <--> logic-0

1 <--> logic-1

z <--> high-impedance

z <--> don't-care (in casex and casez statements)

x <--> don't-care

x <--> unknown

Bit-widths

- **Data Type**
  - Net data type
  - Register data type
- **Constants**
  - Simple decimal
  - Base format
- **Parameter**

Continuous Assignment Statement

- **A continuous assignment statement represents logic that is derived from the expression on the right-side-hand of the assignment statement driving the net that appears on the left-side-hand of the assignment .**
- **The target of a continuous assignment is always a net driven by combinational logic.**
- **Assign a = b & c;**

Procedural Assignment Statement

- **Blocking Procedural Assignment**
  - =
- **Non-blocking Procedural Assignment**
  - <=
- **Assignment Restrictions**
  - A target cannot be assigned using a blocking assignment and a non-blocking assignment

Operators

- Logical Operators： &&、||、 !
- Arithmetic Operators： +、 -、 *、 /、 %、 mod
- Relational Operators： >、 <、 >=、 <=
- Equality Operators： ==、 !=、 ===、 !==
- Shift Operators： >>(L)、 >>>(A)、 <<(L)、 <<<(A)
- Bit Operators： &、 ~&、 ^、 ~^、 |、 ~|
- Conditional Operators： ?:

Operator Precedence

	Op	Meaning
Highest	~	NOT
	*, /, %	MUL, DIV, MOD
	+, –	PLUS, MINUS
	<<, >>	Logical Left/Right Shift
	<<<, >>>	Arithmetic Left/Right Shift
	<, <=, >, >=	Relative Comparison
	==, !=	Equality Comparison
	&, ~&	AND, NAND
	^, ~^	XOR, XNOR
	|, ~|	OR, NOR
Lowest	?:	Conditional

Statements (if & case)

- **If statement**
- **Case statement**
  - casez、casex

```
casez (ProgramCounter)
  4'b???1 : DoCommand = 0;
  4'b??10 : DoCommand = 1;
  4'b?100 : DoCommand = 2;
  4'b1000 : DoCommand = 3;
  default : DoCommand = 0;
endcase
```

```
casex (Select)
  5'bxxxx1 : BitPosition = 1;
  5'bxxx1x : BitPosition = 2;
  5'bxx1xx : BitPosition = 3;
  5'bx1xxx : BitPosition = 4;
  5'b1xxxx : BitPosition = 5;
  default : BitPosition = 0;
endcase
```

```
if (ProgramCounter [3])
  DoCommand = 0;
else if (ProgramCounter [2:3] == 2'b10)
  DoCommand = 1;
else if (ProgramCounter [1:3] == 3'b100)
  DoCommand = 2;
else if (ProgramCounter [0:3] == 4'b1000)
  DoCommand = 3;
else
  DoCommand = 0;
```

```
if (Select [1])
  BitPosition = 1;
else if (Select [2])
  BitPosition = 2;
else if (Select [3])
  BitPosition = 3;
else if (Select [4])
  BitPosition = 4;
else if (Select [5])
  BitPosition = 5;
else
  BitPosition = 0;
```

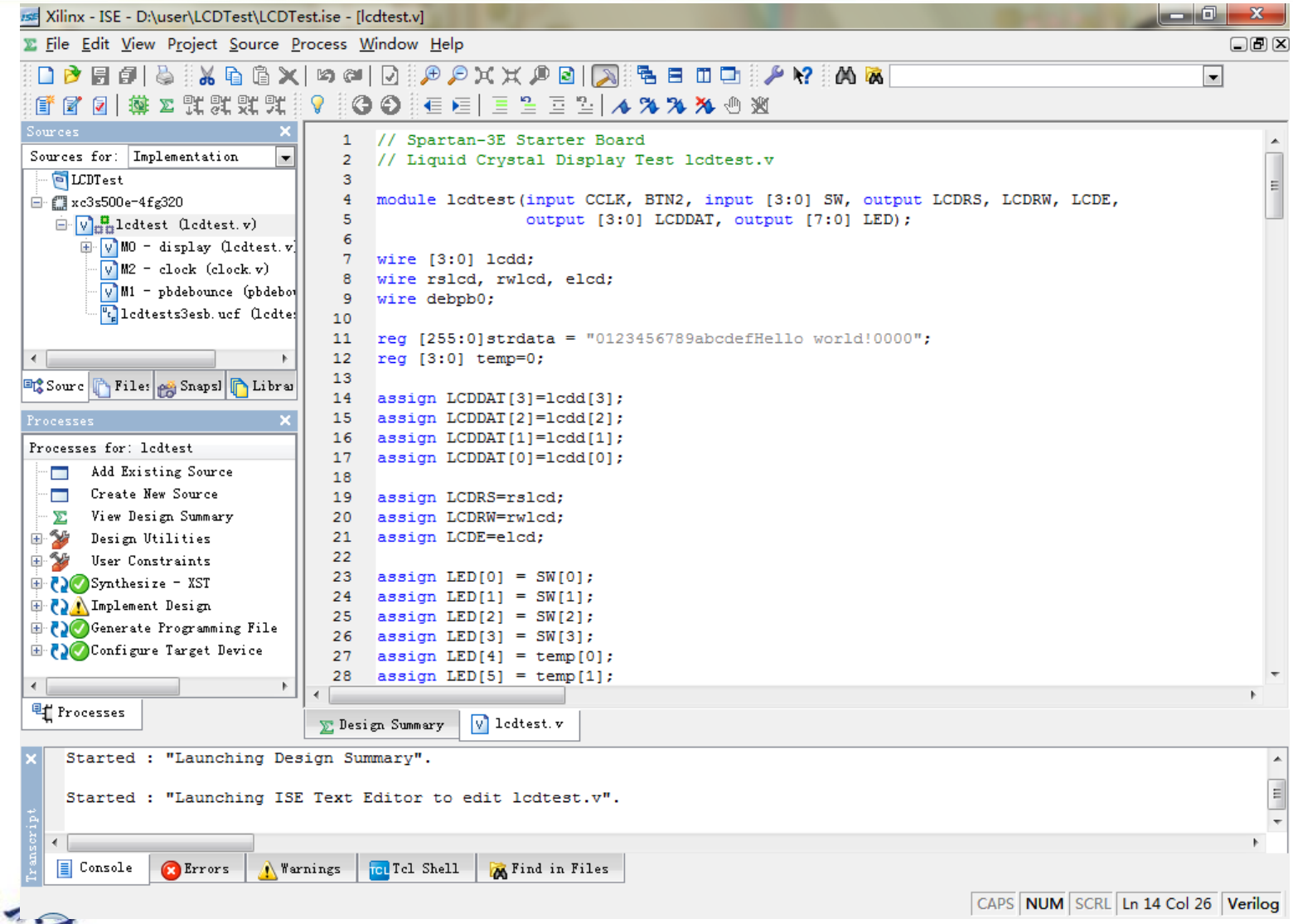
Loop Statement

- **While loop**
- **For loop**
- **Forever loop**
- **Repeat loop**

Module

- module module_name(port1, port2, …)
  - input, output, inout
  - parameter
  - wire, reg
  - Initial statement
  - Always statement
  - Module instantiation
  - Gate instantiation
  - function, task
  - UDP instantiation
  - Continuous assignment
- endmodule

Xilinx ISE 10.1 i

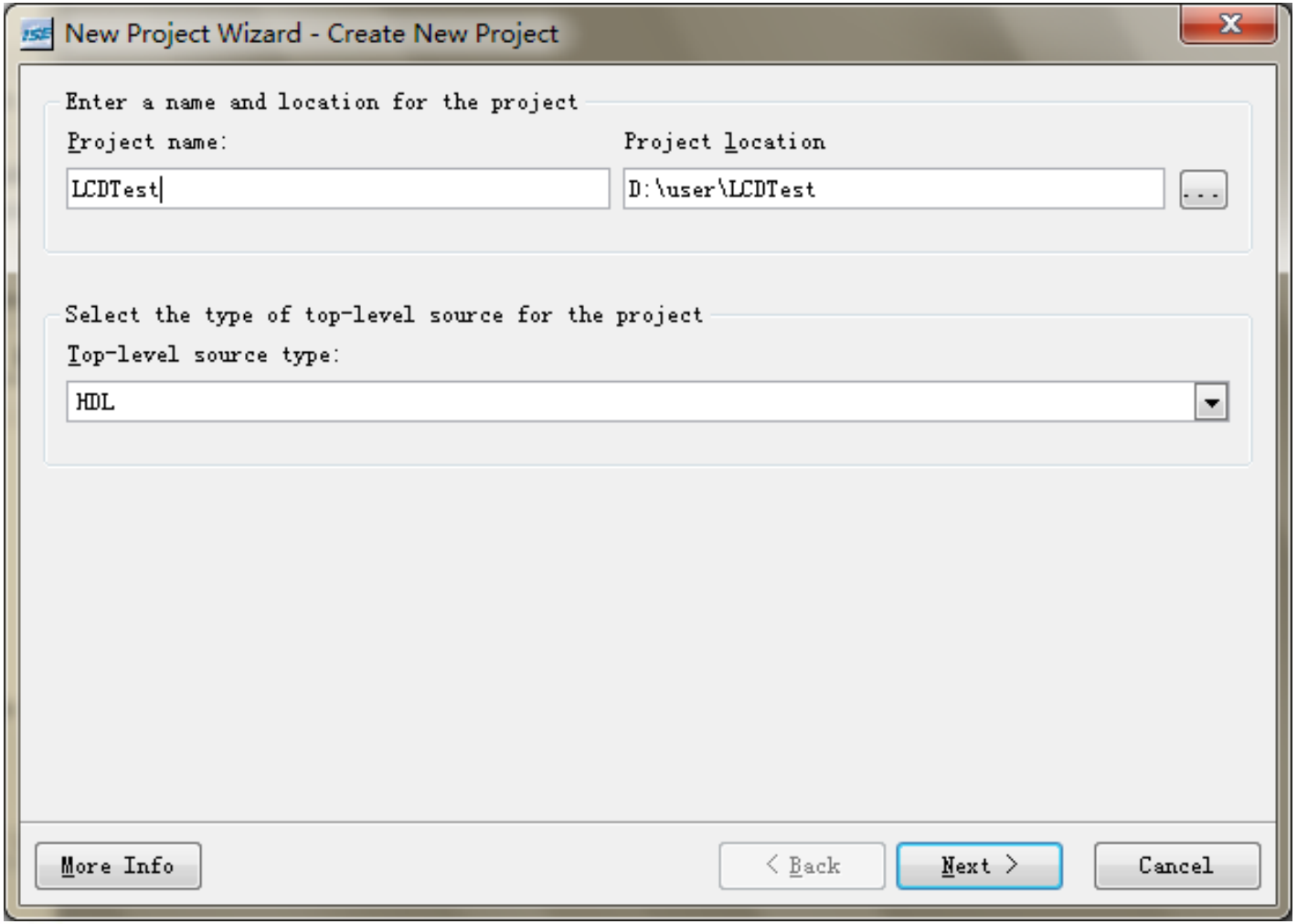


ZheJiang University

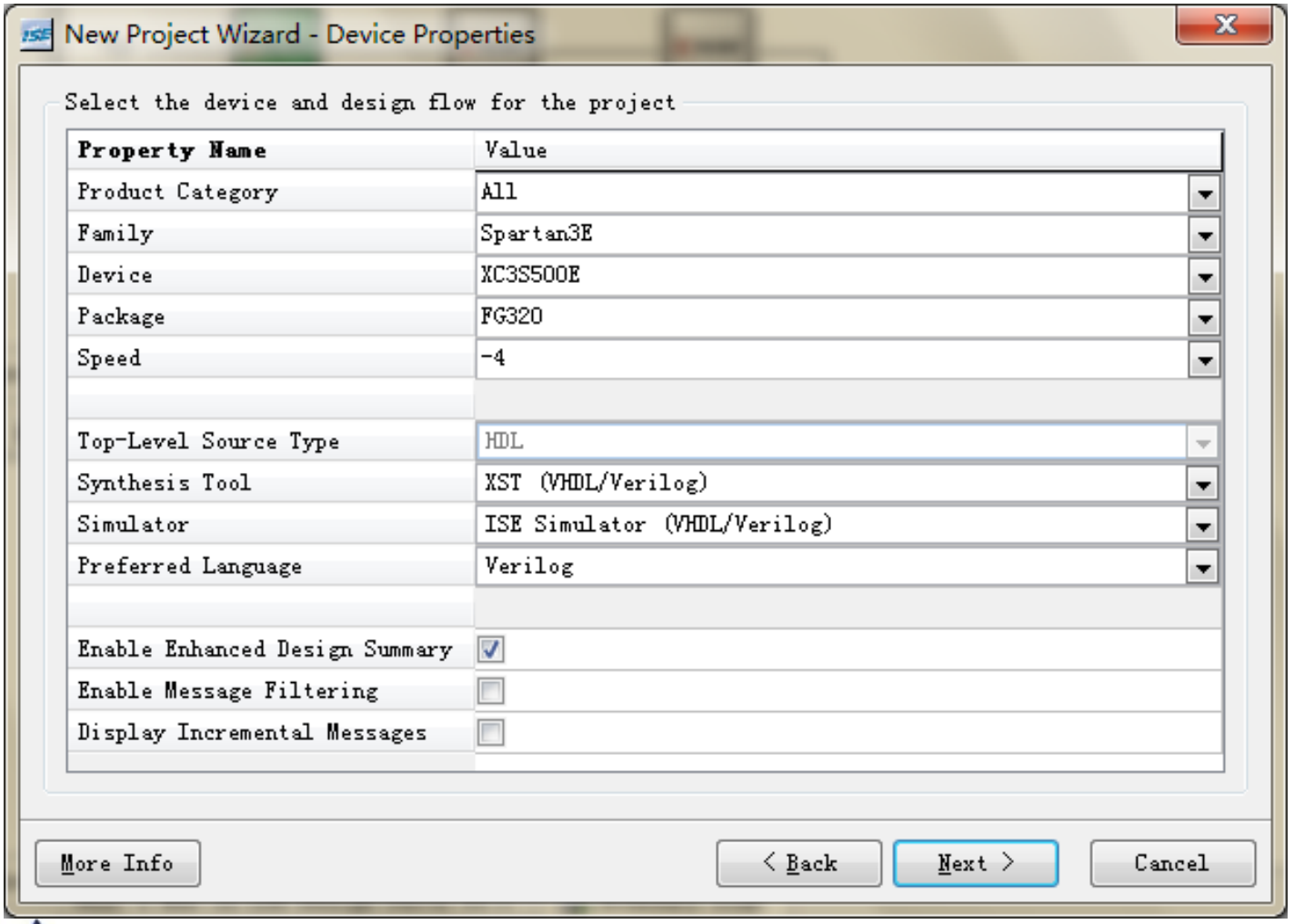
Example

- **Design a Counter:**
  - **The first line of LCD displays “0123456789abcdef”.**
  - **The second line of LCD displays “Hello World!000x”, x is the value of the counter.**
  - **Push “BTN_SOUTH” button, the value of the counter is increased by 1 from 0 to f.**
  - **The four slide-buttons is associated with the 0-3 LEDs, the value of the counter displays on 4-7 LEDs.**
  - **Modulized display part**

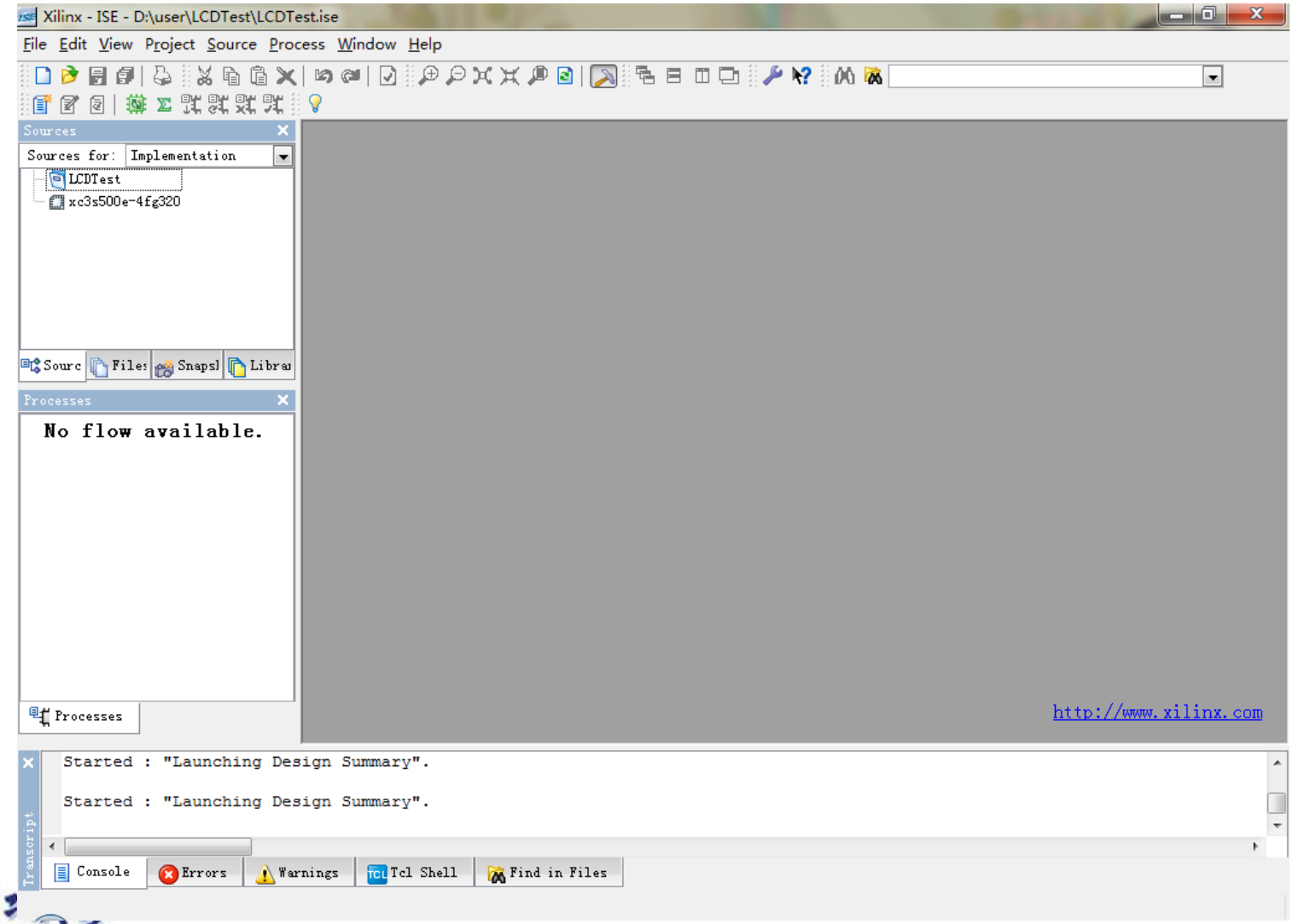
Step 1: New Project



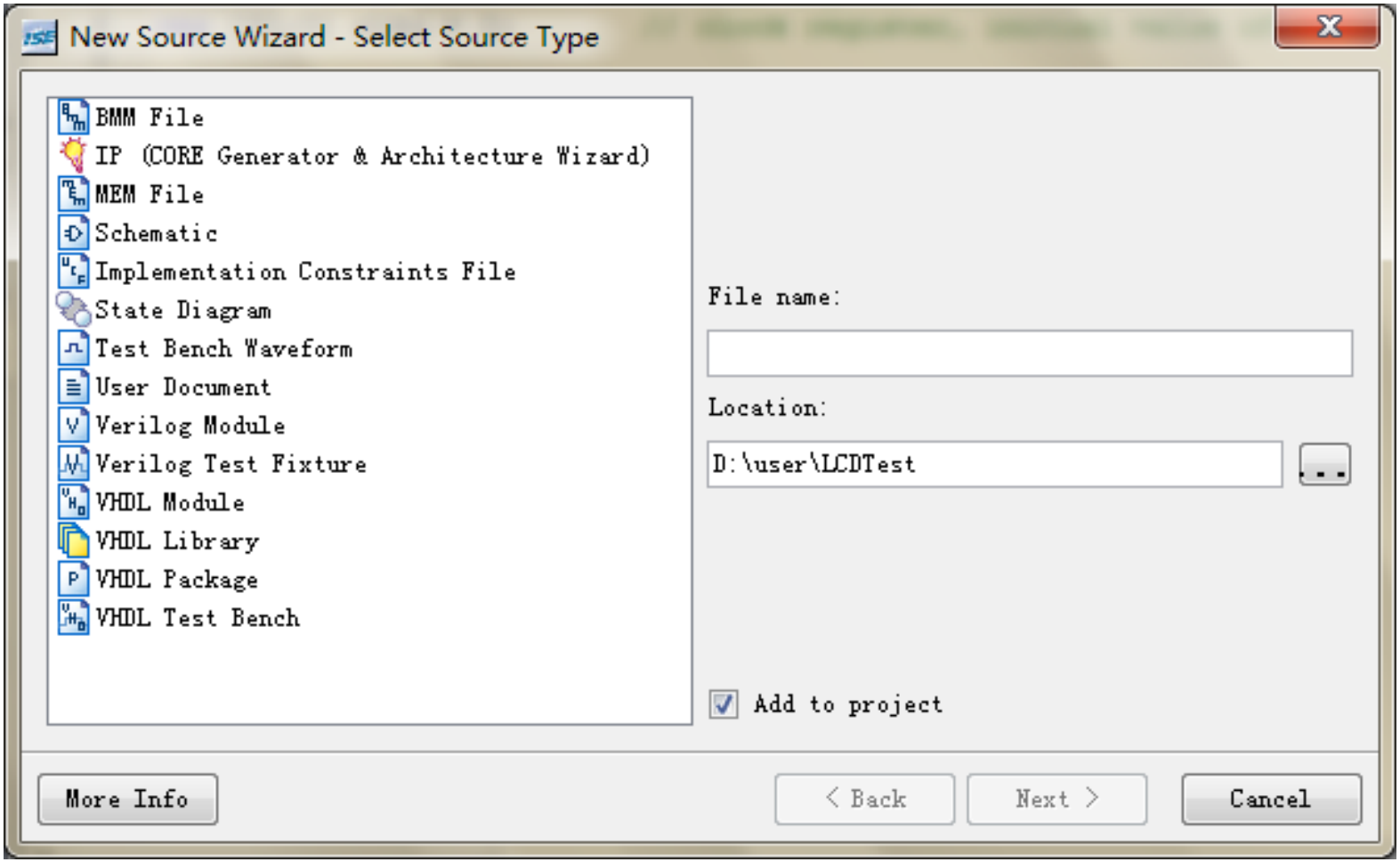
Step 2: Device Properties



Step 3: Create Project Completion



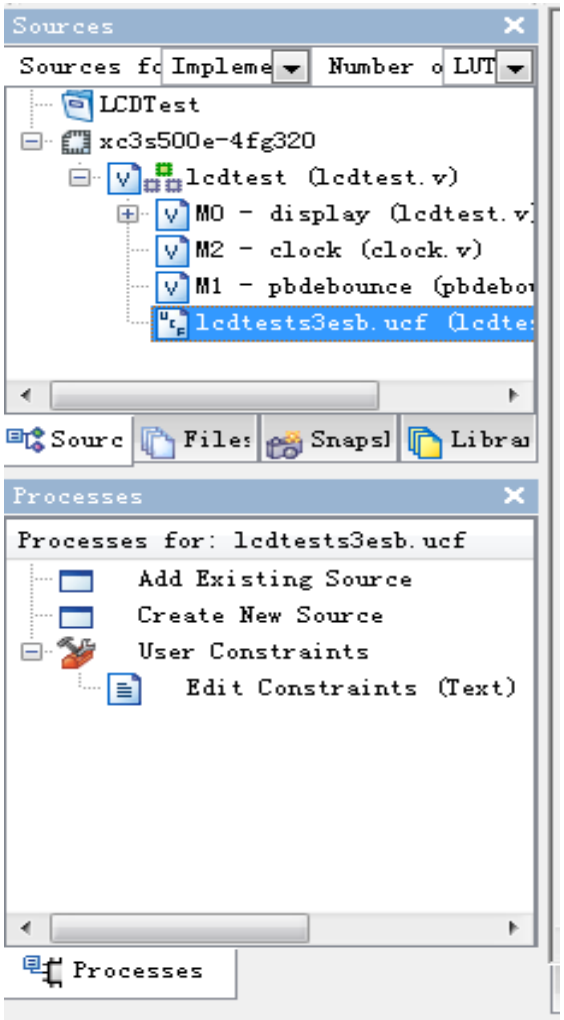
Step 4: New Verilog Module File & UCF File



Step 5: Write Vrilog HDL Code & UCF Code

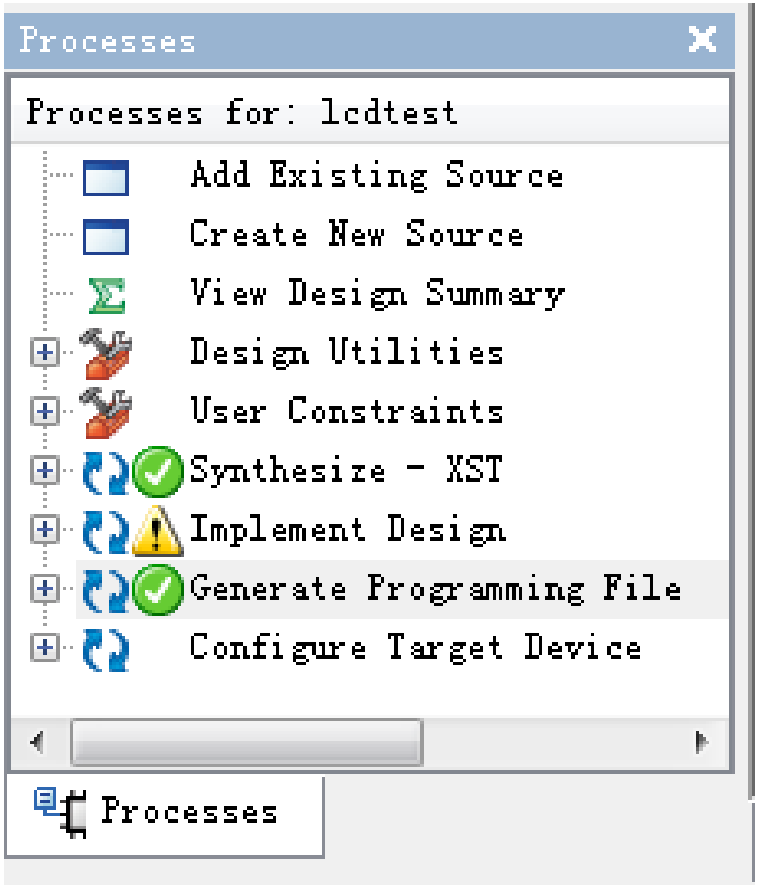
- **See**
  - LCDTest.v
  - clock.v
  - pbdebounce.v
  - lcd.v
  - lcdtests3esb.ucf

Step 6: Edit Constraints

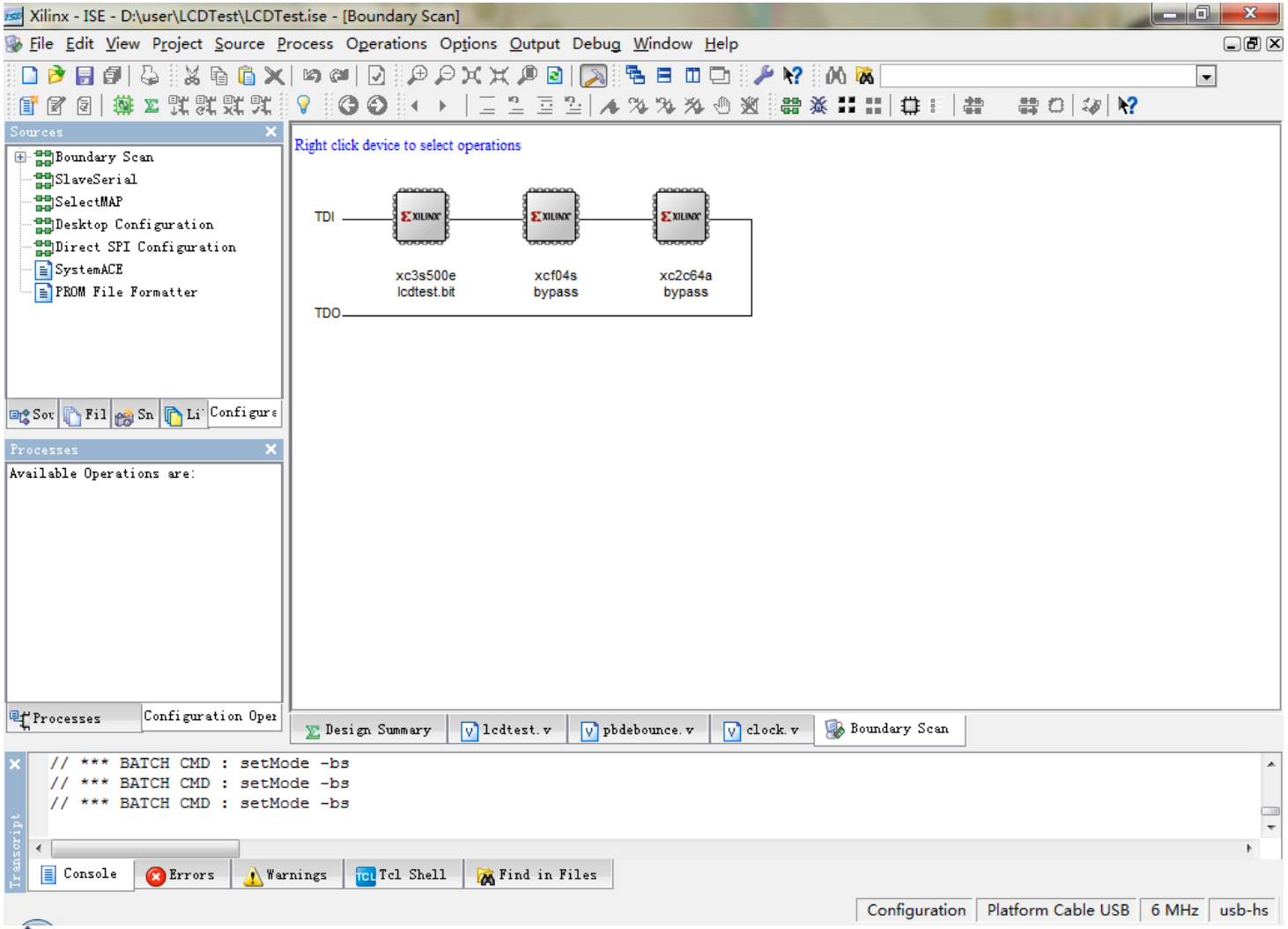


Step 7: Generate Programming File

- **Synthesize**
- **Implement Design**
- **Generate Programming File**



Step 8: Program



浙江大学 ZheJiang University

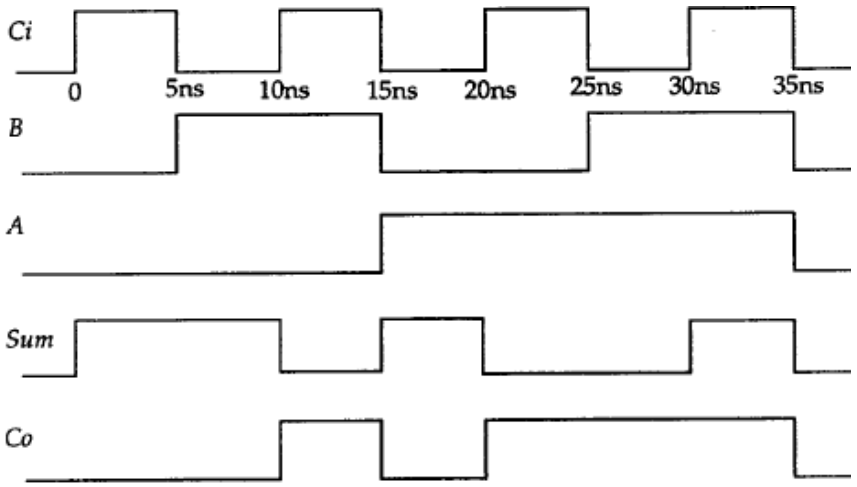
Behavior Simulation Example

```
`timescale 1ns/1ns
module Top;
        reg A, B, Ci;
        wire Co, Sum;

        full_adder F1(A, B, Ci, Sum, Co);
        Initial begin ： ONLY_ONCE
                reg [3:0] i;
                for (i = 0; i < 8; i = i + 1)
                begin
                        {A, B, Ci} = i;
                        #5 $display (“A, B, Ci = %b%b%b”, A, B, Ci,“ : : :
Co,                 Sum=%b%b”, Co, Sum);
                end
        end
endmodule
```

Result

- A, B, Ci = 000 ::: Co, Sum = 00
- A, B, Ci = 001 ::: Co, Sum = 01
- A, B, Ci = 010 ::: Co, Sum = 01
- A, B, Ci = 011 ::: Co, Sum = 10
- A, B, Ci = 100 ::: Co, Sum = 01
- A, B, Ci = 101 ::: Co, Sum = 10
- A, B, Ci = 110 ::: Co, Sum = 10
- A, B, Ci = 111 ::: Co, Sum = 11



Report Format

- 一、实验目的和要求
- 二、实验内容和原理
- 三、实验过程和数据记录
- 四、实验结果分析
- 五、讨论与心得

Thanks!

浙江大学
ZheJiang University