



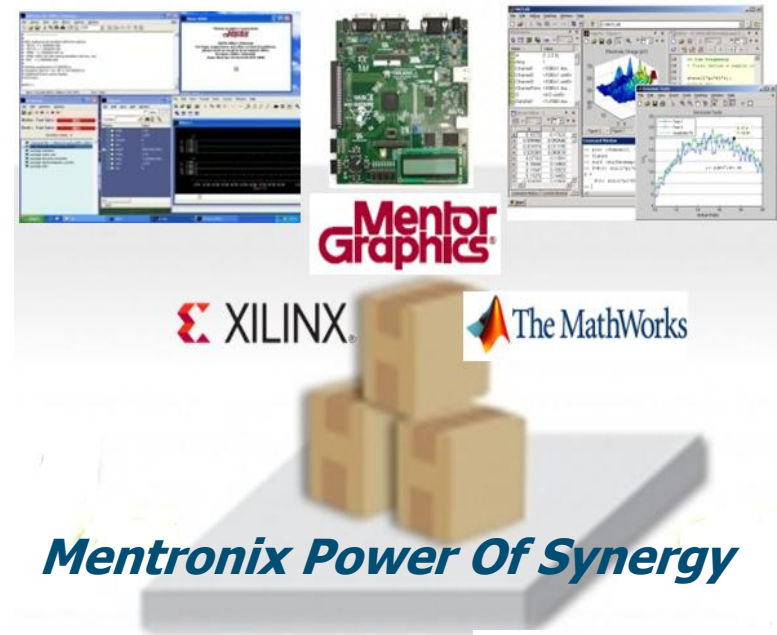
Design Contest

Moustafa Ali
Senior Application Engineer
Training Specialist

Agenda

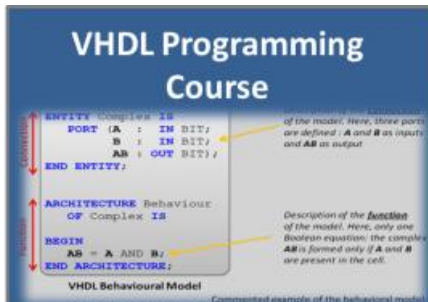
- Mentrionix
- Project Submission, and Evaluation
- Project
- Mentor Flow
- Demonstration

- Founded 2003, headquartered Cairo, Egypt
- An integrated solution provider for MENA region specialized in the field of electronic design.
- Mission is to foster the Micro-electronics Industry by Empowering the Design Community with state-of-the-art Technology.
- Help to promote excellence in the nation's industrial and educational system. Increase knowledge and awareness in educational and industrial sector.
- Provide best in class customer support , help customers to overcome the toughest design challenges, and deliver designs on-time



- **Professional Technical Training Service**
 - Mentor Graphics EDA solutions

VHDL Programming Course



ENTITY Complex IS
PORT (A : IN BIT;
B : IN BIT;
AB : OUT BIT);
END ENTITY;

ARCHITECTURE Behaviour
OF Complex IS
BEGIN
AB = A AND B;
END ARCHITECTURE;

VHDL Behavioural Model

Commented example of the behavioural model.

FPGA Design Course



XILINX
SPARTAN
3

PCB Design Course



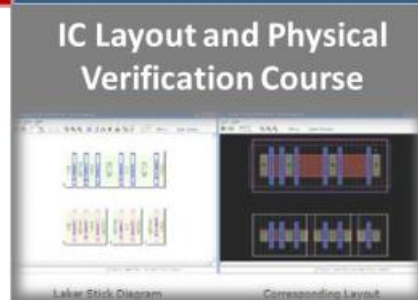
IC from Design to Tape-out Course



Give your Career a LIFT

Join our
Powerful.. Practical.. Professional
Courses

IC Layout and Physical Verification Course



Layout Stick Diagram

Corresponding Layout

RF Analysis Course



Develop your talents &
Knowledge to the fullest to
the professional Electronic
world

[Register Now!](#)

PCB Signal Integrity Course



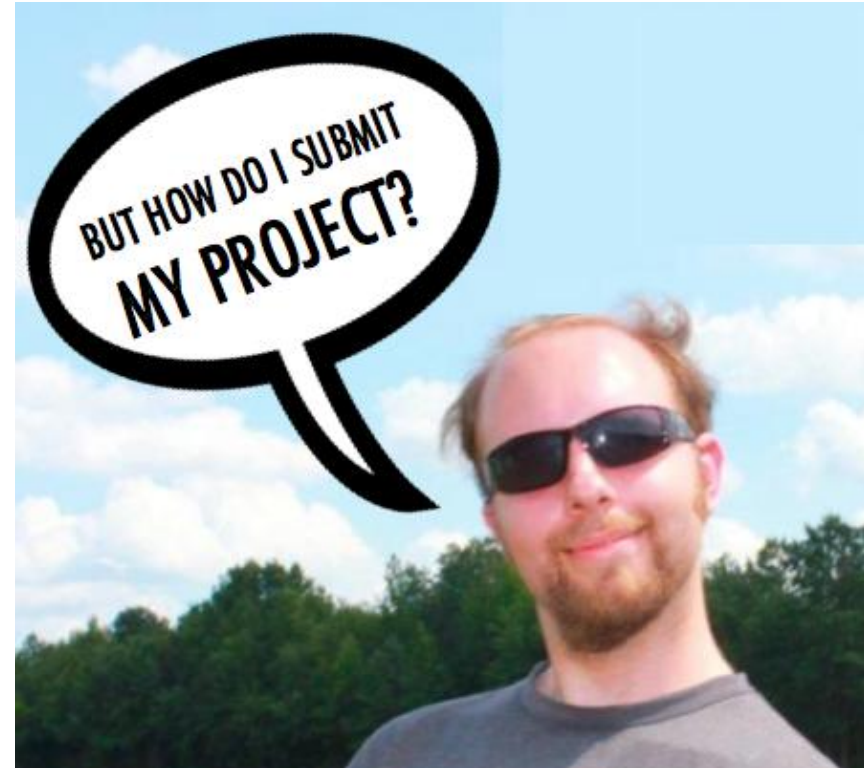
Mentronix

Power Of Synergy

www.mentronix.net

Project Submission, and Evaluation stage 1:

- Your final submission should include the following:
 - The software copy of all the projects required deliverables
- A document describing:
 - Project Implementation method.
 - Problems faced during project.
 - Any reference to design principles and theory where appropriate
 - Acknowledgement of partial or incomplete solutions



Project Submission, and Evaluation stage 2:

- The 5 shortlisted Teams will be expected to host their presentation and discuss their approach, design method and solutions with the Technical committee
 - presentation should not have more than 15 slides.
 - Presentation must include: The proposed solution's name, team name, University/College affiliation, The perspective taken to address the Project/Design challenge, concise description of the proposed solution and Clear illustrations.

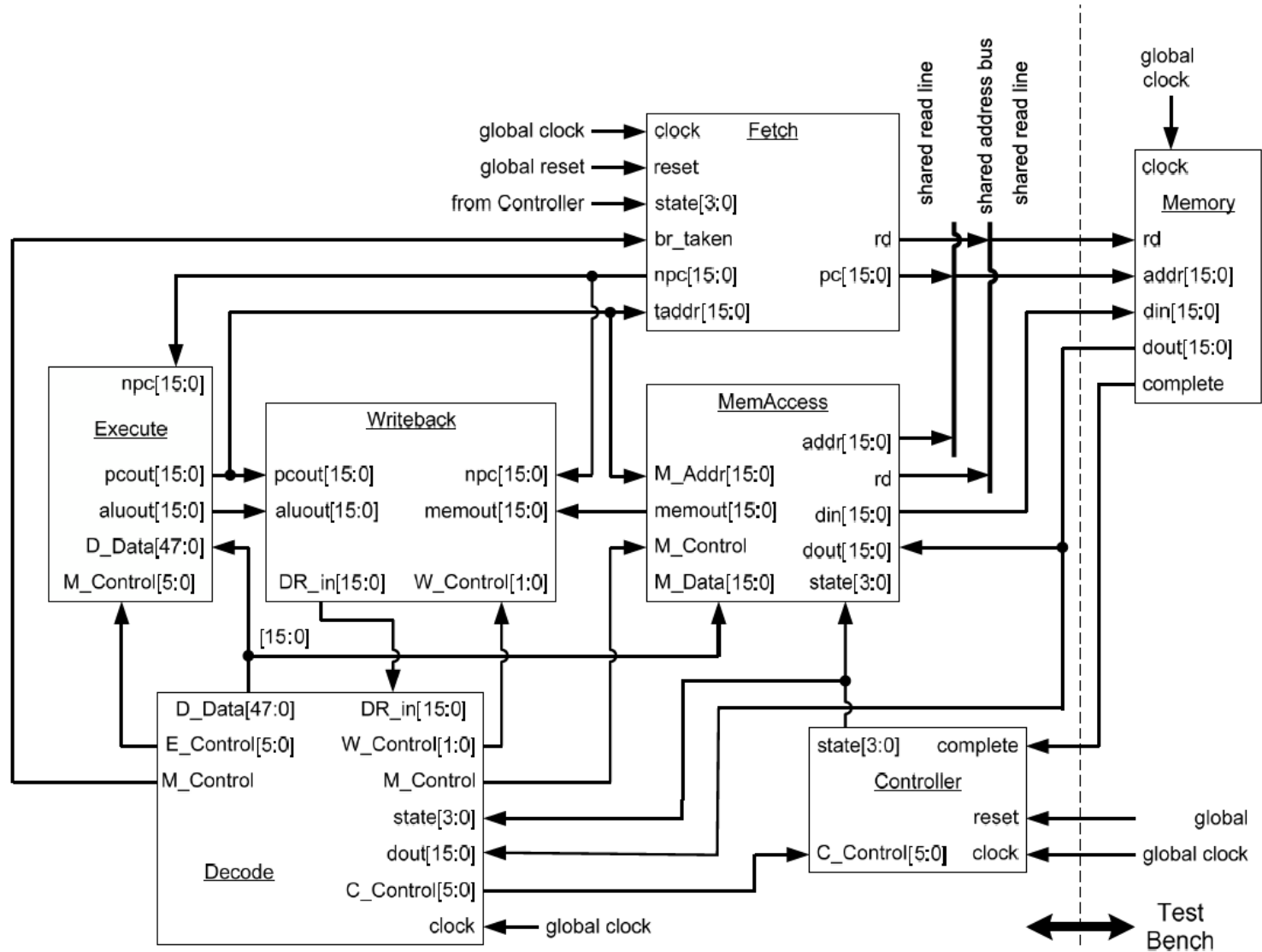


Project Submission, and Evaluation stage 3:

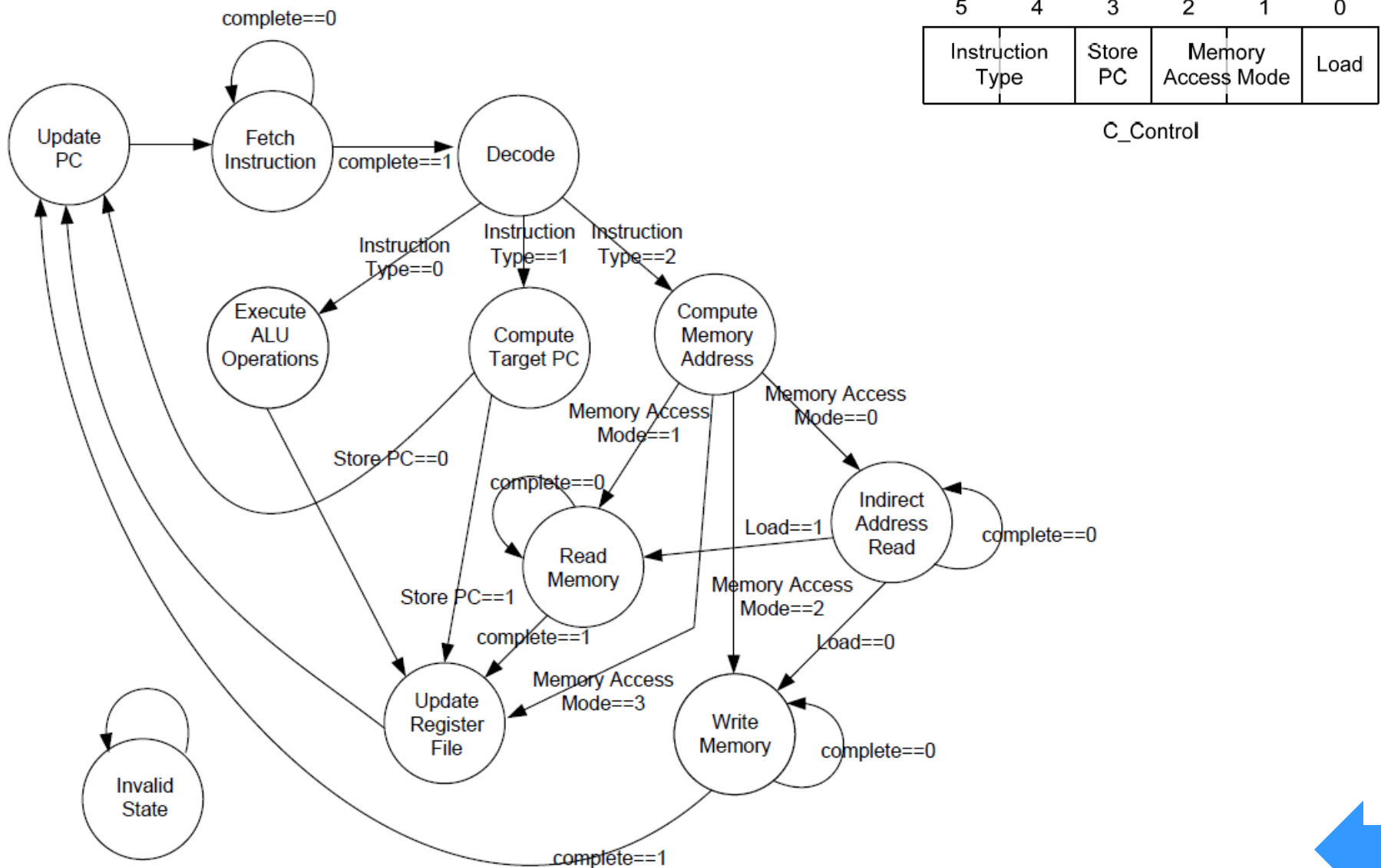
- Three out of the five shortlisted Teams will be selected by the Technical committee



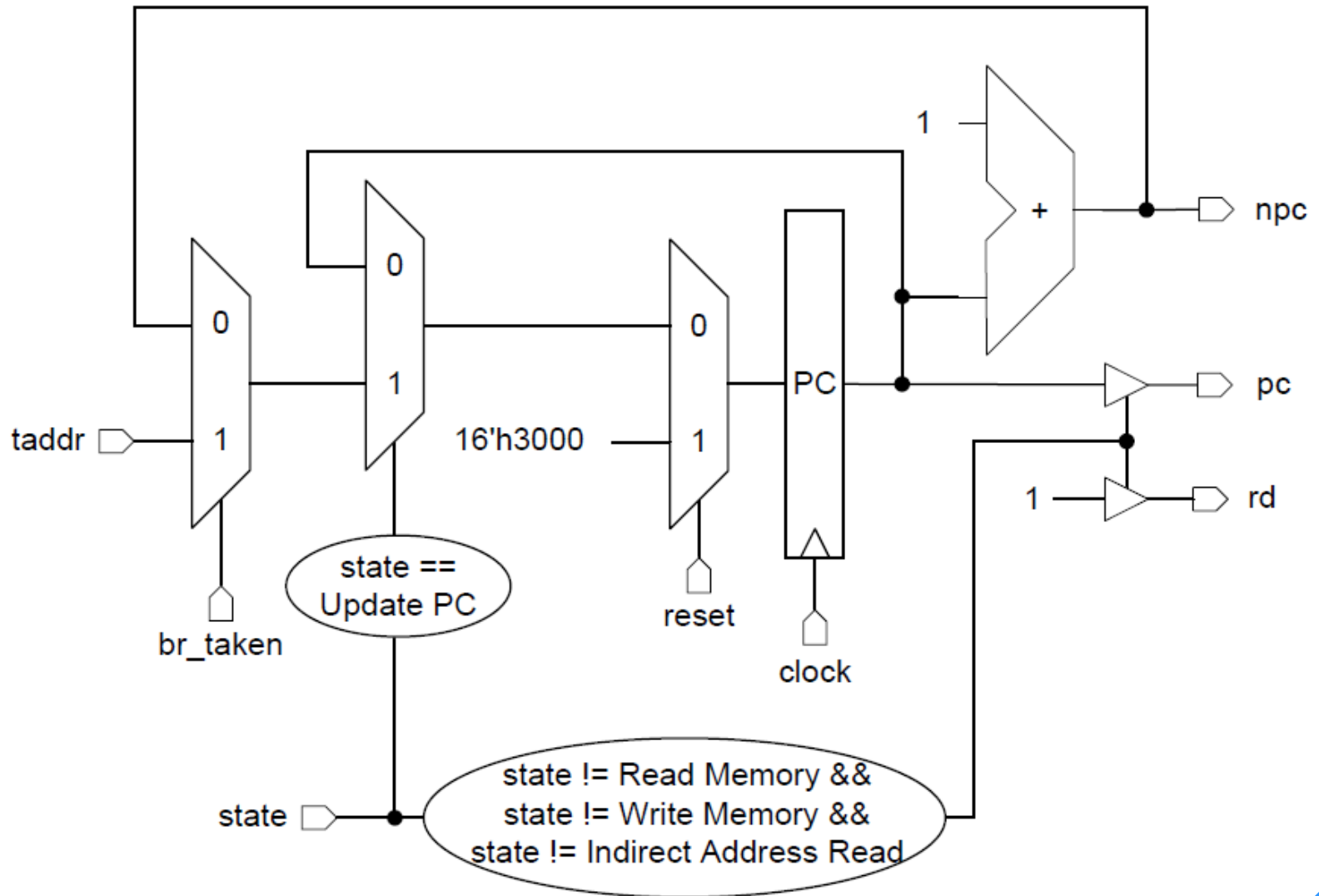
Project



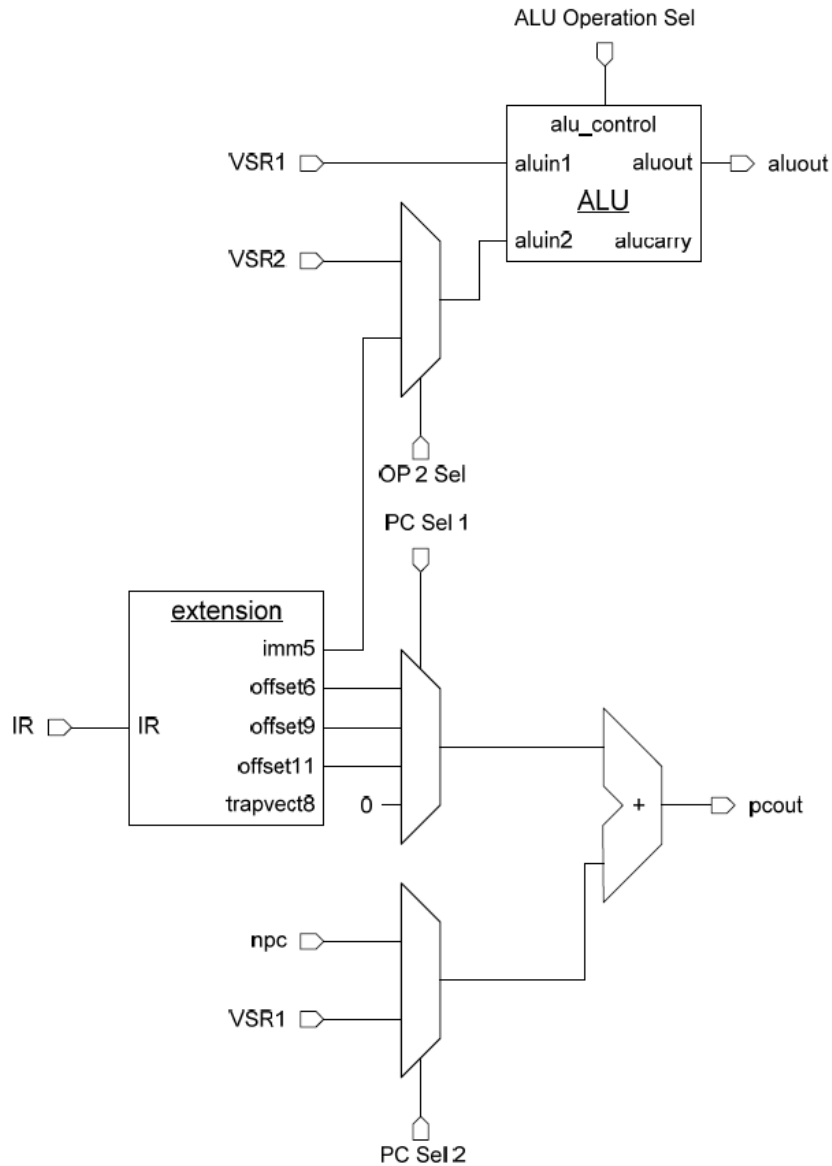
Controller



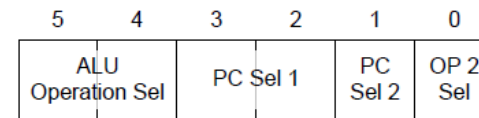
Fetch



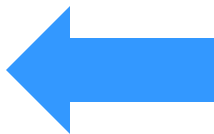
Execute



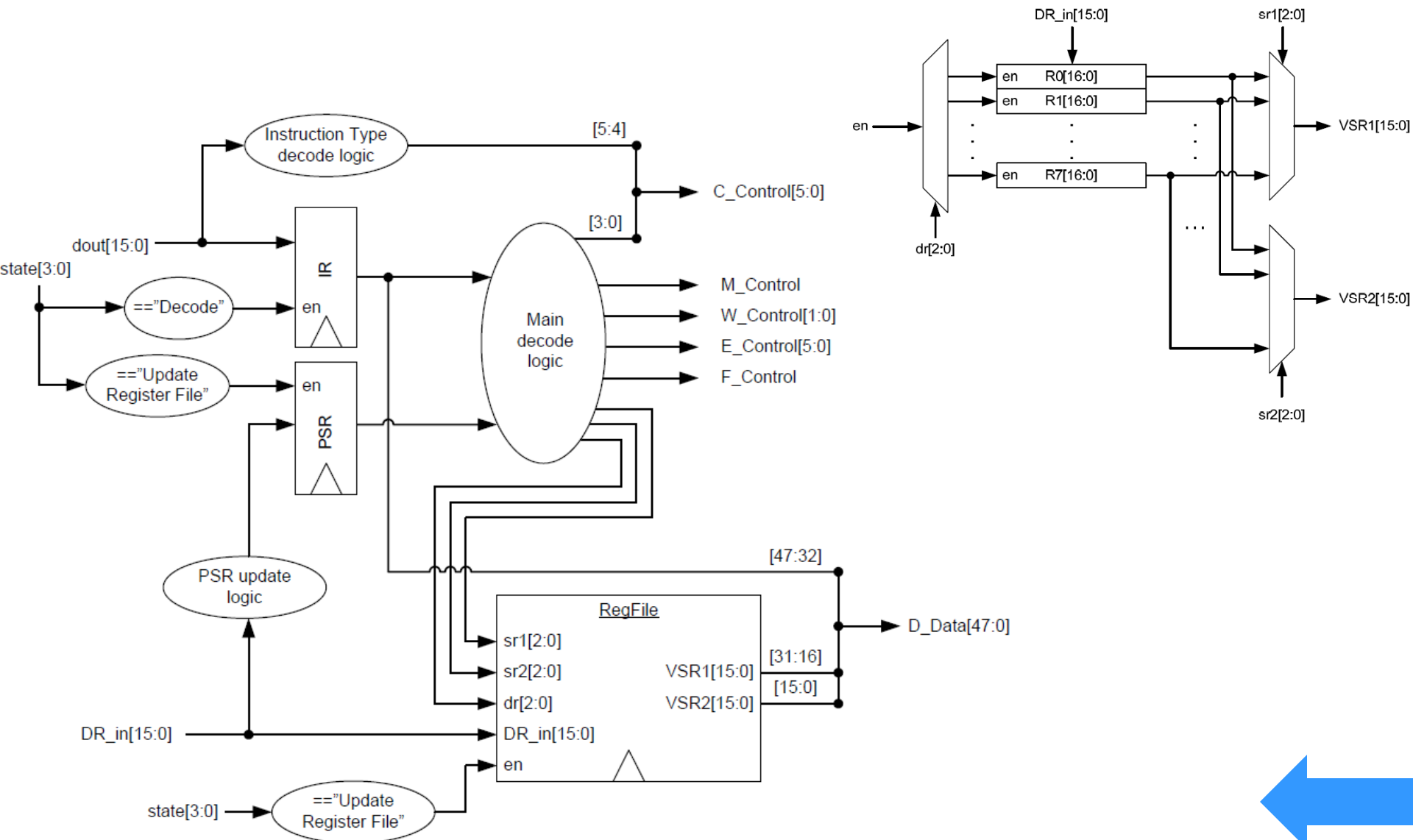
D_Data



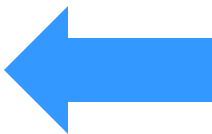
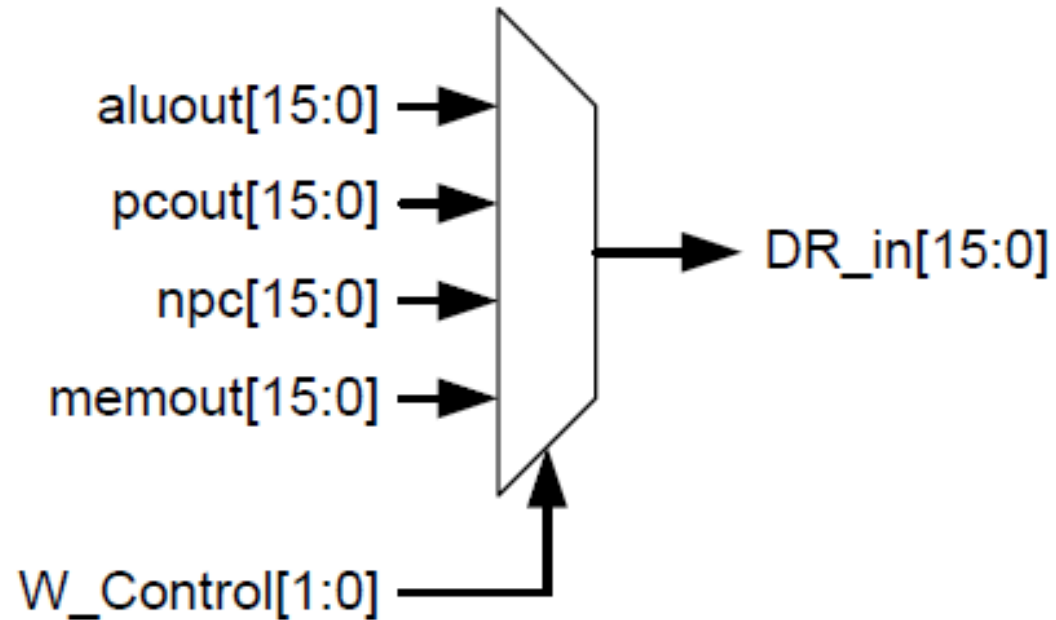
E_Control



Decoder



Write back



ISA

ADD	0001	DR	SR1	0	00	SR2
ADD	0001	DR	SR1	1	imm5	
AND	0101	DR	SR1	0	00	SR2
AND	0101	DR	SR1	1	imm5	
NOT	1001	DR	SR	111111		
BR	0000	n	z	p	PCoffset9	
JMP	1100	0	00	BaseR	000000	
JSR	0100	1	PCoffset11			
JSRR	0100	0	00	BaseR	000000	
RET	1100	0	00	111	000000	

LD	0010	DR	PCOffset9										
LDI	1010	DR	PCOffset9										
LDR	0110	DR	BaseR	offset6									
LEA	1110	DR	PCOffset9										
ST	0011	SR	PCOffset9										
STI	1011	SR	PCOffset9										
STR	0111	SR	BaseR	offset6									
TRAP	1111	0000	trapvect8										
RTI	1000	000000000000											
reserved	1101												

ISA

■ Opcodes

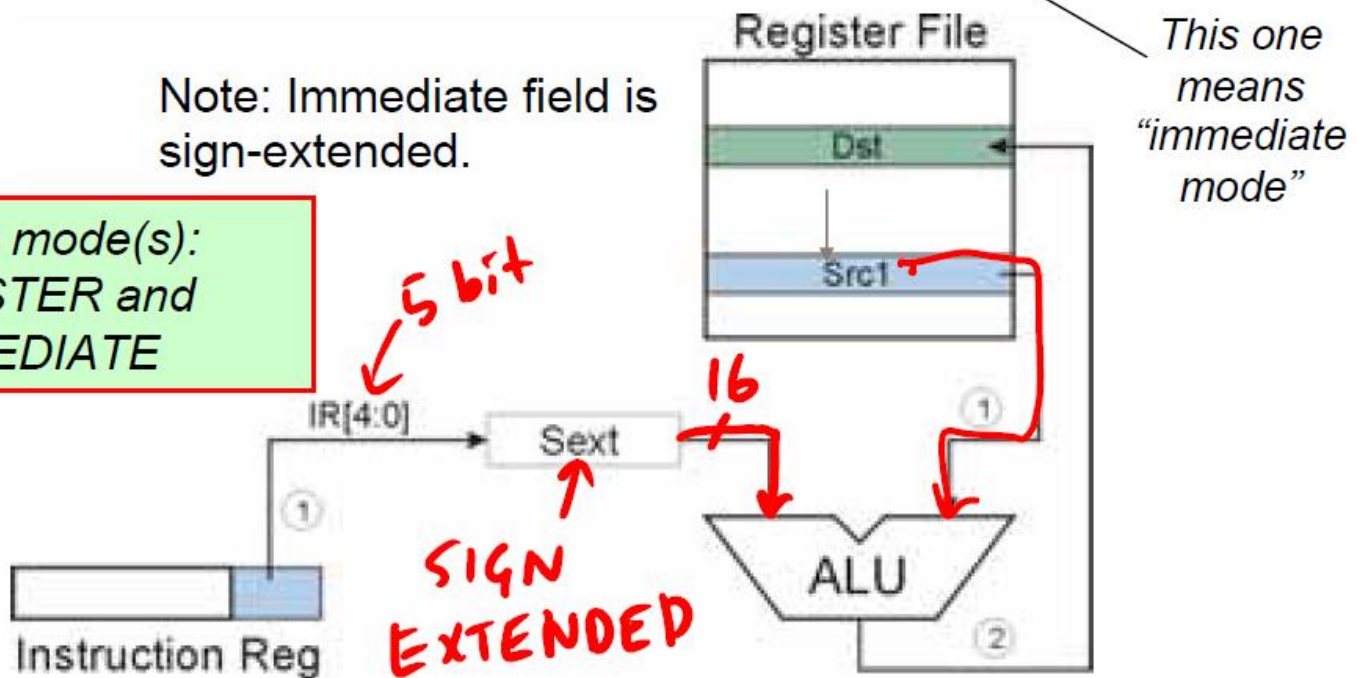
- ◆ 16 opcodes (1 unused/reserved)
- ◆ Operate (Logical or Arithmetic) instructions:
 - ★ ADD, AND, NOT
- ◆ Data movement instructions:
 - ★ LD, LDI, LDR
 - ★ ST, STR, STI
 - ★ LEA
- ◆ Control instructions:
 - ★ BR, JSR/JSRR, JMP, RTI, TRAP
- ◆ Some opcodes set/clear condition codes, based on result:
 - ★ N = negative (< 0)
 - ★ Z = zero
 - ★ P = positive (> 0)

ADD/AND

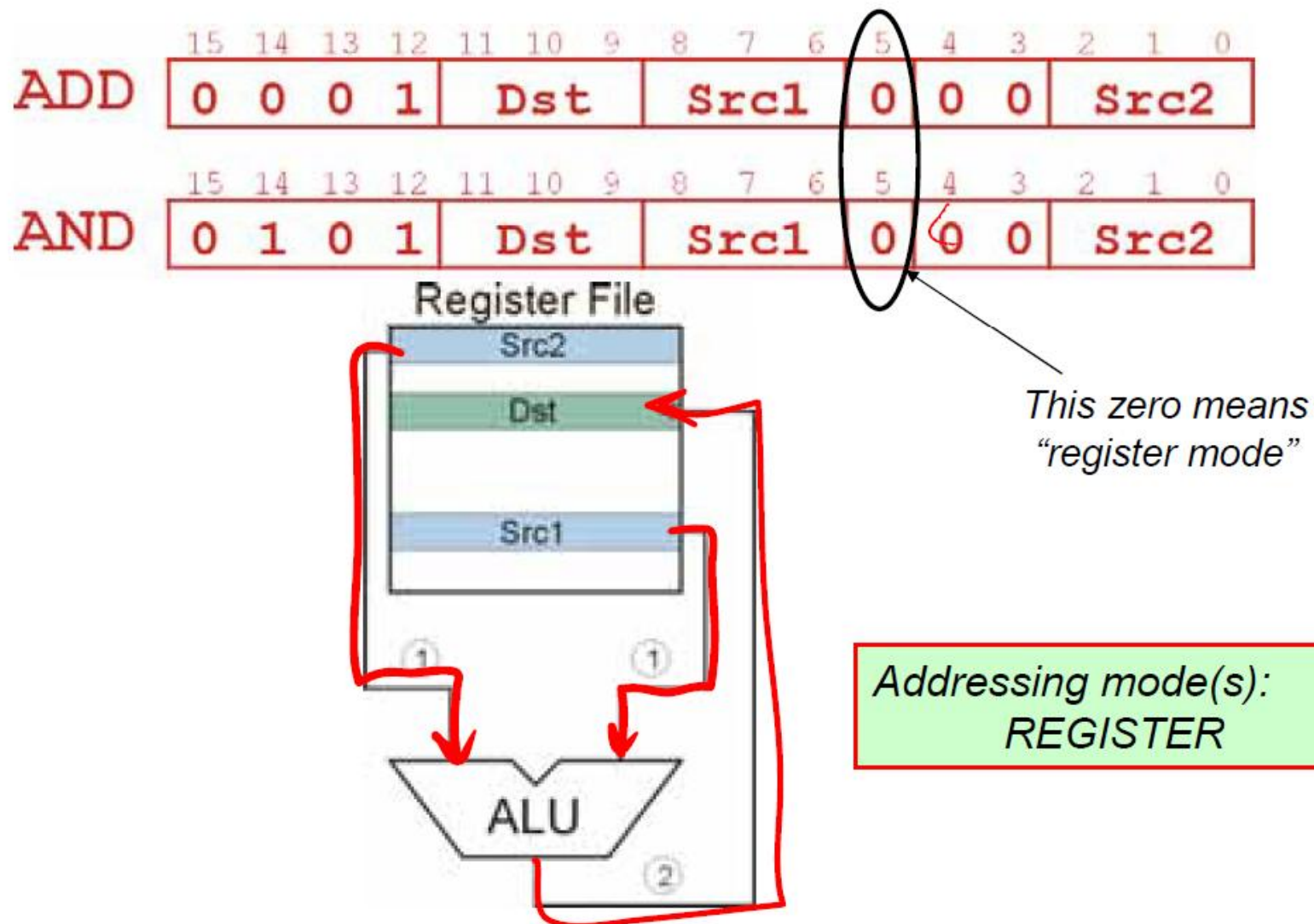


Note: Immediate field is sign-extended.

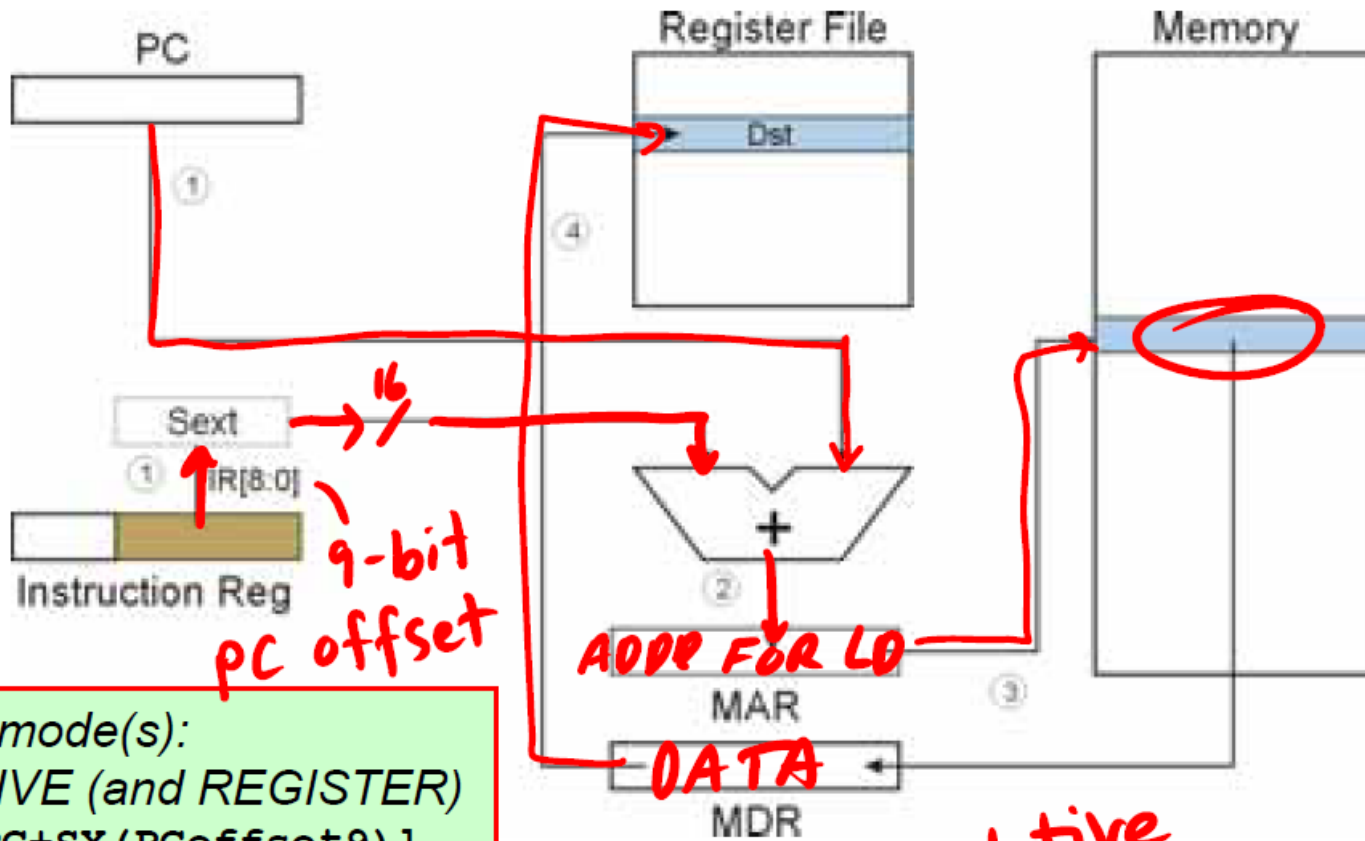
Addressing mode(s):
REGISTER and IMMEDIATE



ADD/AND



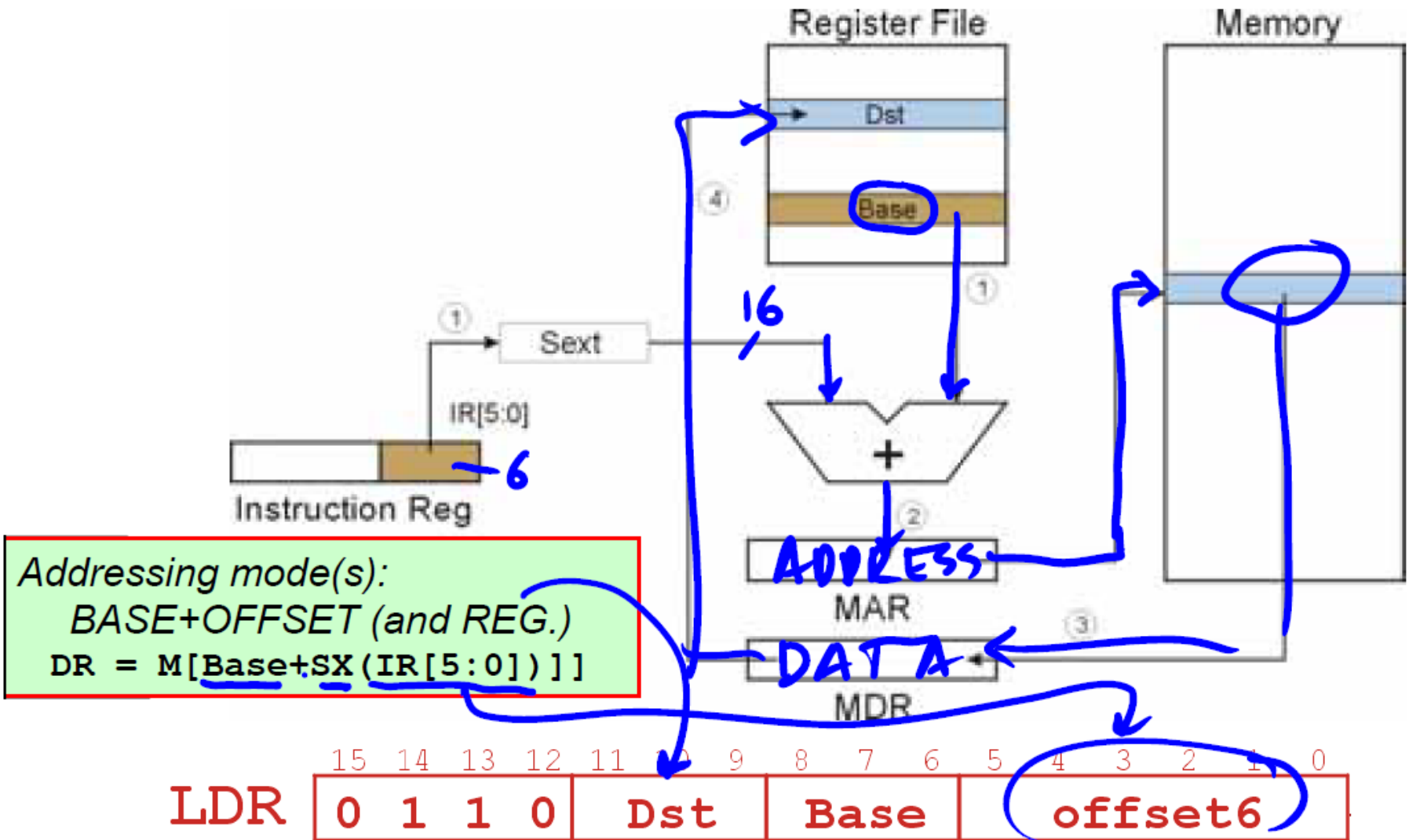
LD (Load Direct)



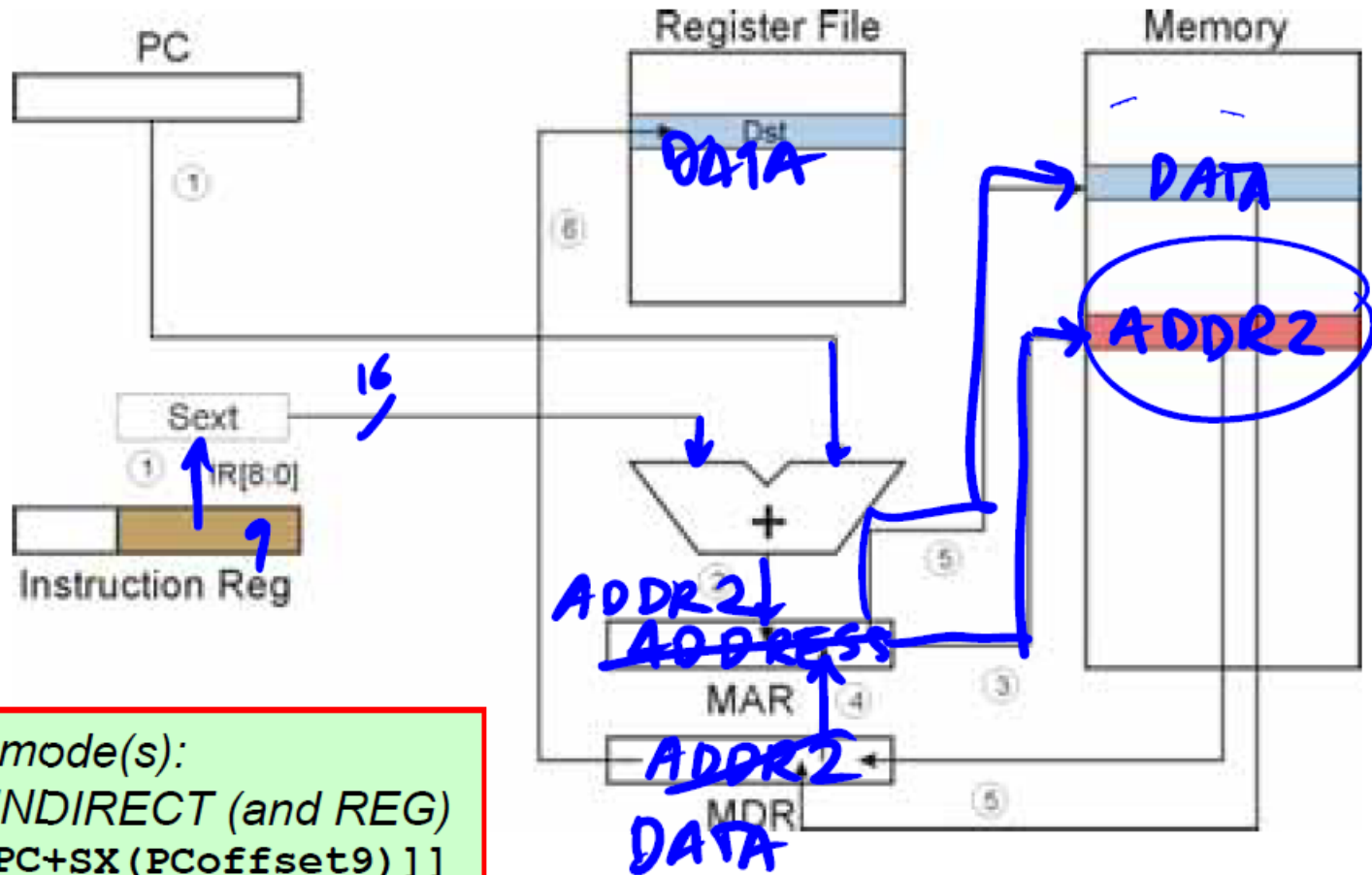
Addressing mode(s):
PC-RELATIVE (and REGISTER)
 $DR = M[PC + SX(PCoffset9)]$



LDR (Load Base + Offset)



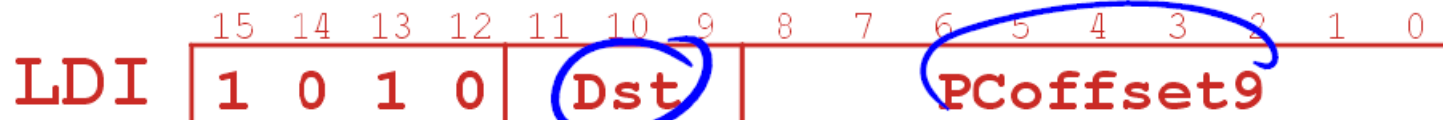
LDI (Load Indirect)



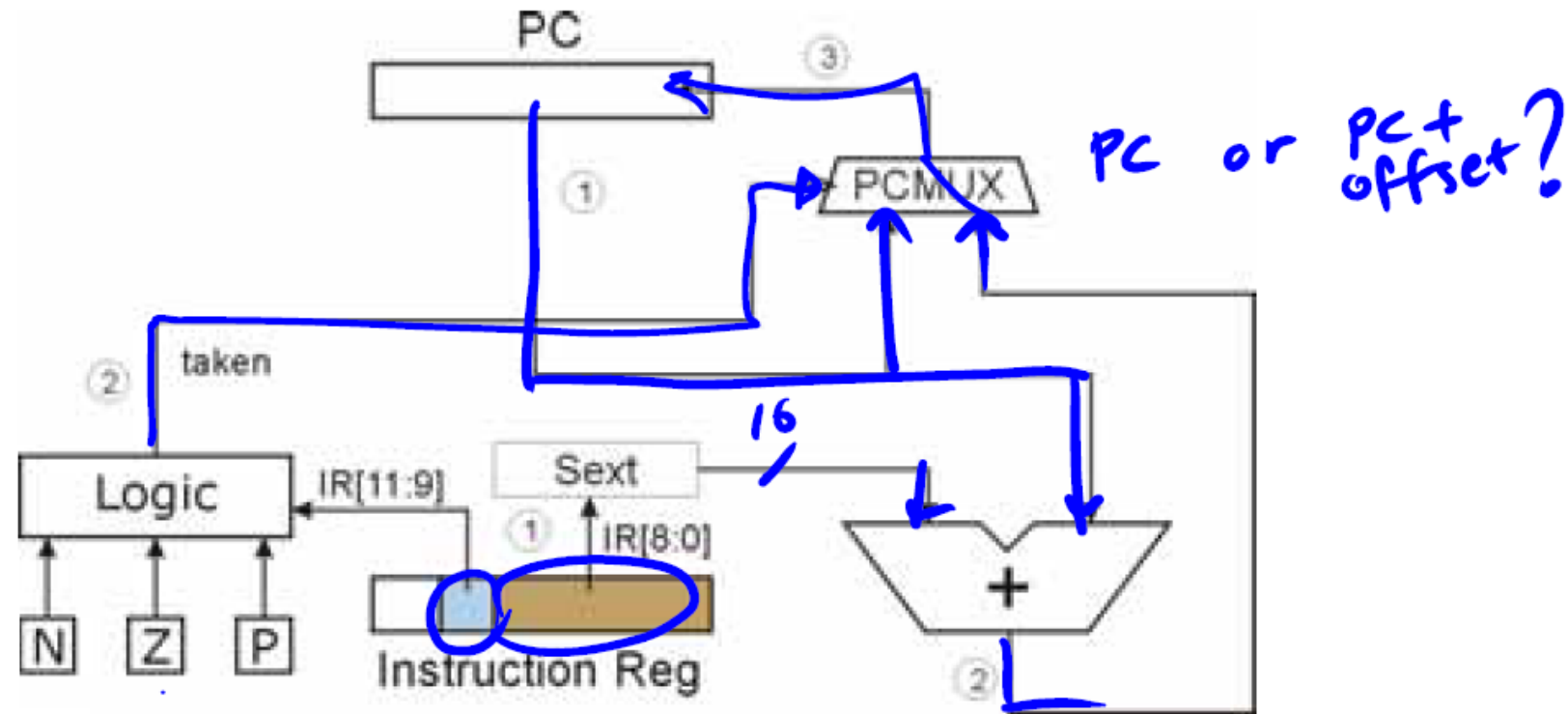
Addressing mode(s):

MEMORY INDIRECT (and REG)

$DR = M[M[PC+SX(PCoffset9)]]$

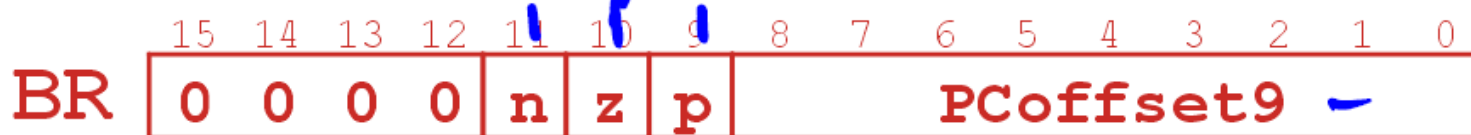


BR (Branch)

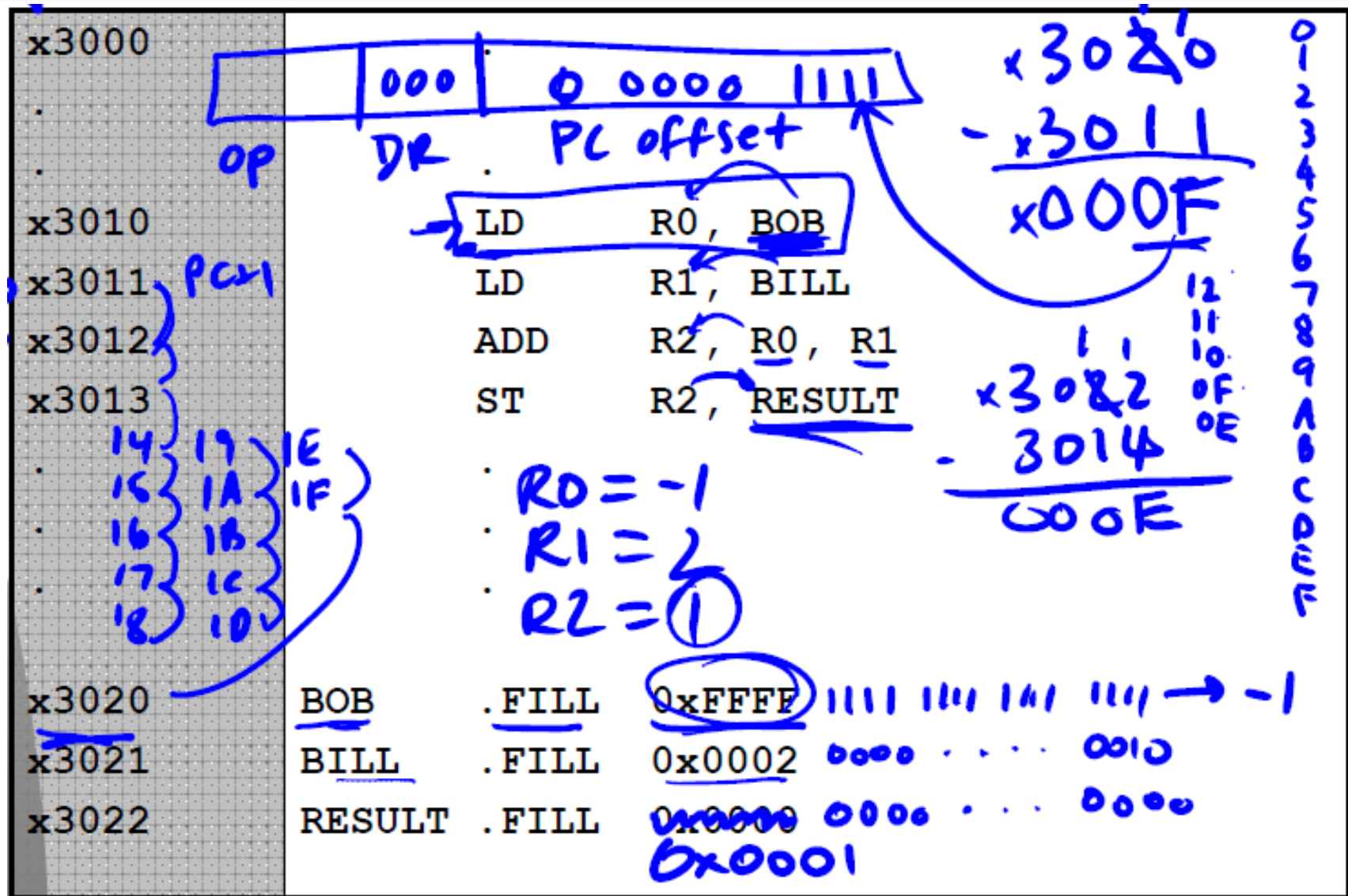


Addressing mode(s):
 PC-RELATIVE
 $PC = PC + SX(PCoffset9)$

BRnzp



Example 1



Example 2

x3000	.		
.	.		
.	.		
x3010	<u>LDI</u>	R0, <u>BOB_P</u>	R0 = -1
x3011	LDI	R1, BILL_P	R1 = 2
x3012	ADD	R2, <u>R0</u> , <u>R1</u>	R2 = 1
x3013	STI	R2, RES_P	
.	.		
.	.		
x301D	RES_P	.FILL 0x3022	
x301E	BOB_P	.FILL 0x3020	
x301F	BILL_P	.FILL BILL x5021	
x3020	BOB	.FILL 0xFFFF	
x3021	BILL	.FILL 0x0002	
x3022	RESULT	.FILL 0x0000 0x0001	

3021
5020
3021
FFFF
0002
0000

Example 3

- Compute the sum of 12 integers

AND ^{???} R2, R2, #0
 AND R0, R0, #0

BR_n SUM
 (fall through...)
 ST R2, TOMEM

SUM:

ADD R0, R0, #1 ; R0=1

ADD R2, R2, R0 ; R2=1

*ADD R1, R0, #-12 ; if R0=12 then stop

BR_{pz} DONE

BR_{nzp} SUM

DONE: ST R2, TOMEM

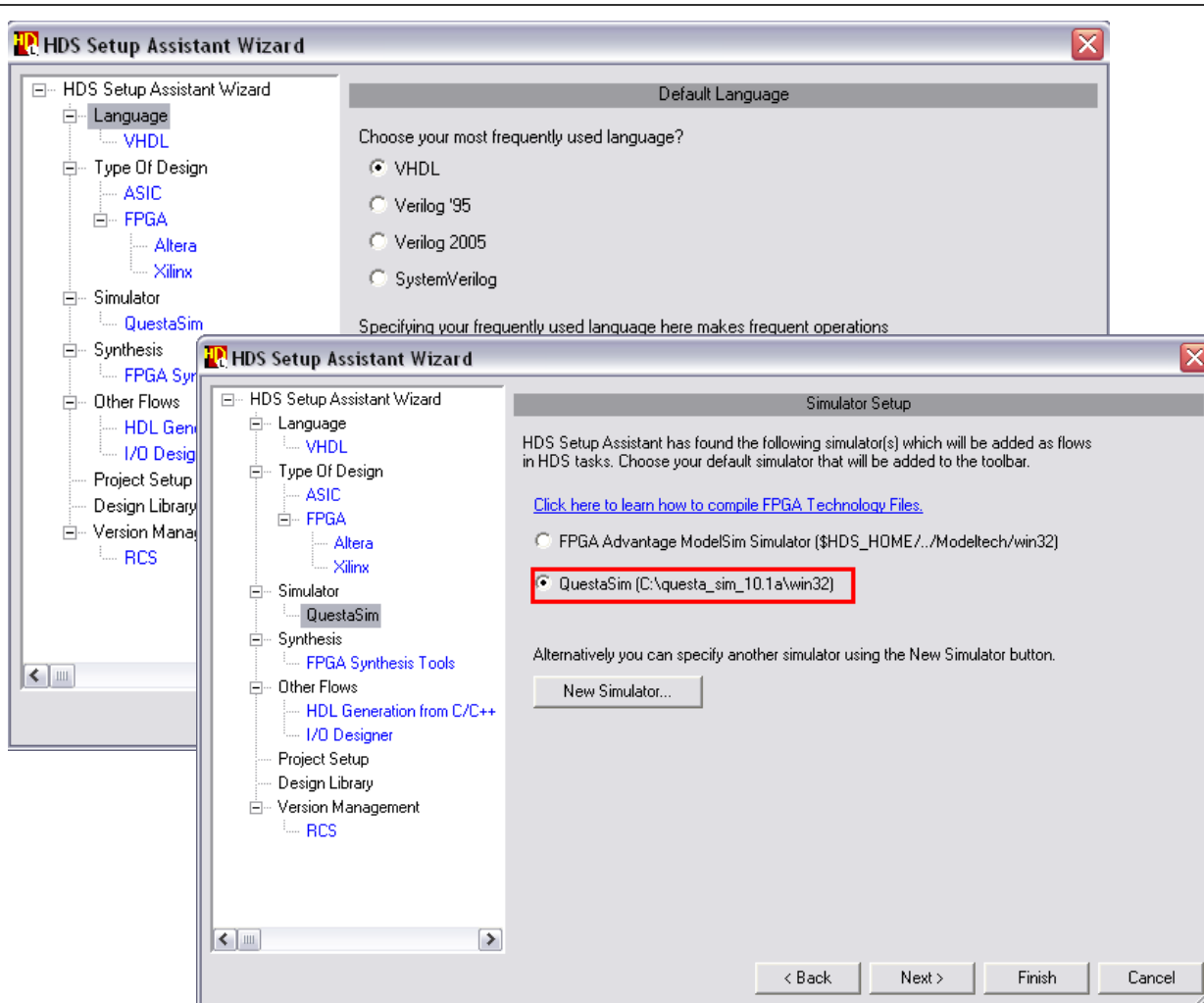
(R0-12)
 [PZ] n

01100
 -12=10100



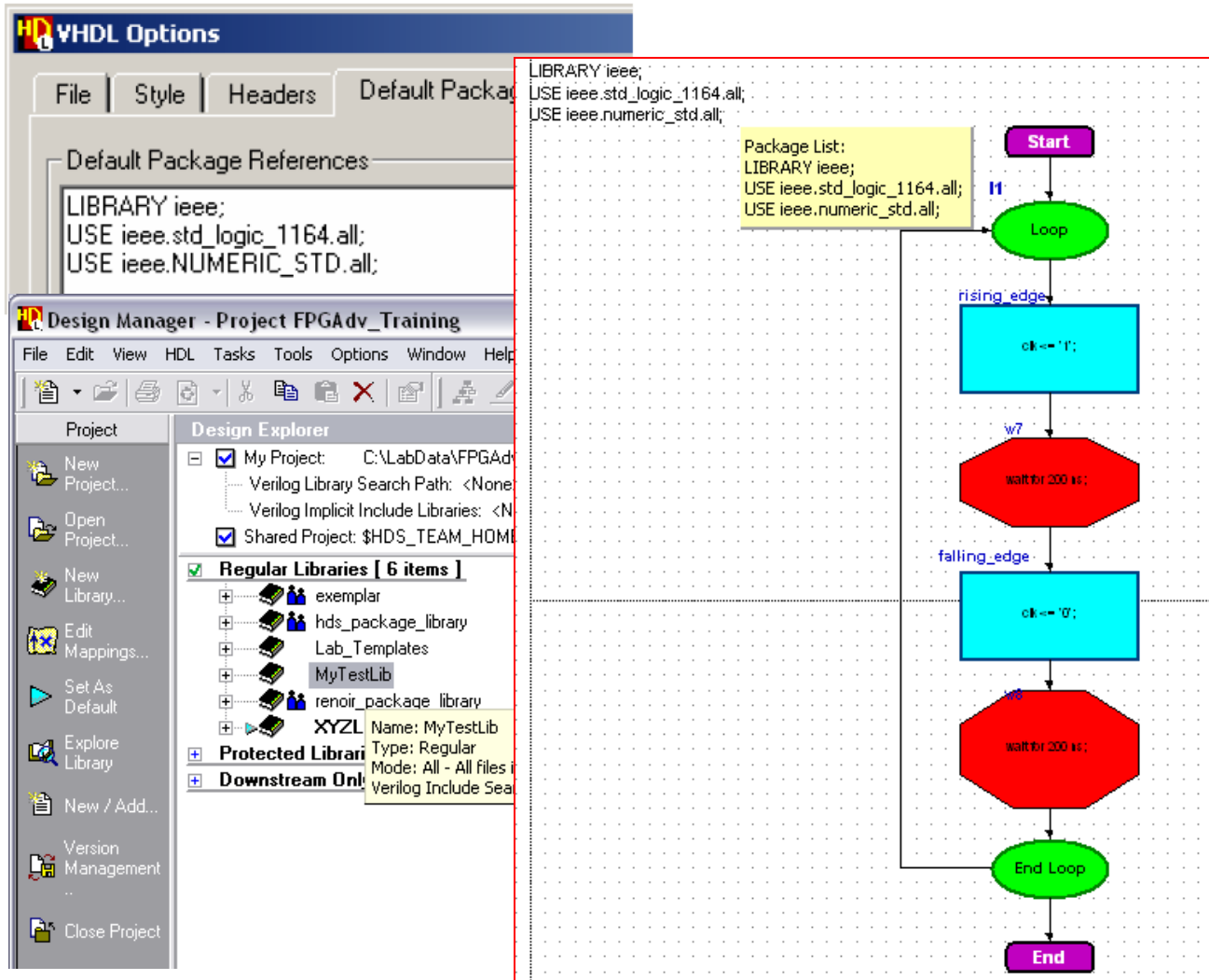
DEMO

Step 1: opening Project in HDL Designer



1. Click the FPGAdv with PS 8.2 icon on the desktop.
2. If you do not see the HDS Setup Assistant Wizard, select HDS Setup Assistant from the Help menu.
3. Select Language: **VHDL**
Type Of Design: **FPGA**,
Technology: **Altera**,
Xilinx
Simulator: **FPGA Advantage**
QuestaSim Simulator
4. Finally, click **Finish**

Step 1: opening Project in HDL Designer



1. Options > VHDL

We only want to use the shown two IEEE packages.

2. Open the *MyTestLib* Library by double-clicking the library name in the Design Explorer window.

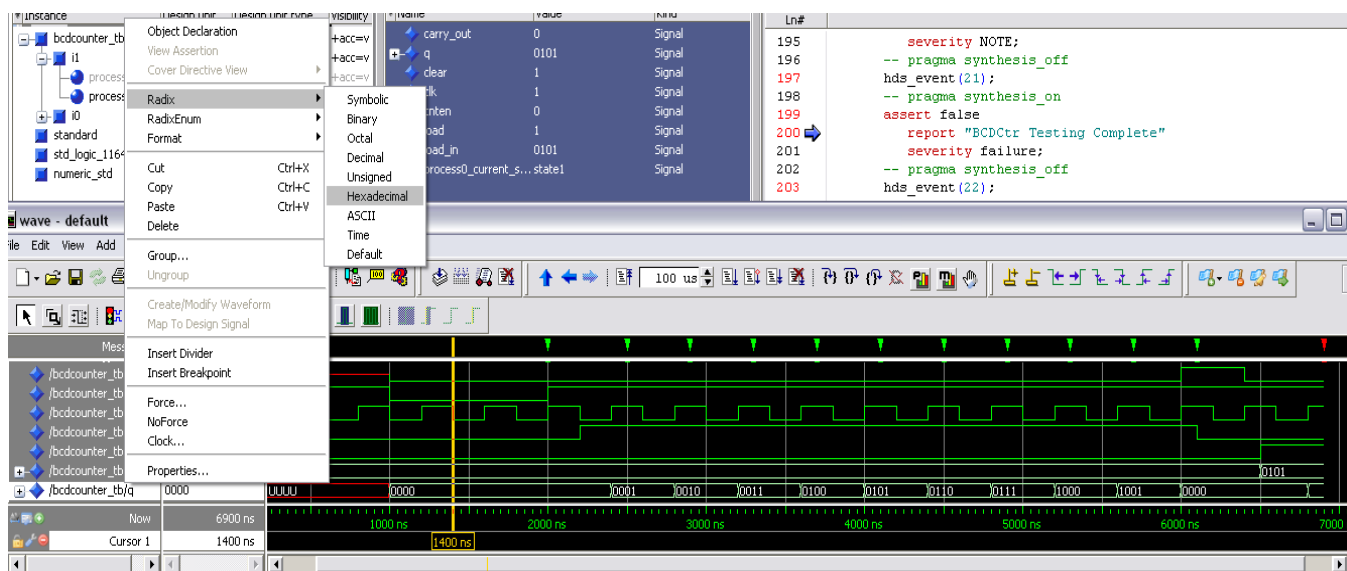
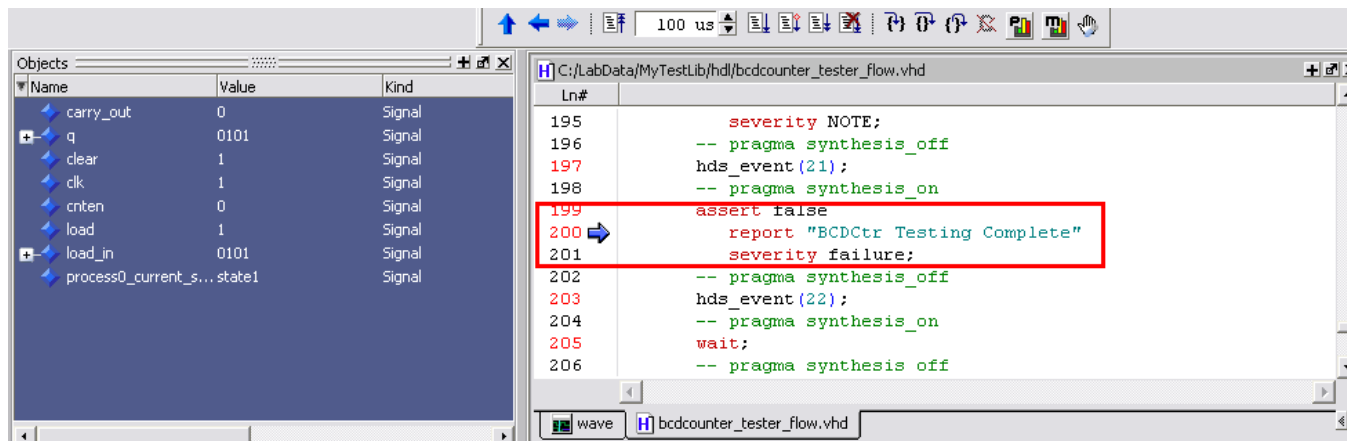
3. Open BCDCounter_tb

4. Double click the BCDCounter to open it.

5. Open the BCDCounter_tester see the Two processes

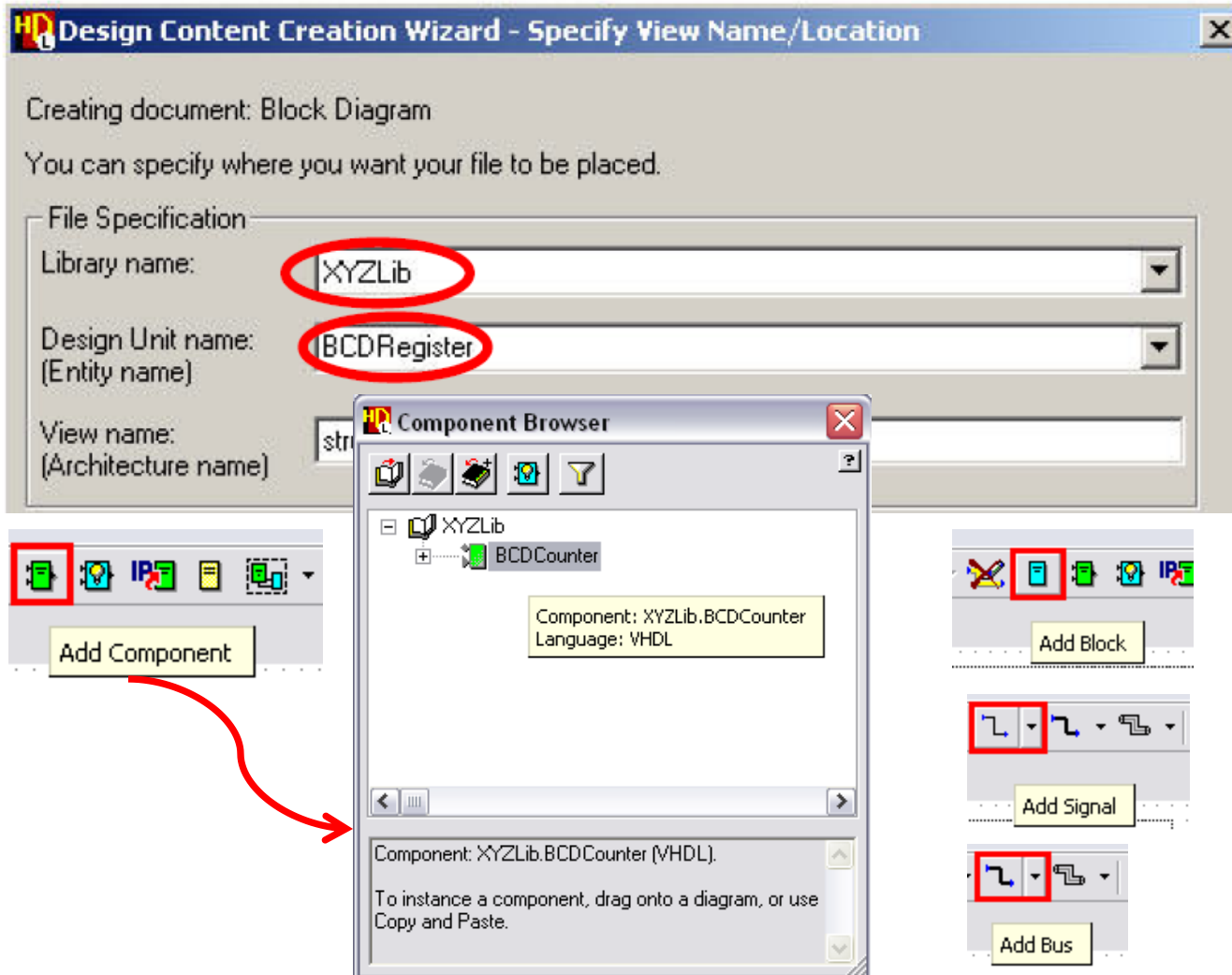
6. Run the Simulation for BCDCounter_tb.

Step 1: opening Project in HDL Designer



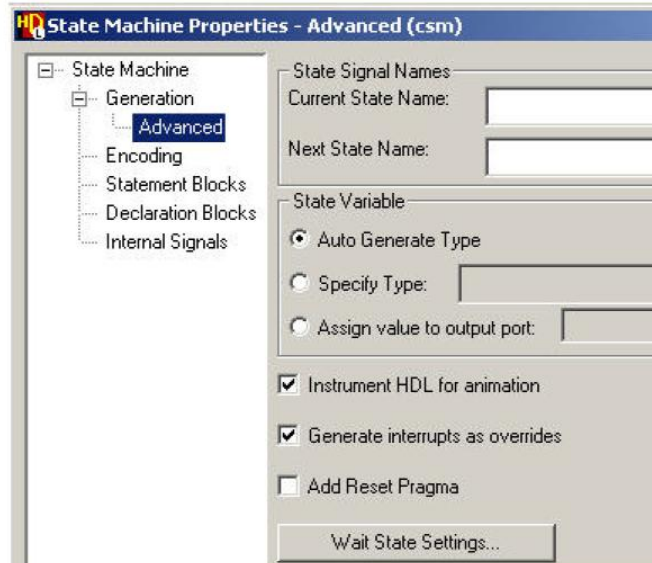
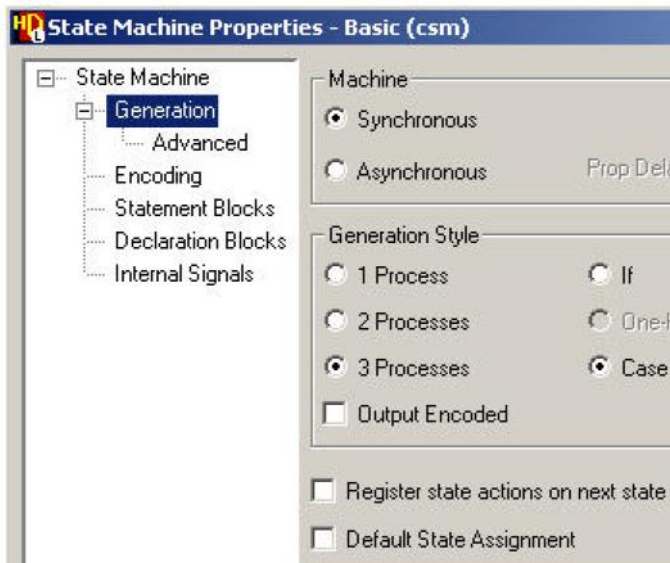
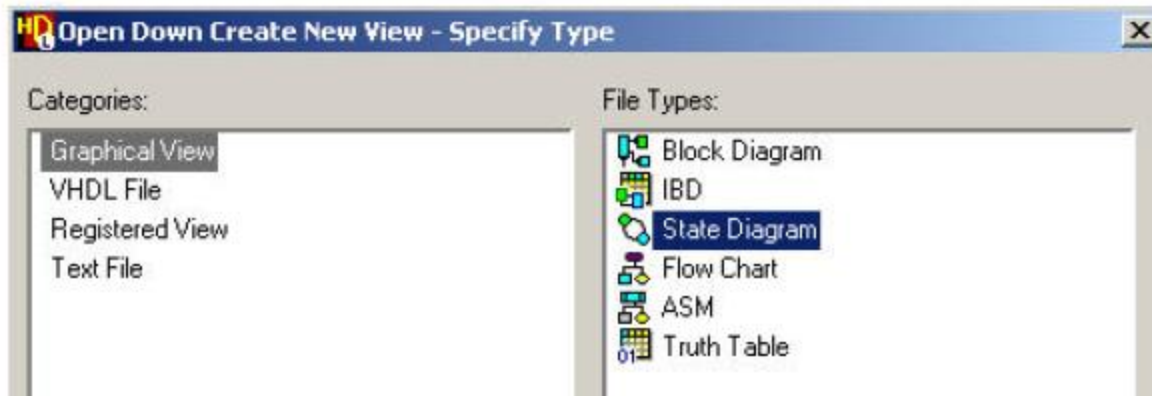
1. Menu: File > Wave
2. Drag and drop the signals in the object window.
3. Write 100 us in the run time.
4. Select the wave tab.
5. Move the cursor to the needed location for test.
6. Select q and RMB > Radix > Hexadecimal
7. what is the Green and Red Marks in the Wave window.

Step 2: Block Diagram Creation



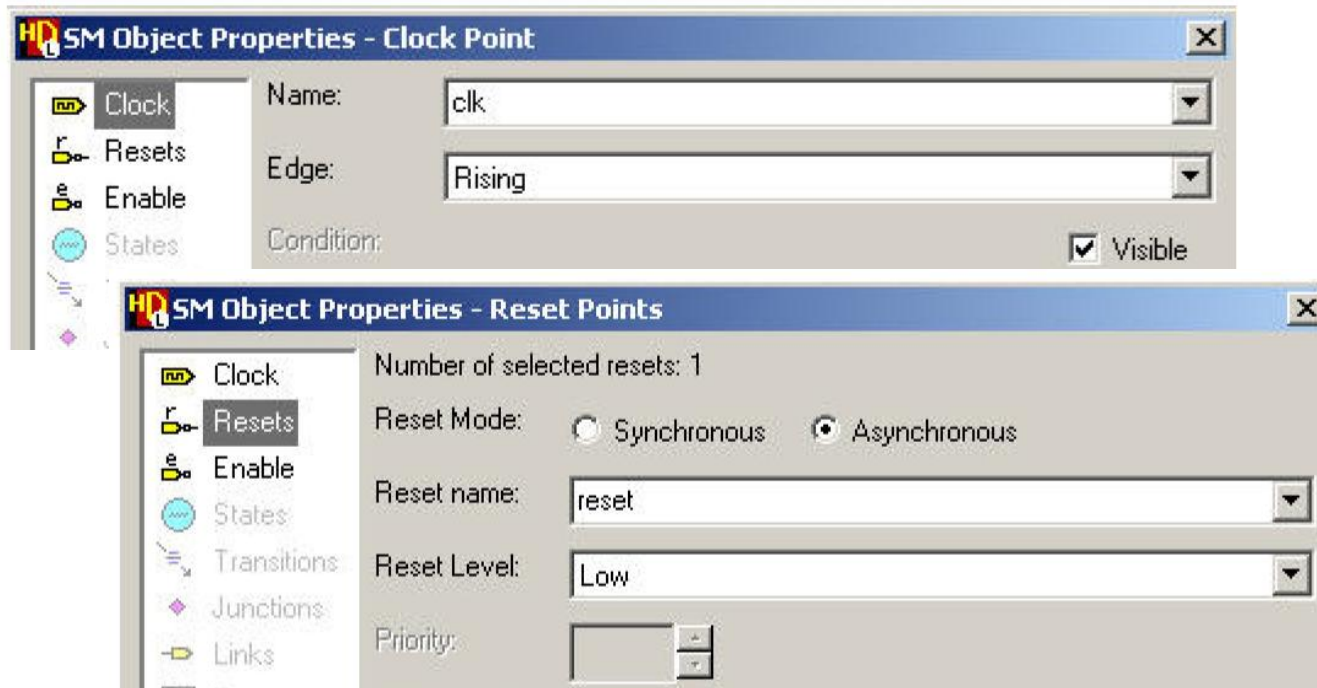
1. menu: **File > New > Design Content.**
2. Complete the Fields as shown Then Click **Finish.**
3. Instantiate two BCDCounter components from XYZLib by click Add Component button then drag and drop BCDCounter.
4. Add a new block called BCDRegControl by Add Block icon.
5. Complete the design by adding wires and busses from the shown Icons

Step 3: FSM Creation



1. Double-click the BCDRegControl block. Select a file type of Graphical View > State Diagram in the Open Down Create New View dialog box. Click Next. In the Specify View Name wizard dialog box keep the default view name fsm.sm and finish the process.
2. Open the State Machine Properties dialog box.
3. Set the shown settings.

Step 3: FSM Creation

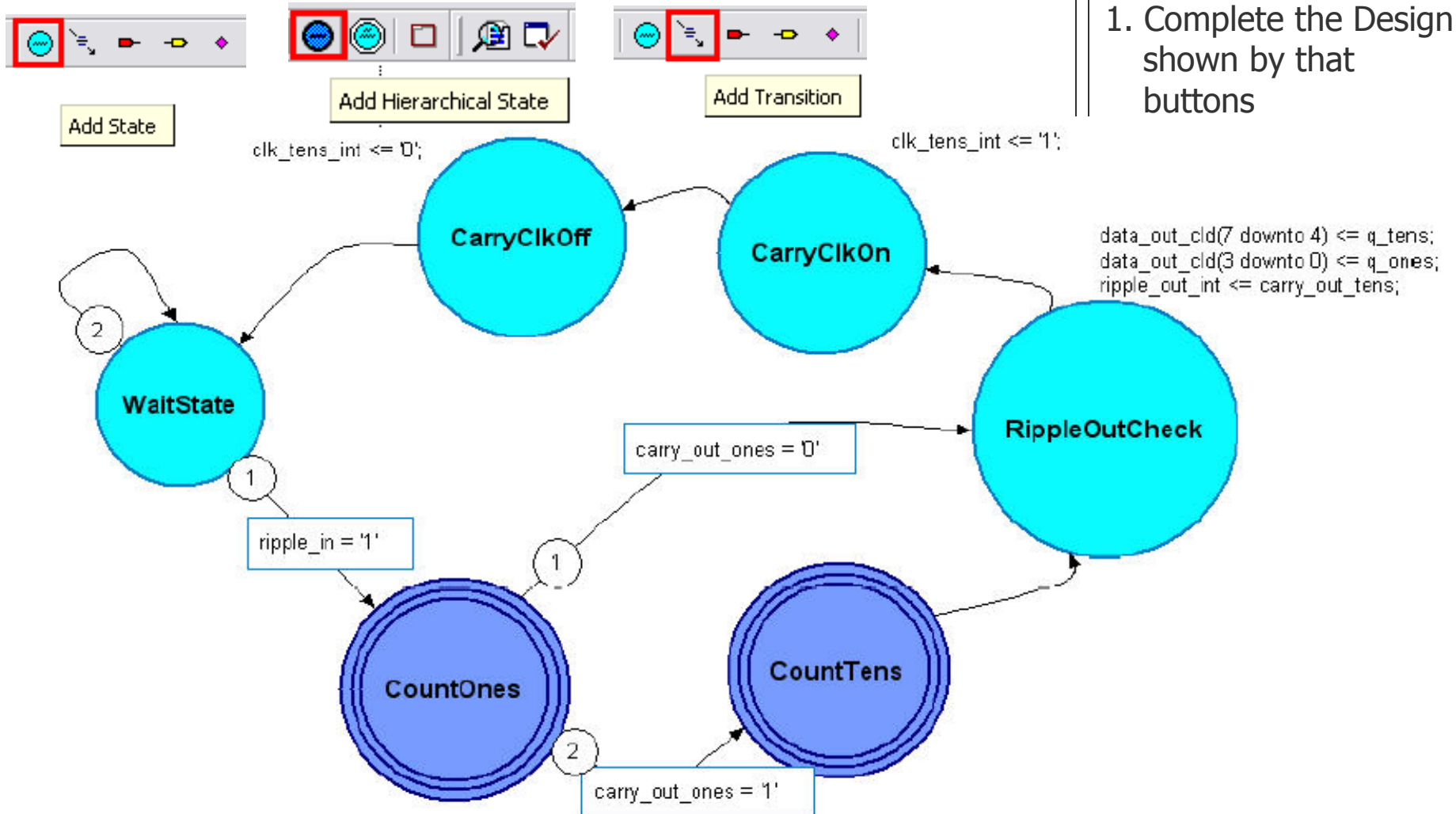


1. Double-click the CLK signal in the State Diagram. *The SM Object Properties box should come up.*
2. Set the shown settings to the clock and Reset signals
3. Select the Signals Table in the Structure Navigator, and set the parameters as shown in the next slide.
4. Delete the "load_en" port from the BCDRegControl table.

Step 3: FSM Creation

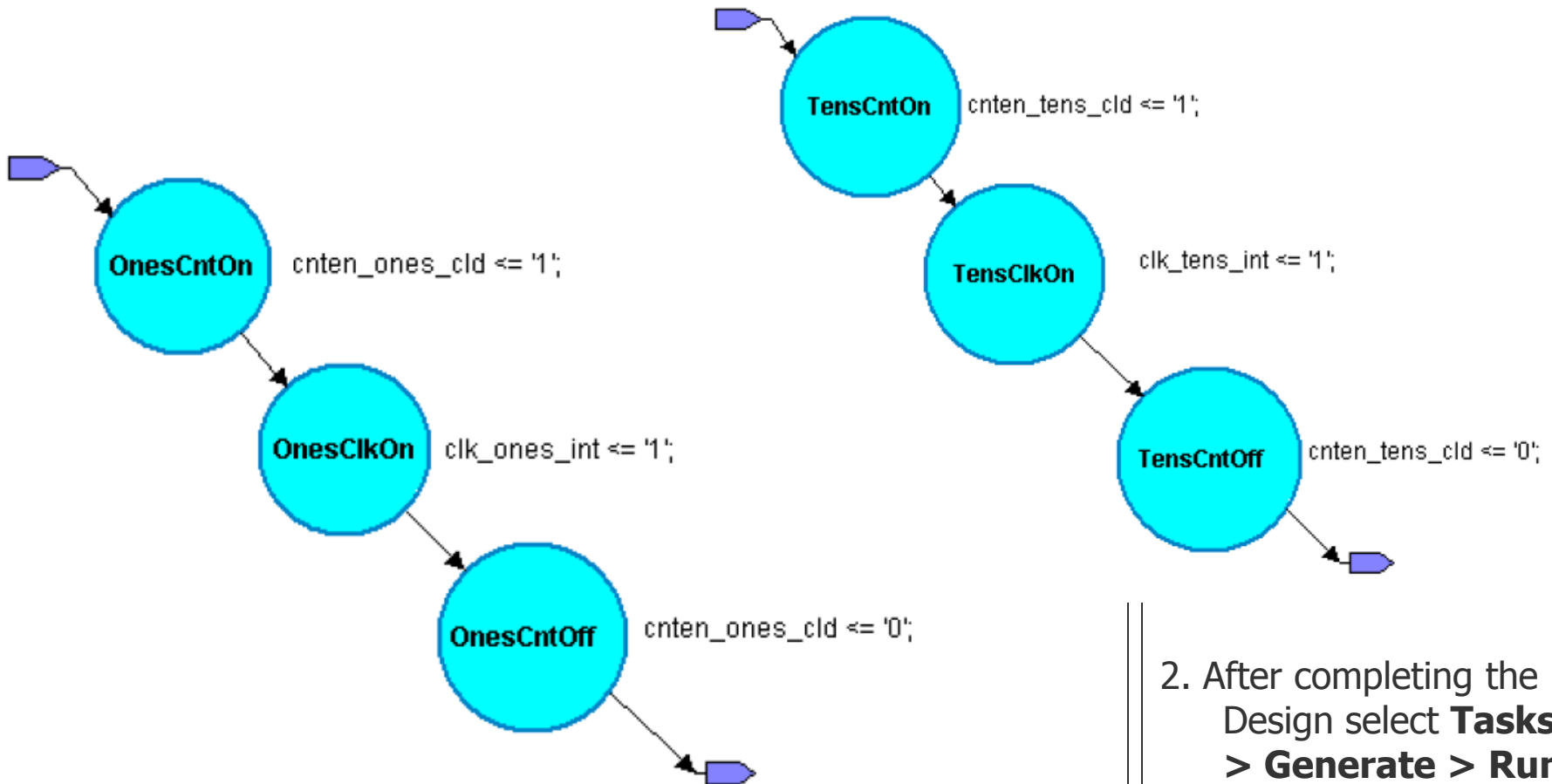
	A	B	C	D	E	F	G	H	I	J	K	L
A	Group	Name	Mode	Type	Bounds	Initial	Category	Assign In	Expression	Scheme	Default	Reset
1		carry_out_ones	IN	std_logic			Data					
2		carry_out_tens	IN	std_logic			Data					
3		clk	IN	std_logic			Clock (Rising)		clk'EVENT AND clk = '1'			
4		ripple_in	IN	std_logic			Data					
5		reset	IN	std_logic			Reset (Async Low)		reset = '0'			
6		q_tens	IN	unsigned	(3:0)		Data					
7		data_in	IN	unsigned	(7:0)		Data					
8		q_ones	IN	unsigned	(3:0)		Data					
9		ripple_out	OUT	std_logic			Data	<auto>		Registered	'0'	'0'
10		load_tens	OUT	std_logic			Data	<auto>		Registered	'0'	'0'
11		load_ones	OUT	std_logic			Data	<auto>		Registered	'0'	'0'
12		load_in_tens	OUT	unsigned	(3:0)		Data	<auto>		Clocked		(others => '0')
13		load_in_ones	OUT	unsigned	(3:0)		Data	<auto>		Clocked		(others => '0')
14		data_out	OUT	unsigned	(7:0)		Data	<auto>		Clocked		(others => '0')
15		cnten_tens	OUT	std_logic			Data	<auto>		Clocked		'0'
16		cnten_ones	OUT	std_logic			Data	<auto>		Clocked		'0'
17		clk_tens	OUT	std_logic			Data	<auto>		Registered	'0'	'0'
18		clk_ones	OUT	std_logic			Data	<auto>		Registered	'0'	'0'

Step 3: FSM Creation



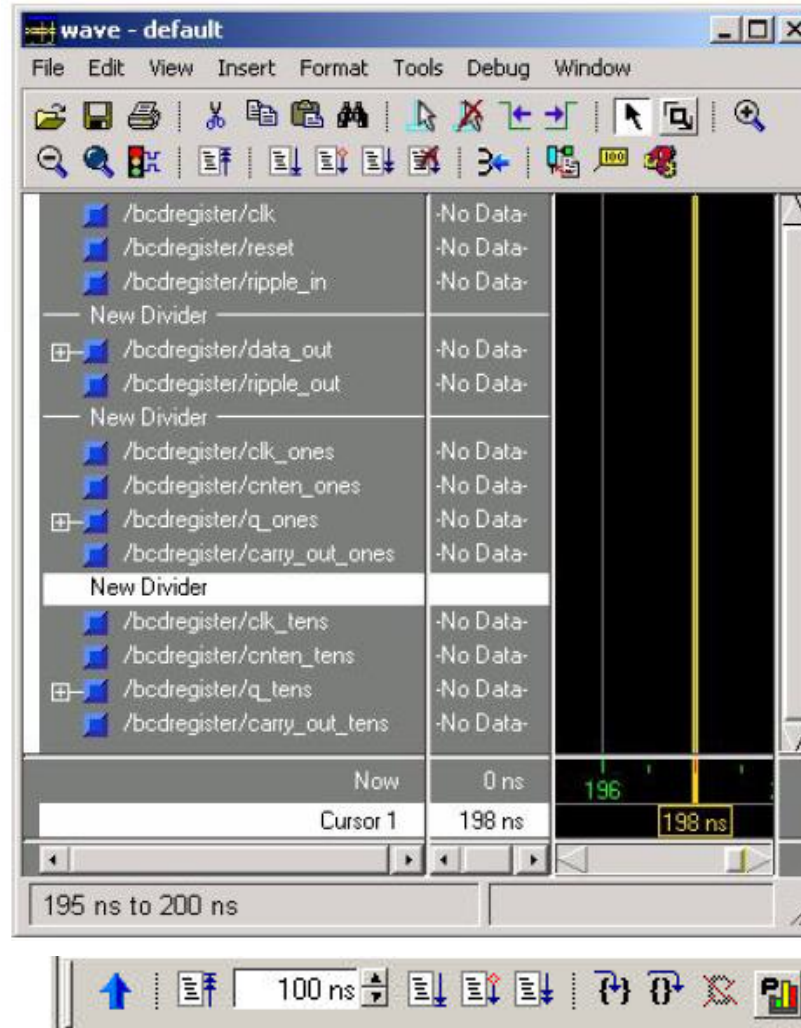
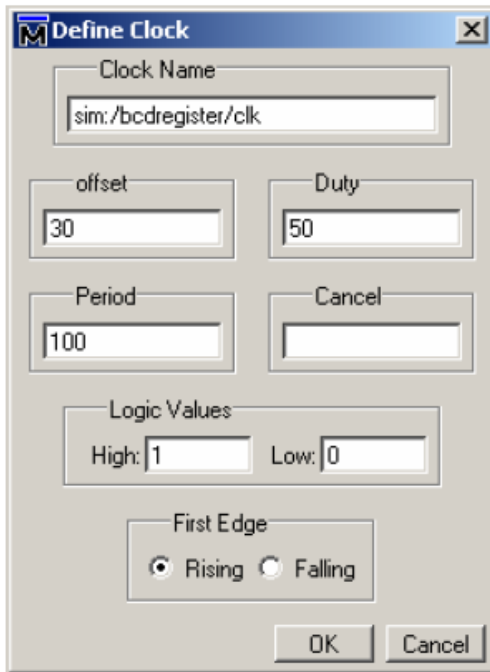
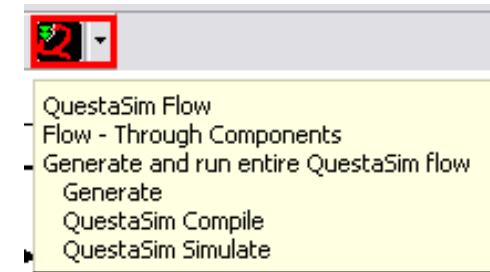
Step 3: FSM Creation

1. Complete the Hierarchal States



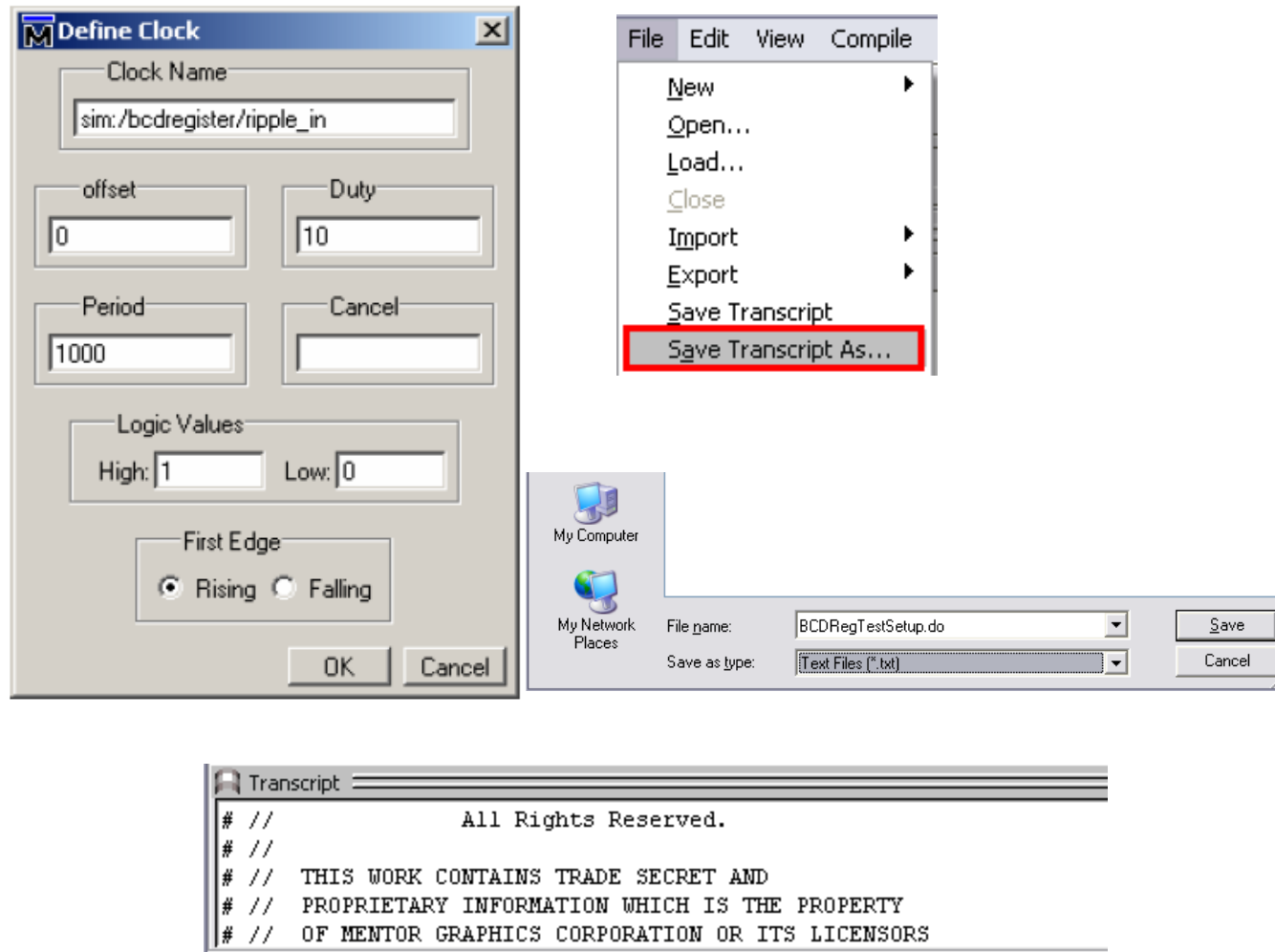
2. After completing the Design select **Tasks > Generate > Run Single**.

Step 4: Testing the BCD Register



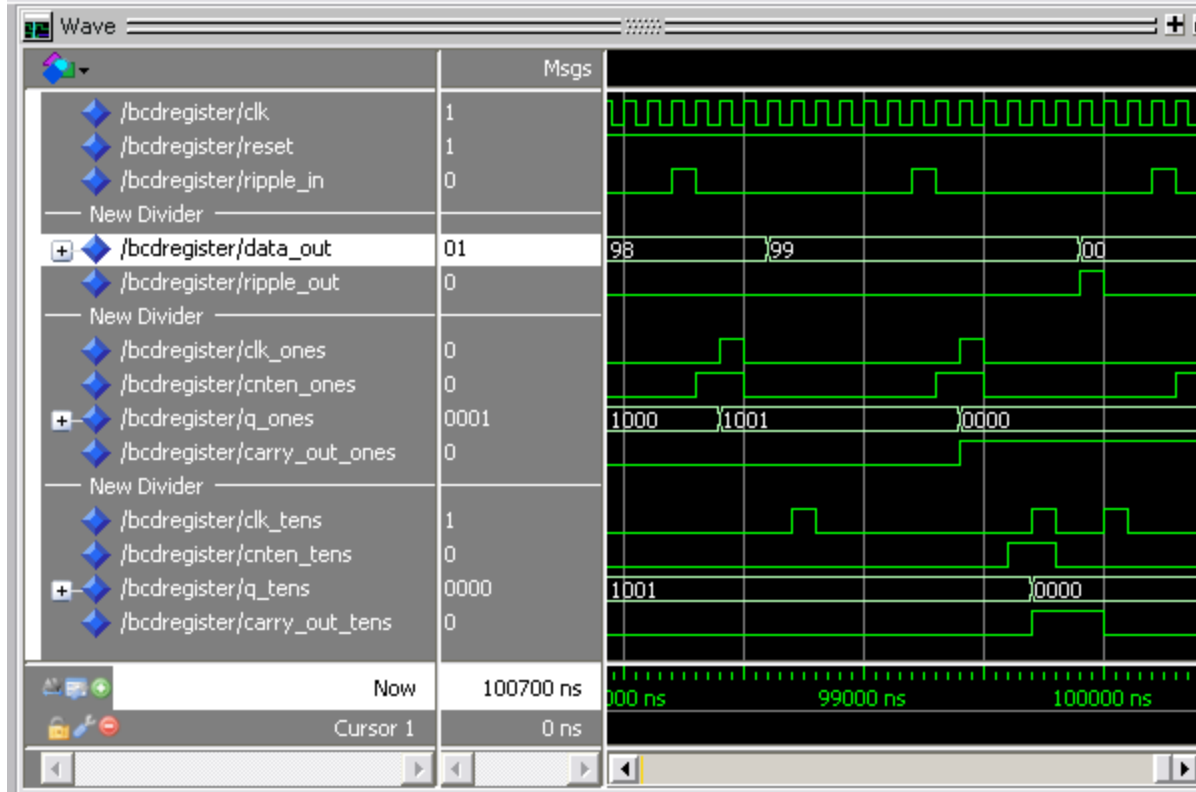
1. Select the BCDRegister block diagram design.
2. Invoke Questasim by clicking the Simulate toolbar icon.
3. You can drag one or multiple signals from the Objects to the Wave window.
4. Set the Run Length variable to 100ns.
5. Select Clk and RMB > Clock, don't change the default value.
6. Force load_en to "1".
7. Force reset to "0" and run the simulation for 100 ns to reset the design.

Step 4: Testing the BCD Register



1. Force *reset* to "1" and run the simulation for another 100ns.
2. Define ripple_in as a clock with the shown settings and run 100 ns.
3. Highlight the Transcript Window, and then use menu item File > Save Transcript As to save the transcript window to the file BCDRegTestSetup.do.
4. Highlight the Wave Window and select menu item File > format to save its formatting in the file wave_BCDReg.do.

Step 4: Testing the BCD Register

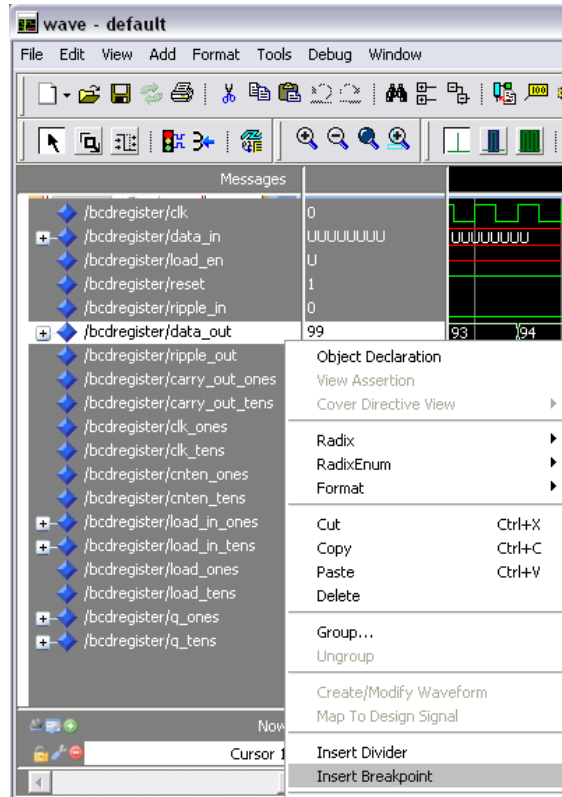


1. Increase the Run Length to 50us.
2. Repeat until you see the Overflow
3. Restart the Simulation using the **Restart icon**.
4. Reset the default Run Length to 100 ns from 50 us
5. Get your saved initialization settings by using the Main pulldown menu item

Tools > TCL > Execute Macro

and then selecting the *BCDRegTestSetup.do* file you saved earlier.

Step 4: Testing the BCD Register

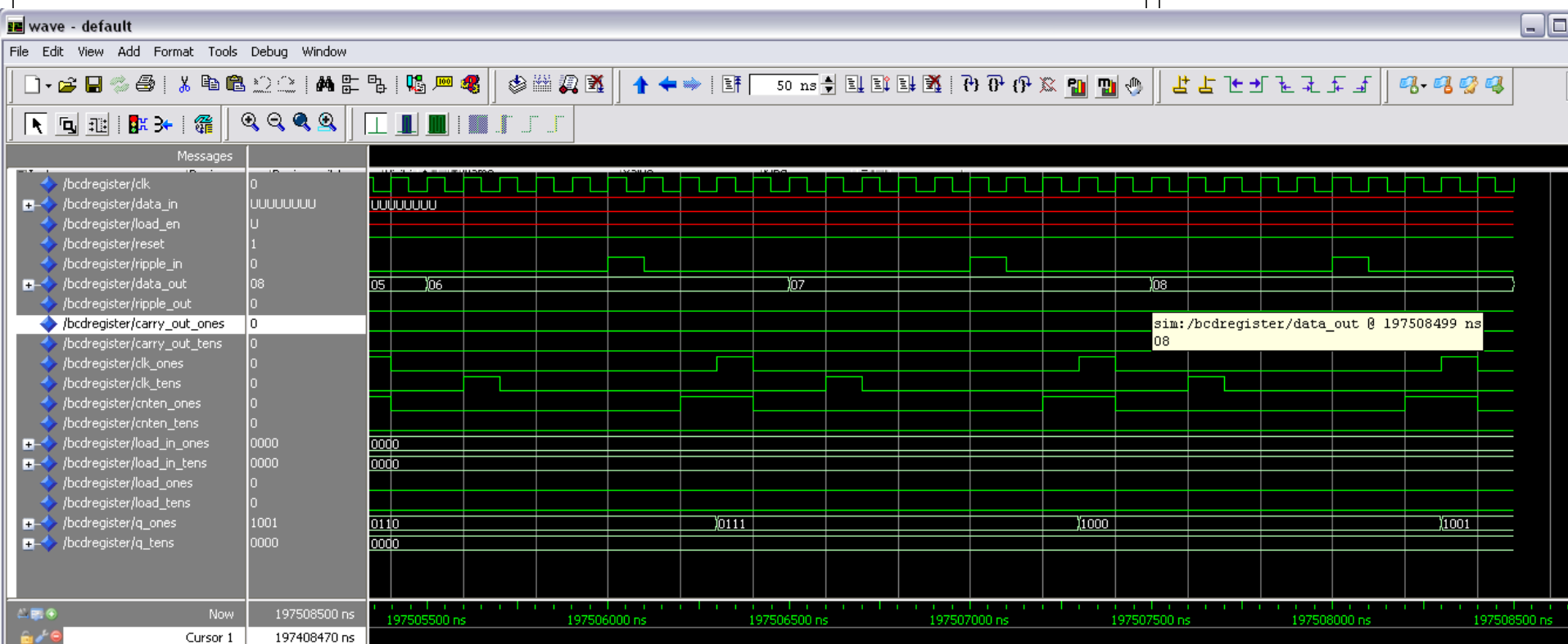


```
Transcript
# 0
# Simulation stop requested.
VSIM 31> run -all
# Break on sim:/bcdregister/data_out
# Simulation stop requested.
VSIM 32> when -label sim:/bcdregister/data_out sim:/bcdregister/data_out=="00001001" {echo {Break on sim:/bcdregister/data_out=="00001001"} ; stop}
VSIM 33> run -all
# Break on sim:/bcdregister/data_out=="00001001"
# 0
# Simulation stop requested.
VSIM 34>
```

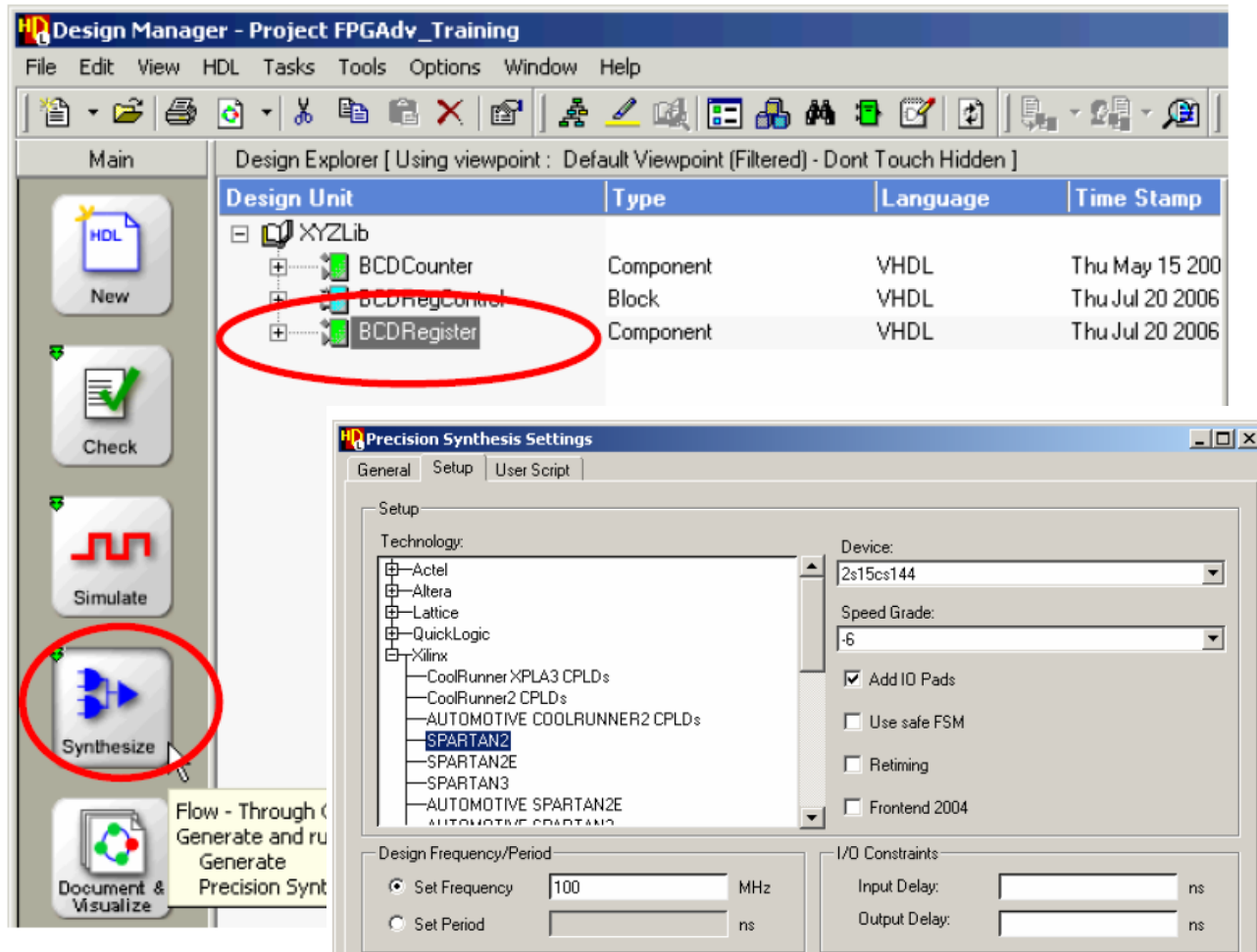
1. set a signal breakpoint on the *data_out* signal by selecting the signal in the Objects window, and issue RMB > Insert Breakpoint.
2. Run the simulation using the run -all command.
3. the simulation stops at the next change of *data_out*.
4. Select the transcript and press up button to get the last commands
5. In the when command change it as shown.

Step 4: Testing the BCD Register

1. Run the simulation then you find that the simulation stops at 09 Hexadecimal.

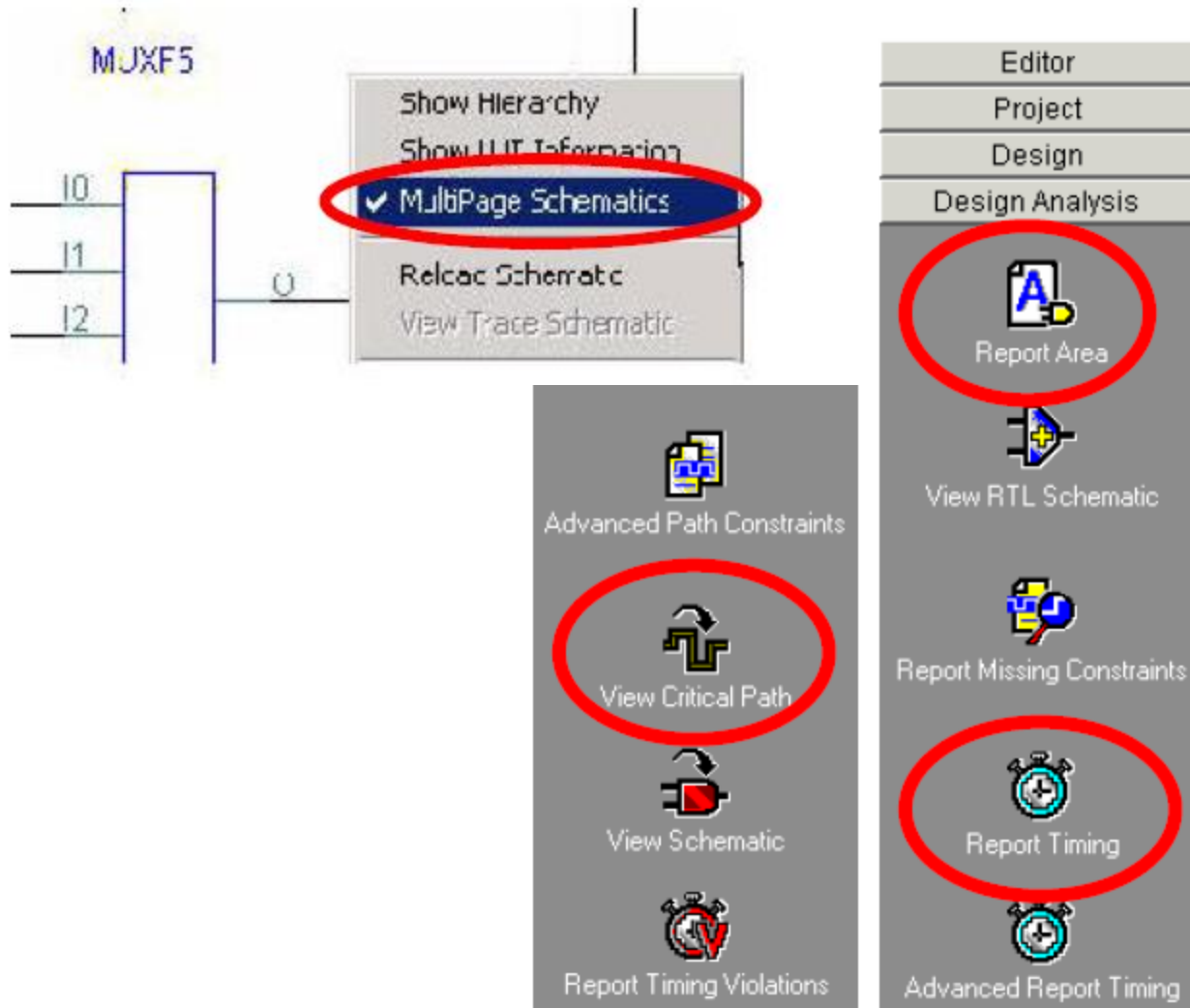


Step 5: Synthesizing the BCD Register



1. Select the BCDRegister design.
2. Click the shortcut bar icon **Synthesize**.
3. Set the Precision Synthesis Settings as shown
4. Double-click the RTL Schematic in the Output Files section of the Project Files panel.
5. Double-click the largest of the blocks, which should be your *BCDRegControl block*.
6. use the Next Page and Previous Page arrows in the Design Bar.

Step 5: Synthesizing the BCD Register



1. You can view the source HDL for the BCDRegControl logic by selecting an instance and clicking **RMB > Trace to HDL Source**.
2. **RMB > Open Up**.
3. To display the whole design in a single schematic deselect the Multiple Page Schematic option in the Schematic Viewer popup window.
4. Open up the Design Analysis section
5. Use the Report Area and the Report Timing icons

Step 6: Creation Testbench

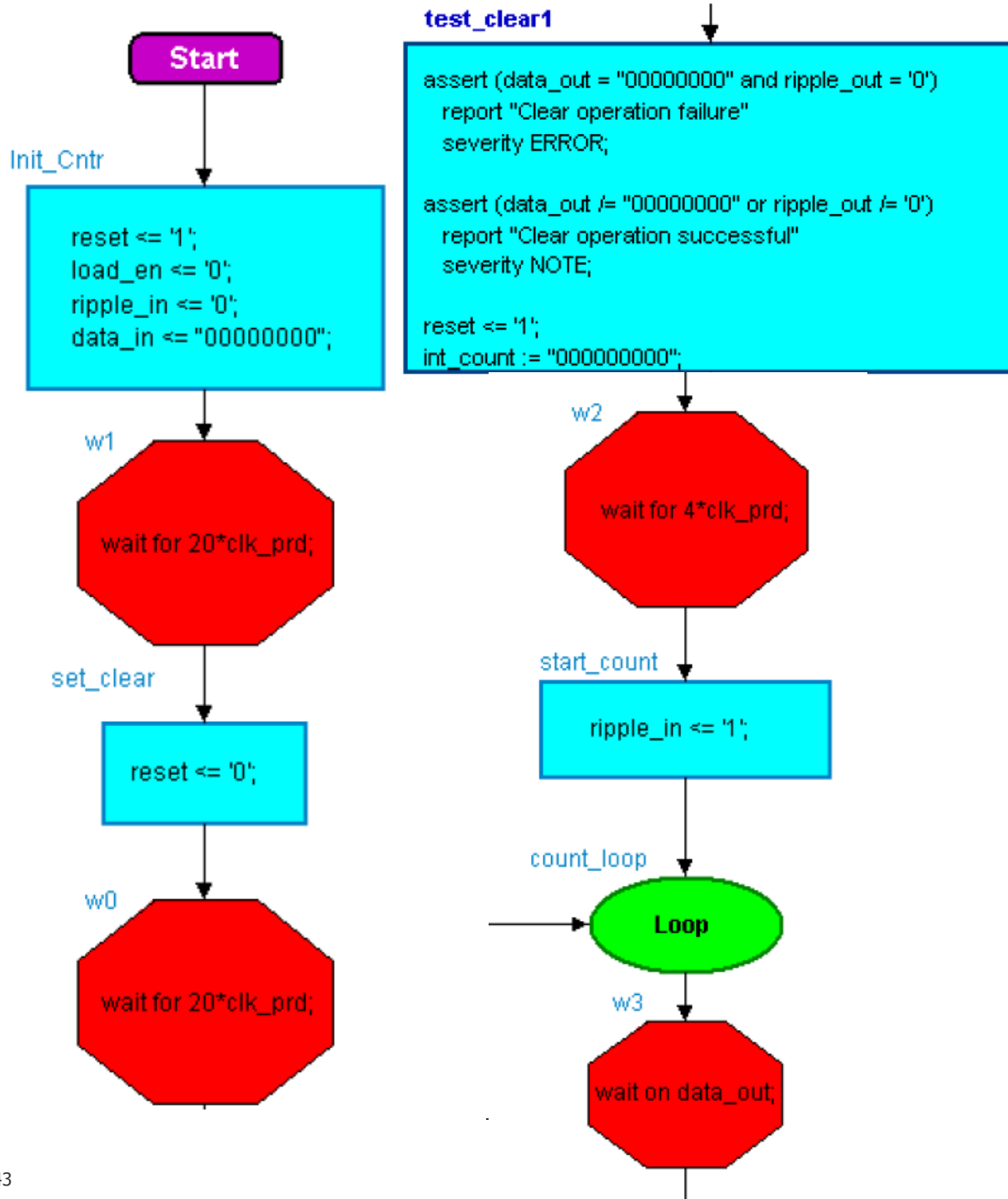
Change
to this

The 'Create Test Bench' dialog box is shown with the following settings:

- Select the type of Test Bench to create :**
 - ☒ Graphical
 - ☐ Textual
- Select the language to use for the Test Bench :**
 - ☒ VHDL
 - ☐ Verilog
- Test Bench library:** MyTestLib (selected in the dropdown)
- Test Bench Design Unit:** BCDRegister_tb
- Test Bench View / Architecture name:** struct
- Tester Block:**
 - ☒ Add Tester Block
 - Name of Tester Block:** BCDRegister_tester
- Unit Under Test:**
 - View / Architecture to be tested:** Default

Buttons at the bottom: OK, Cancel, Help.

1. Select the pulldown menu item **File > New > Test Bench**.
2. Configure the dialog box as shown
3. Click OK in the dialog box.
4. Open the BCDRegister_tb block diagram in the MyTestLib library.
5. Double-click the BCDRegister_tester block to open the Open Down Create New View dialog box.
6. Click Next, and then Finish to close the dialog box.
7. Rename the Flowchart Process to "Tester" using pulldown menu: Diagram > Rename Flow Chart.
8. Implement a flow chart to test your *BCDRegister design*. Use the following diagrams as a guide.



1. First line for test the Reset
2. second line begin with report the functionality of reset

Include the text from the template file *test_clear1.txt*.

3. Enable the ripple_in to start counter running

```

int_count := incbcd_count(int_count);

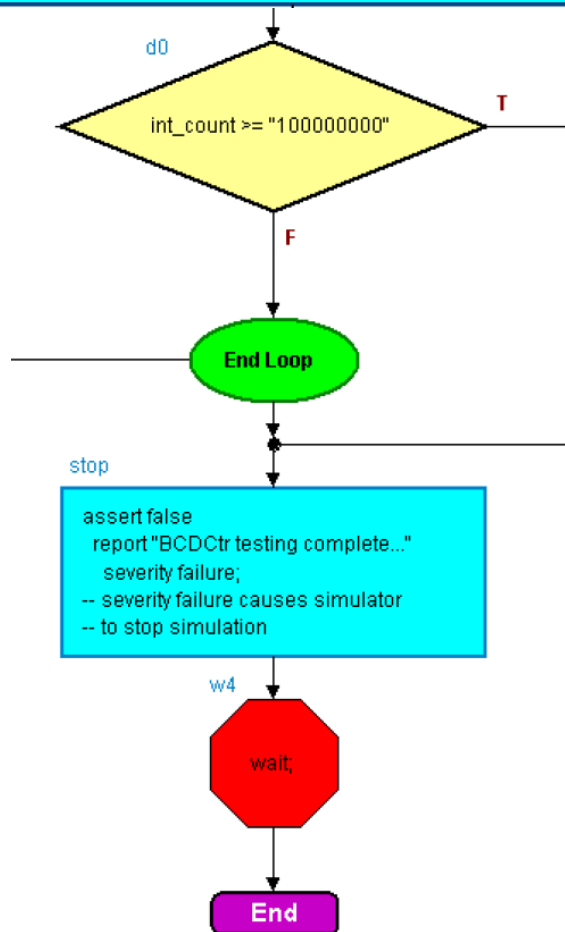
-- To read an assert use the phrase " make sure that the following is satisfied
-- otherwise report that there is a problem
-- eg make sure the int_count(7 downto 0) = data_out AND ripple_out = int_count(8)
-- otherwise the increment will have failed.

assert NOT ( int_count(7 downto 0) /= data_out OR ripple_out /= int_count(8) )
Report "Increment test FAILED"
SEVERITY FAILURE;

assert NOT (data_out = "00010000" AND data_out'LAST_VALUE = "00001001")
Report "09 to 10 Rollover PASSED"
SEVERITY Note;

assert NOT (data_out = "00000000" AND data_out'LAST_VALUE = "10011001")
Report "Rollover PASSED"
SEVERITY Note;

```



1. Check that the output data is correct by comparing with Ref Model.

Include the text from the template file increment_test.txt.

2. Close the loop and end the test bench.

Include function text from the file function_incbcd_count.txt to Process Declarations tab Flowchart Properties

Flow Chart Properties

Generation | Architecture Declarations | Concurrent Statements | **Process Declarations**

Process Declarations:

```

VARIABLE int_count : unsigned (8 DOWNTO 0);

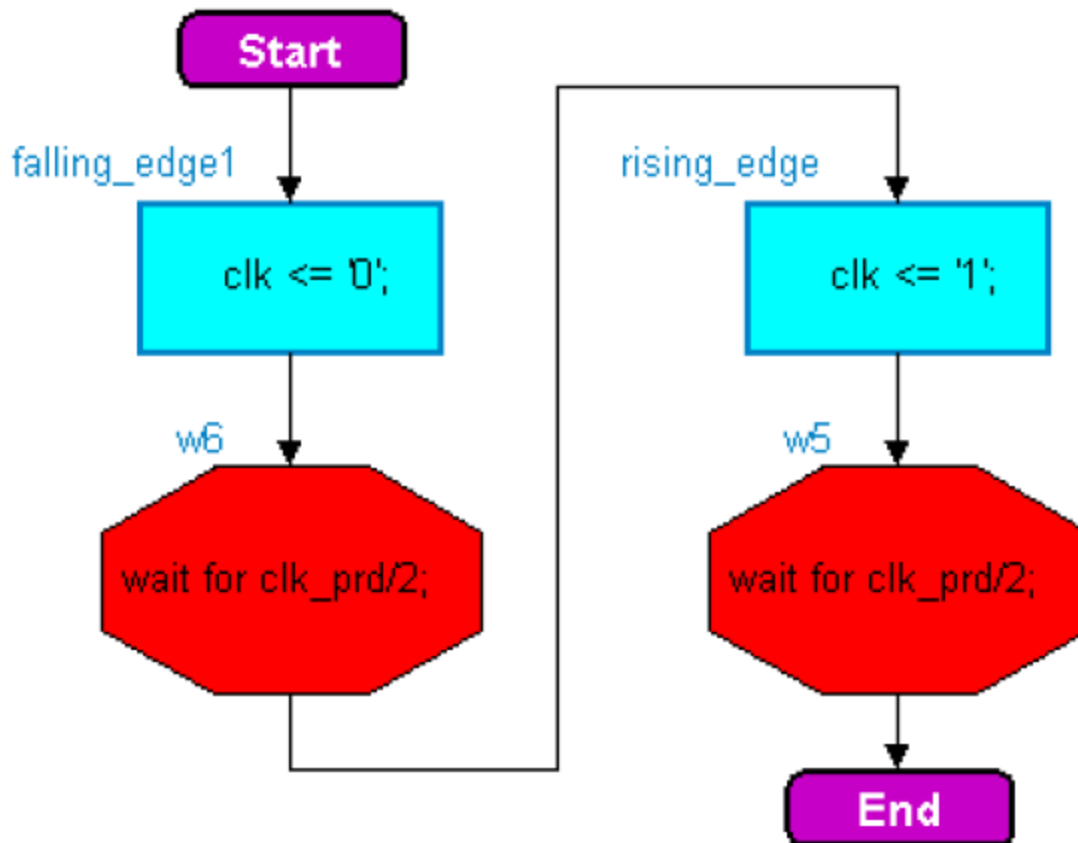
-- The following function counts in BCD

function incbcd_count (A : unsigned (8 downto 0) ) return unsigned is

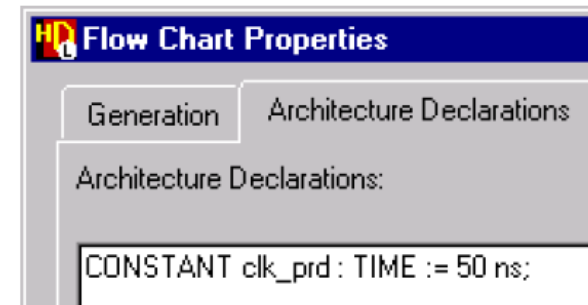
variable vA1 : unsigned (3 downto 0);
variable carry : std_logic;
variable ripple : std_logic;

```

Step 6: Creation Testbench

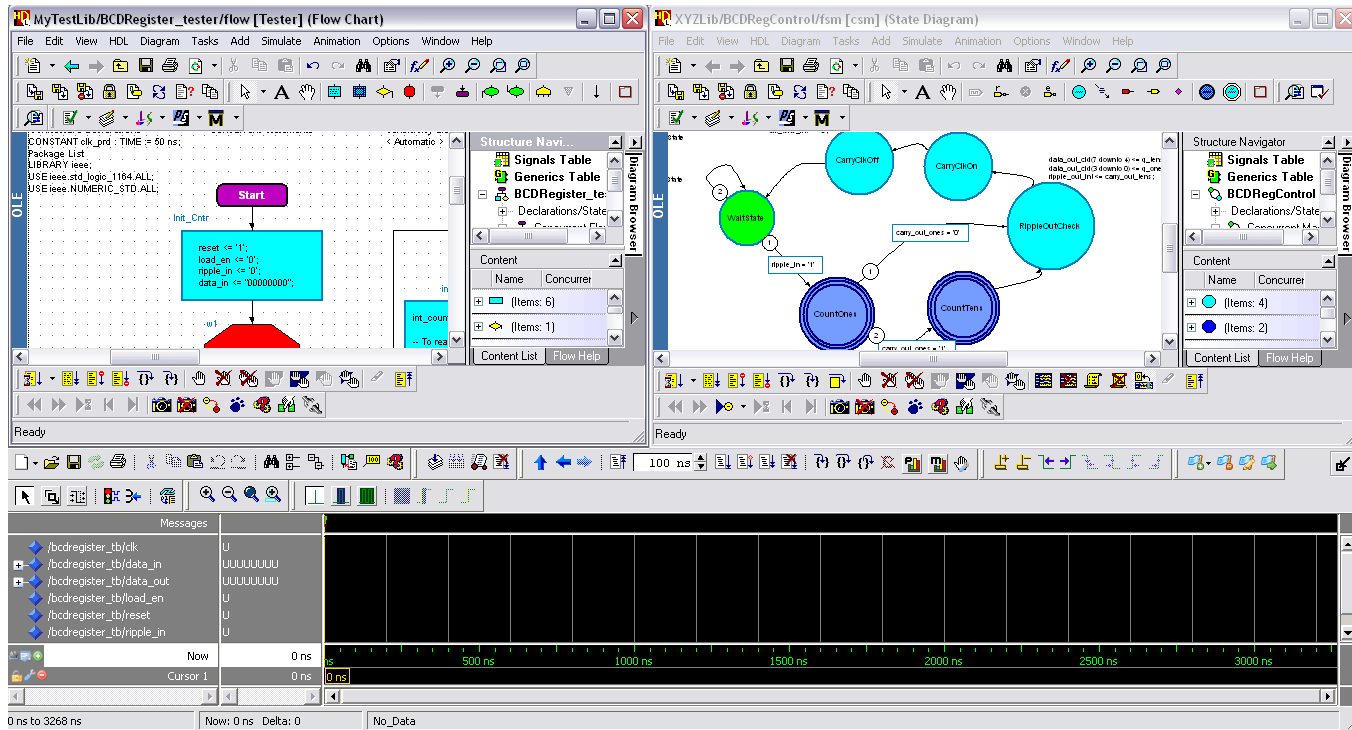


1. Open a concurrent flow chart using the pulldown menu: **Add > Concurrent Flow Chart menu entry.**
2. Declare the Constant `clk_prd` in the Flow Chart Properties dialog box:



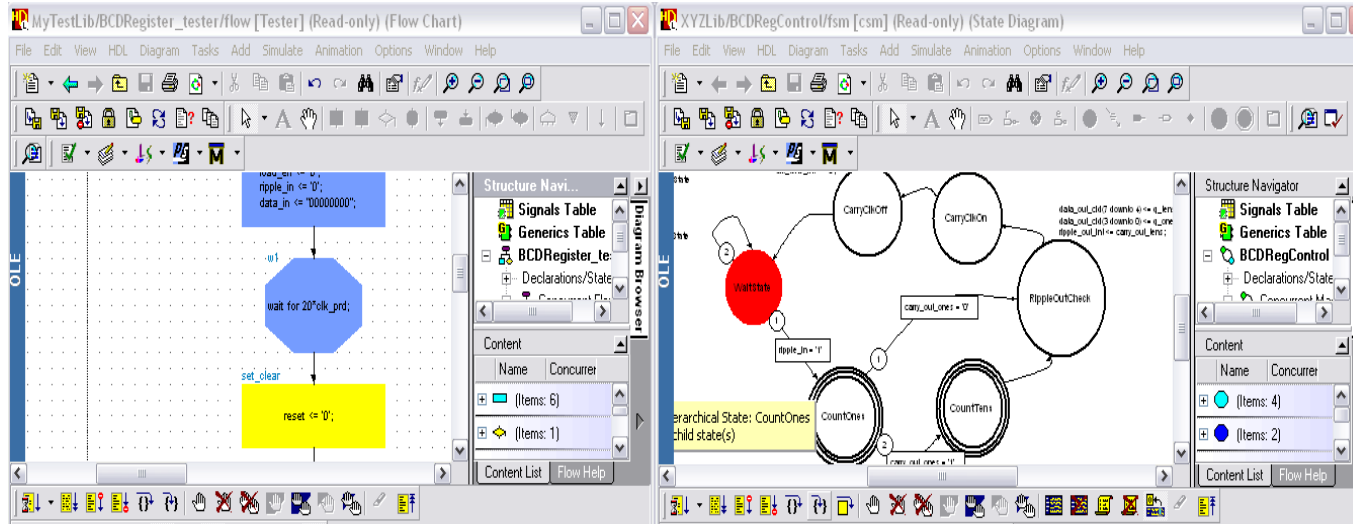
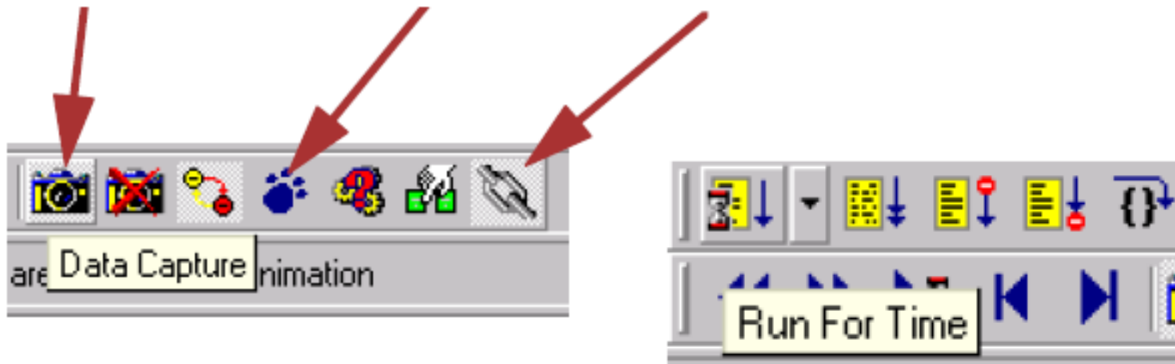
3. Rename the Flowchart Process to "clk_gen"
4. Generate HDL for the BCDRegister_tb design

Step 7: Advanced Simulation Options



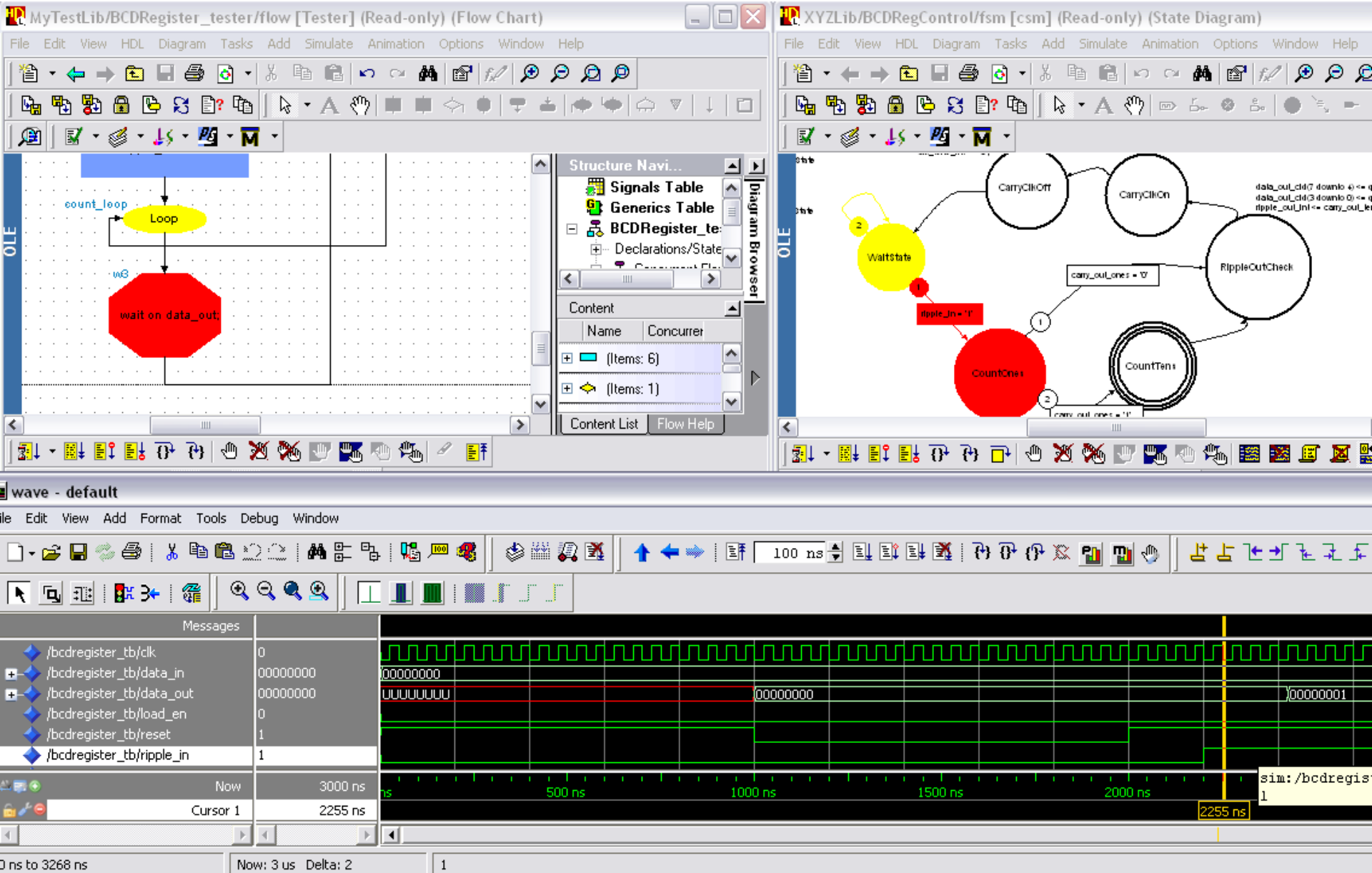
1. Make sure you have enabled Instrument HDL for Animation in the BCDRegister_tester and in the BCDReg_control design.
2. open the BCDRegister_tb Block Diagram.
3. Run the Simulator.
4. Select all signals in the Block Diagram and click the Add Wave
5. Open the Flowchart diagrams and the State diagrams, and arrange them

Step 7: Advanced Simulation Options

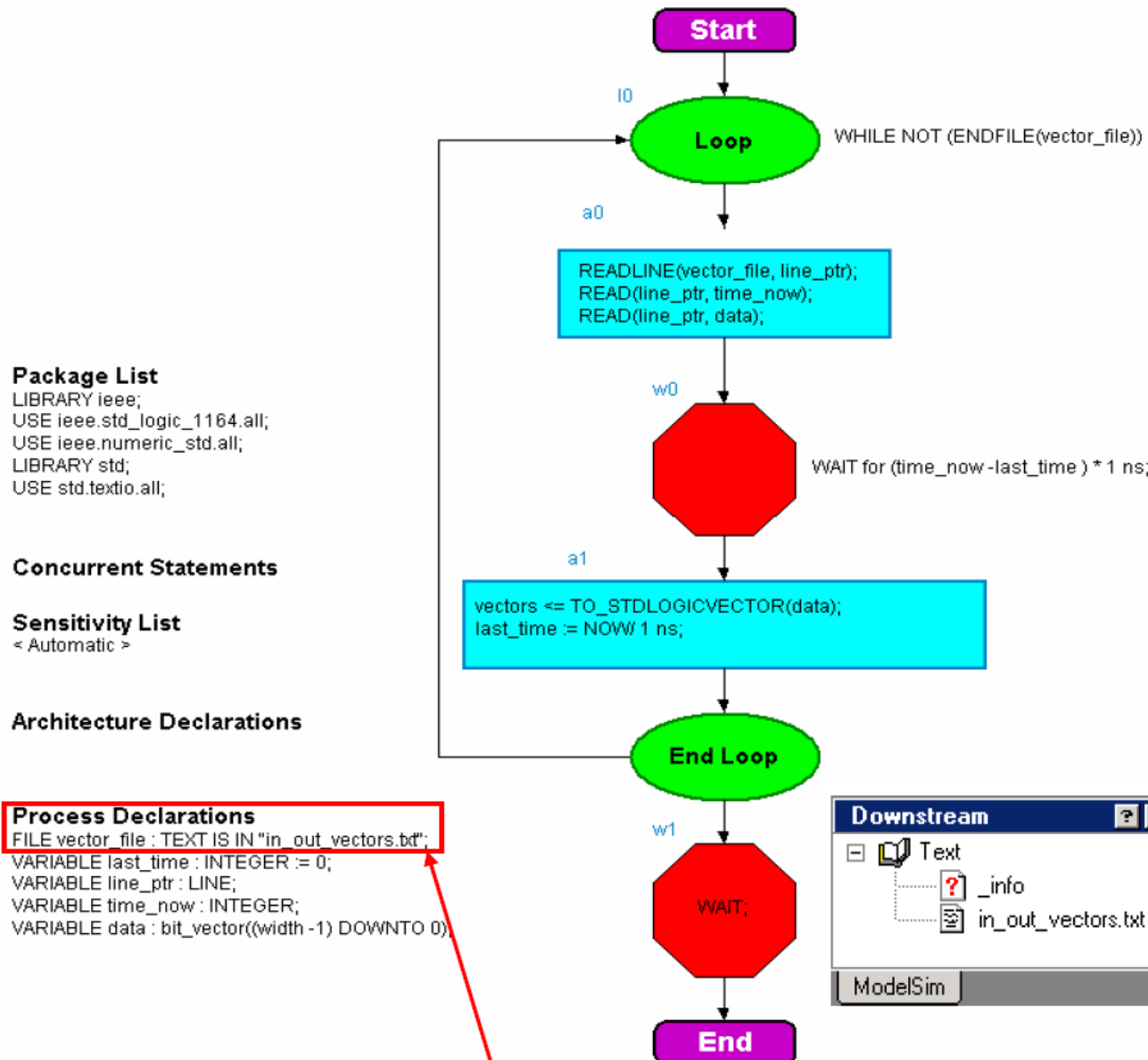


1. Enable DataCapture and Show Animation, Activity Trails (From Start), and link the design windows in both Flowchart and FSM.
2. Run the Simulation from within HDL Designer using the Simulation Control (run for 10000 ns several times) Buttons.
3. Run more than 5 times and use the Cursor to move it in the Wave window and see the effect in the Flow chart and FSM.

Step 7: Advanced Simulation Options

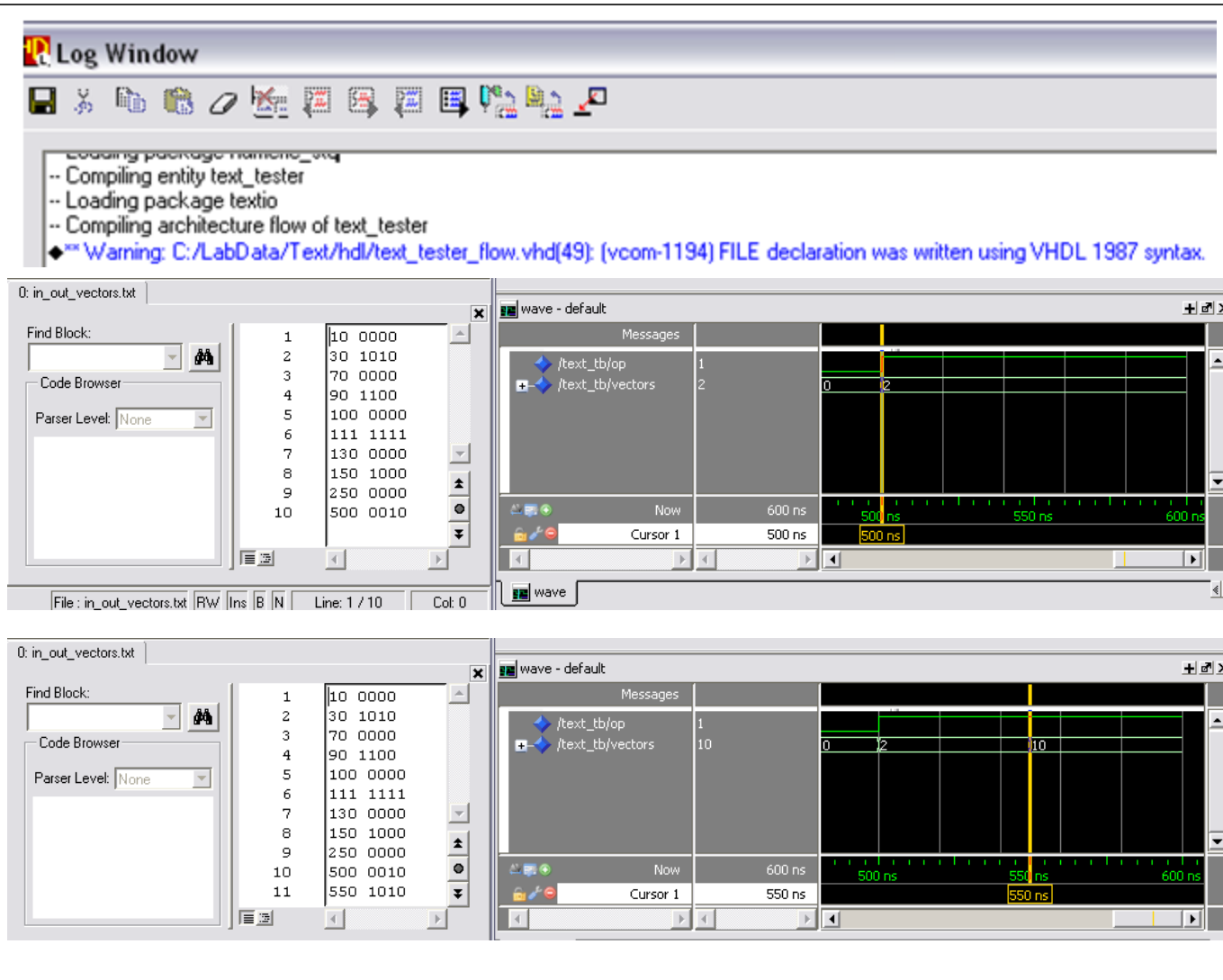


Step 7: Advanced Simulation Options



1. add a new regular Library Mapping named "Text".
2. Open the Library and browse through the three design units.
3. Select the text_tb design unit and compile it.
4. Select the Generate task to generate the HDL for all the design.
5. Open the text_tester design unit.
6. Select text_tester select View > Sub Windows > Side Data/Downstream

Step 7: Advanced Simulation Options



1. Select the QuestaSim Compile task, simulate the text_tb design, and verify the correct behavior of the output.
2. Edit the file in_out_vectors.txt by appending some lines with additional test vectors. Save the file.
3. Restart Simulation.

Questions?

