

Hardware Design Lab

Task 4: Complete CPU

Instructor: Dr. Haiyu Mao

TA:

Zihao Pu

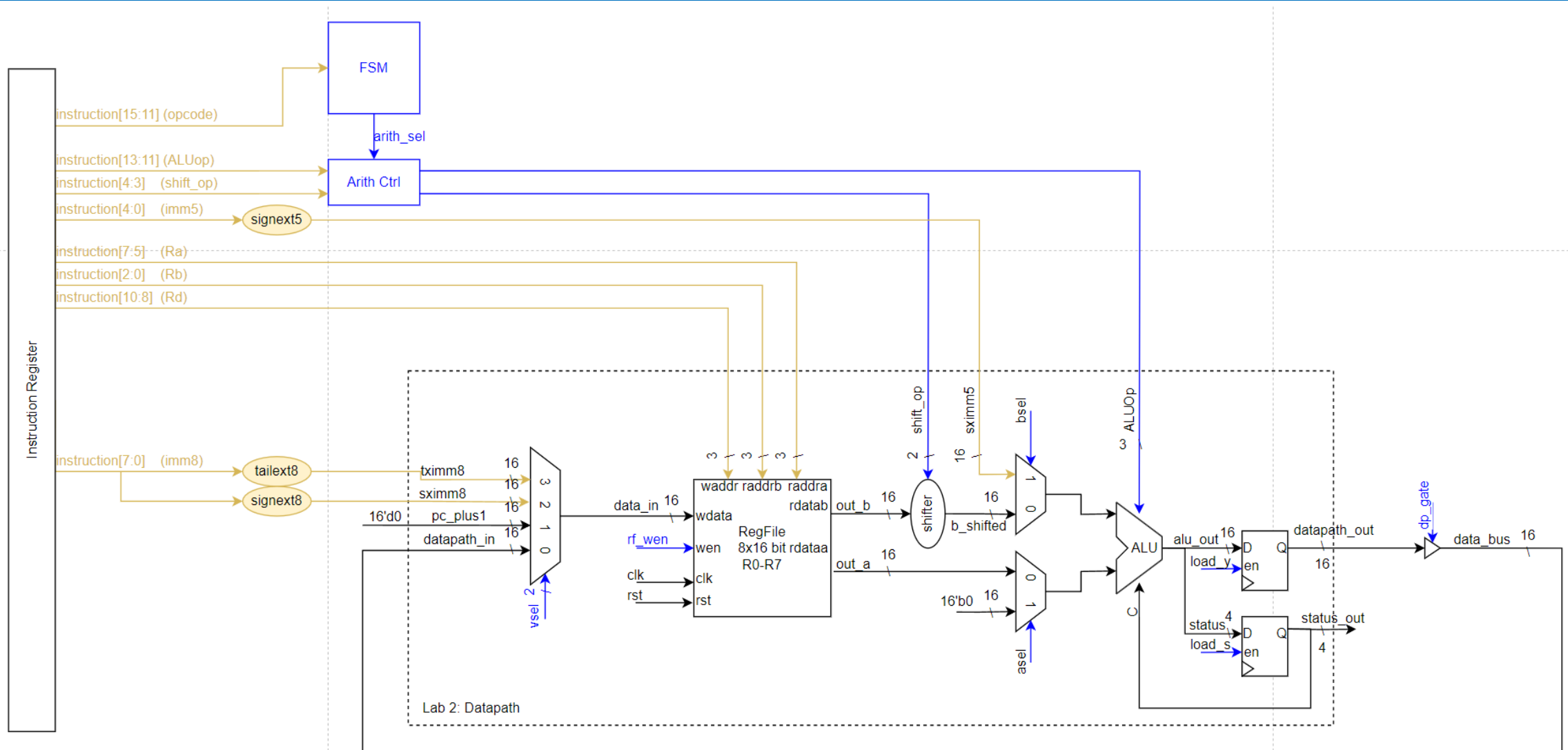
Ali Alsarraf

Stephen Johannesson

Mar 5, 2026

Gratefully acknowledge:
Prof. Onur Mutlu (ETH)
Dr. Yair Linn (TRIUMF Canada)
Prof. Tor Aamodt (UBC)

Lab 3 Recap: What we have accomplished

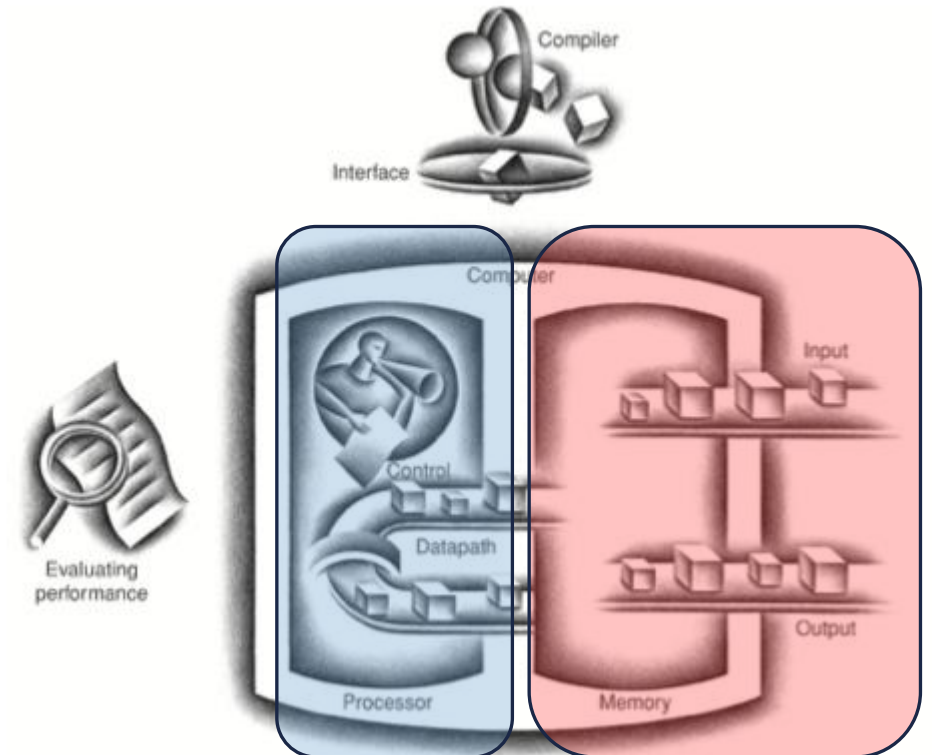


Lab 3 Recap: Supporting Instructions

Mnemonic	Type	OpCode[15:11]		DST [10:8]	SRC1 [7:5]	SHIFT [4:3]	SRC2 [2:0]	Flags	Description & Operation
		ALUOp[13]							
ARITHMETIC		Op[15:14]	:11]	Rd[10:8]	Ra[7:5]	shifto[4:3]	Rb[2:0]		(Shifter applies to Rs2)
ADD	R	00	000	Rd	Ra	Shift	Rb	Z, N, C, V	$Rd = Ra + \text{Shift}(Rb)$
SUB	R	00	001	Rd	Ra	Shift	Rb	Z, N, C, V	$Rd = Ra - \text{Shift}(Rb)$
ADDC	R	00	010	Rd	Ra	Shift	Rb	Z, N, C, V	Add with Carry, $Rd = Ra + \text{Shift}(Rb) + C$
SUBC	R	00	011	Rd	Ra	Shift	Rb	Z, N, C, V	Sub with Carry, $Rd = Ra - \text{Shift}(Rb) - C$
AND	R	00	100	Rd	Ra	Shift	Rb	Z, N	$Rd = Ra \& \text{Shift}(Rb)$
ANDBB	R	00	101	Rd	Ra	Shift	Rb	Z, N	$Rd = Ra \& \text{Bar}(\text{Shift}(Rb))$
OR	R	00	110	Rd	Ra	Shift	Rb	Z, N	$Rd = Ra \text{Shift}(Rb)$
ORBB	R	00	111	Rd	Ra	Shift	Rb	Z, N	$Rd = Ra \text{Bar}(\text{Shift}(Rb))$
		ALUOp[13]							
IMMEDIATE		Op[15:14]	:11]	Rd[10:8]	Ra[7:5]	IMM5[4:0]		Flags	(Merges Shift & Src2 into Imm5)
ADDI	I	01	000	Rd	Ra	sximm5		Z, N, C, V	$Rd = Ra + \text{SignExt}(\text{Imm5})$
SUBI	I	01	001	Rd	Ra	sximm5		Z, N, C, V	$Rd = Ra - \text{SignExt}(\text{Imm5})$
		OpCode[15:11]		DST[10:8]	IMM8[7:0]				
LDI		10000		Rd	sximm8			-	$Rd = \text{SignExt}(\text{Imm8})$
LUI		10001		Rd	sximm8			-	$Rd = \text{TailExt}(\text{Imm8})$

Recap on CPU: A Factory of Data

- ❑ What we have done: The processor (controller and datapath)
- ❑ What we are going to accomplish: Memory, IO, and advanced controller
- ❑ For now, we support sequential arithmetic instructions and load register instructions.
- ❑ What we need to do:
 - To support instruction and data memory.
 - Use PC for automated execution.
 - Support conditional execution: jumping and branching



Recap: Coursework Goal

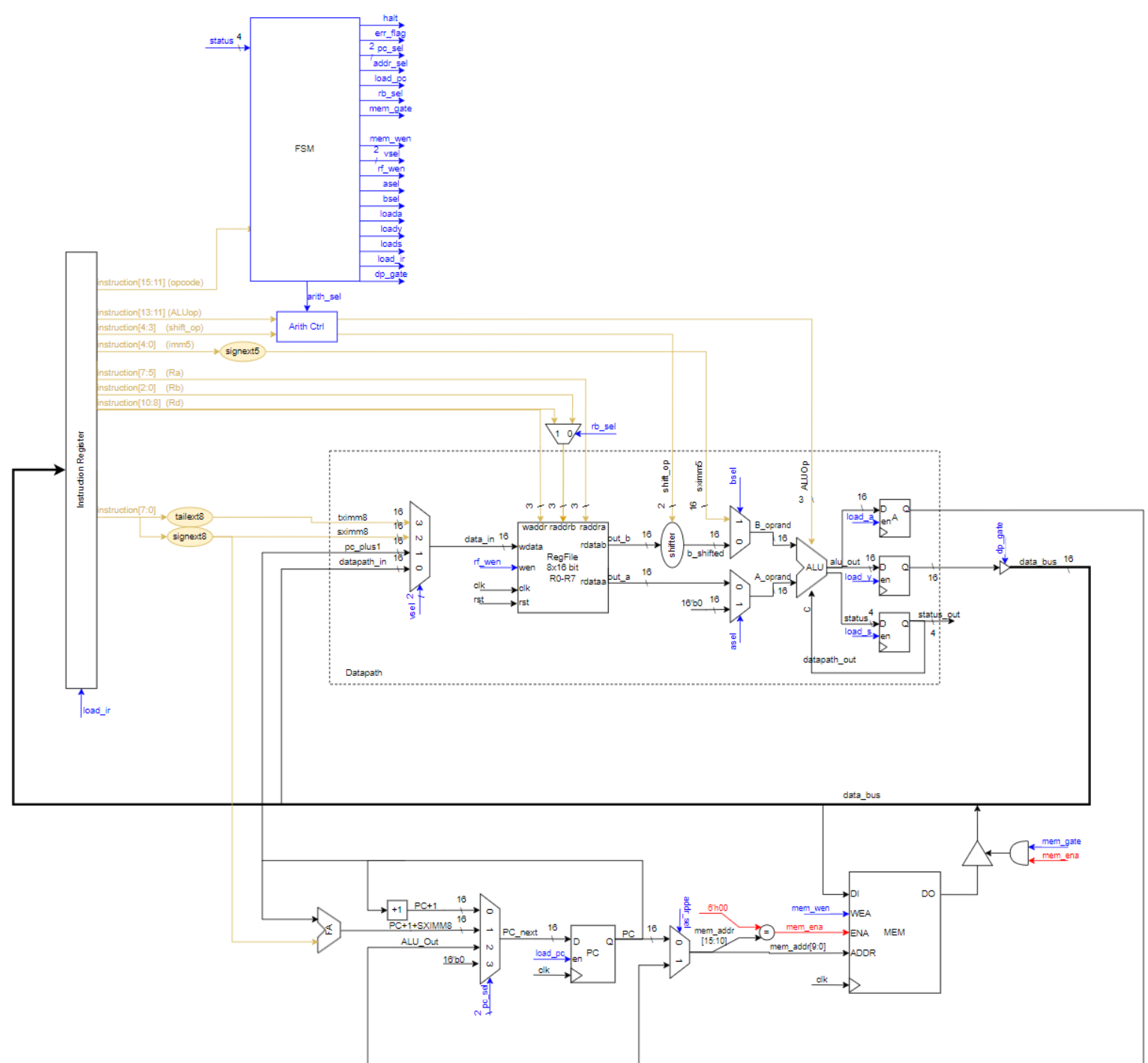
- ❑ 1. Complete the CPU design by filling in the template project
- ❑ 2. Prepare your own program, execute it and show the demo in VIVA
- ❑ 3. Register for VIVA timeslot: [HAD26LAB.xlsx](#)
- ❑ 4. Write your report

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Week 6																				
2	Slot1	Slot 2	Slot3	Slot4	Slot5	Slot6	Slot7	Slot8	Slot9	Slot10	Slot11	Slot12	Slot13	Slot14	Slot15	Slot16	Slot17	Slot18	Slot19	Slot20	
3	Zihao																				
4	Ali																				
5	Stefan																				
6																					
7																					
8																					
9																					
10	Week 7																				
11	Slot1	Slot 2	Slot3	Slot4	Slot5	Slot6	Slot7	Slot8	Slot9	Slot10	Slot11	Slot12	Slot13	Slot14	Slot15	Slot16	Slot17	Slot18	Slot19	Slot20	
12	Zihao																				
13	Ali																				
14	Stefan																				
15																					

Coursework Checklist

- ❑ Finish CPU design
- ❑ Verify with [Verilog Autograder With Waveforms](#)
- ❑ Write your own assembly program
 - Step 1: refer to case 1 and case 2, write your own cases
 - Step 2: use assembler to compile assembly program to binaries
 - Step 3: prepare your own mem_ref.txt and reg_ref.hex, based on your program. These files show the final results in memory and registers.
- ❑ Make a testbench with your program
 - Step 4: in cpu_tb.sv, add your own program in the test cases. Refer to existing cases
- ❑ Download to BASYS3 for Demo
 - Step 5: in Vivado, add your compiled .coe file to memory IP, generate bitstream and download to BASYS3
- ❑ Write your report
- ❑ Wrap them up into a zip file, then submit to KEATS

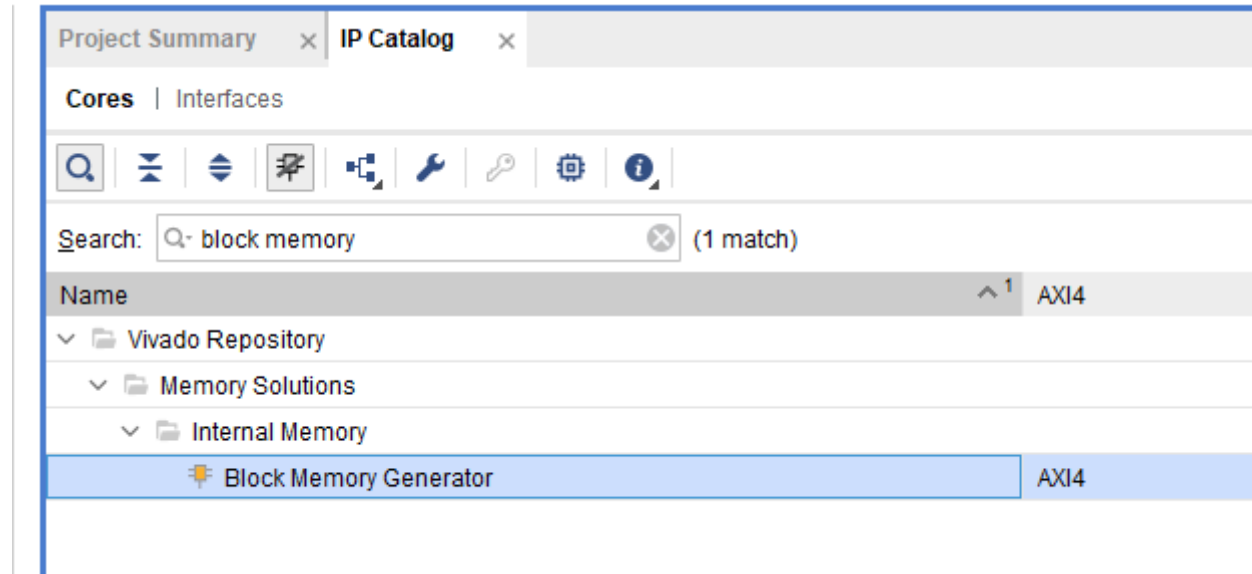
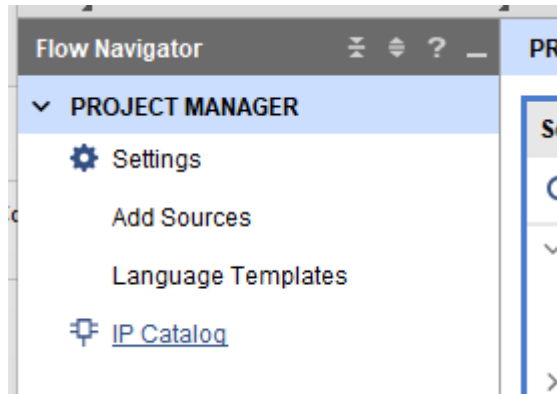
Complete Architecture: Overview



Memory

WHERE PROGRAM AND DATA STORE

Memory Module: Using Vivado Block Memory



Memory Module: Using Vivado Block Memory

Customize IP

Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

Show disabled ports

BRAM_PORTA

- addra[3:0]
- clka
- dina[17:0]
- douta[17:0]
- ena
- wea[0:0]

Component Name: blk_mem_gen_1

Basic Port A Options Other Options Summary

Interface Type: Native Generate address interface with 32 bits

Memory Type: Single Port RAM Common Clock

ECC Options

ECC Type: No ECC

Error Injection Pins: Single Bit Error Injection

Write Enable

Byte Write Enable

Byte Size (bits): 9

Algorithm Options

Defines the algorithm used to concatenate the block RAM primitives. Refer datasheet for more information.

Algorithm: Minimum Area

Primitive: 8kx2

OK Cancel

- ▼ RISKing16
 - > .git
 - > ip
 - > prj
 - > sim
 - > src

Memory Module: Using Vivado Block Memory

The screenshot displays the 'Customize IP' window for the 'Block Memory Generator (8.4)'. The 'Other Options' tab is selected, showing the following configuration details:

- Component Name:** blk_mem_gen_1
- Memory Size:**
 - Write Width: 16 (Range: 1 to 4608 (bits))
 - Read Width: 16
 - Write Depth: 1024 (Range: 2 to 1048576)
 - Read Depth: 1024
- Operating Mode:** Write First
- Enable Port Type:** Use ENA Pin
- Port A Optional Output Registers:**
 - Primitives Output Register
 - Core Output Register
 - SoftECC Input Register
 - REGCEA Pin
- Port A Output Reset Options:**
 - RSTA Pin (set/reset pin) | Output Reset Value (Hex): 0
 - Reset Memory Latch | Reset Priority: CE (Latch or Register Enable)
- READ Address Change A:**
 - Read Address Change A

On the left side of the dialog, the 'IP Symbol' tab is active, showing a block diagram of the 'BRAM_PORTA' component with the following ports:

- addr_A[9:0]
- clk_A
- dina_A[15:0]
- dout_A[15:0]
- ena
- wea_A[0:0]

Memory Module: Using Vivado Block Memory

- **Write First Mode:** In WRITE_FIRST mode, the input data is simultaneously written into memory and driven on the data output, as shown in Figure 3-9. This transparent mode offers the flexibility of using the data output bus during a Write operation on the same port.

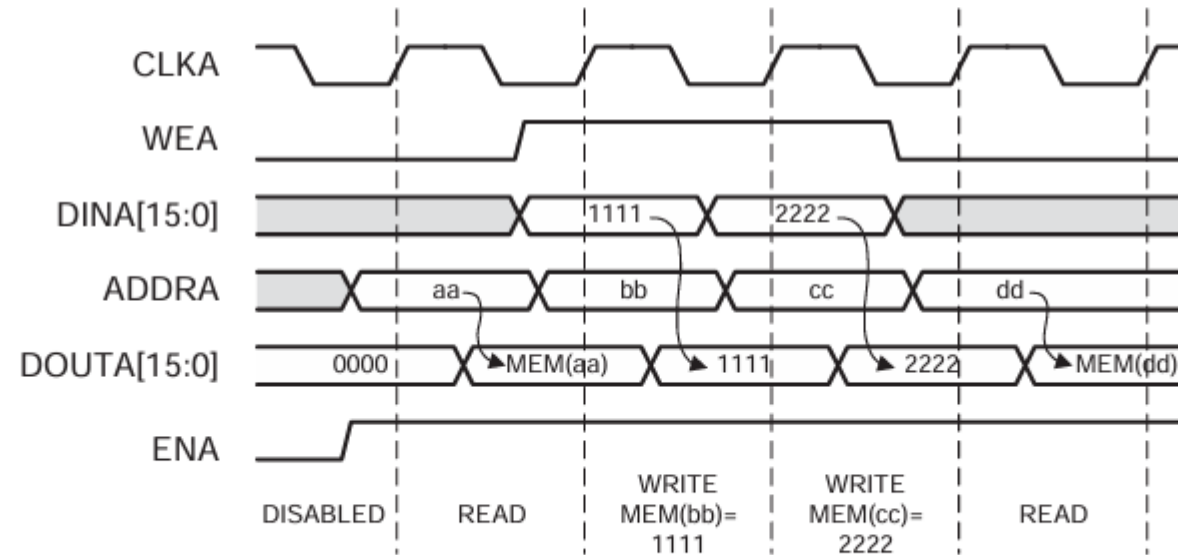


Figure 3-9: Write First Mode Example

<https://docs.amd.com/v/u/en-US/pg058-blk-mem-gen>

Memory Module: Using Vivado Block Memory

Customize IP

Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

Show disabled ports

Component Name: blk_mem_gen_1

Basic Port A Options Other Options Summary

Pipeline Stages within Mux: 0 Mux Size: 1x1

Memory Initialization

Load Init File

Coe File: D:/Code/HAD26Lab/RISKing16/sim/mem_init.coe

Fill Remaining Memory Locations

Remaining Memory Locations (Hex): FFFF

Structural/UniSim Simulation Model Options

Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.



Collision Warnings: All

Behavioral Simulation Model Options

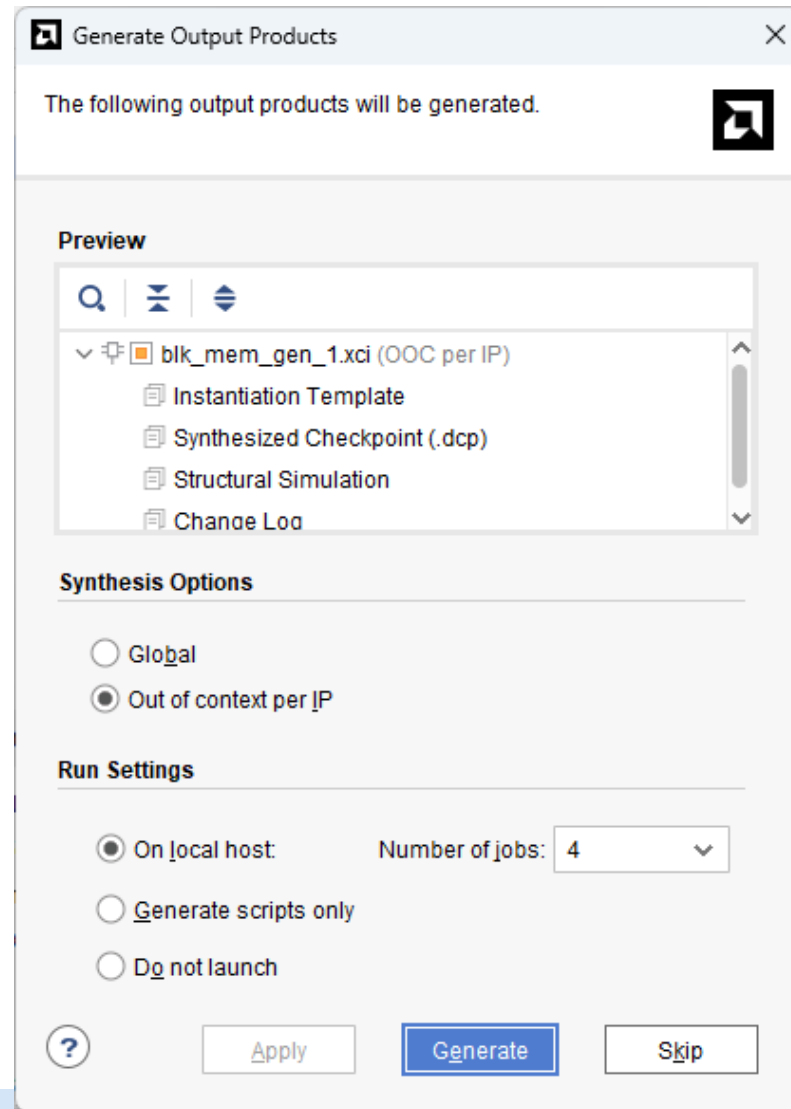
Disable Collision Warnings Disable Out of Range Warnings

OK Cancel

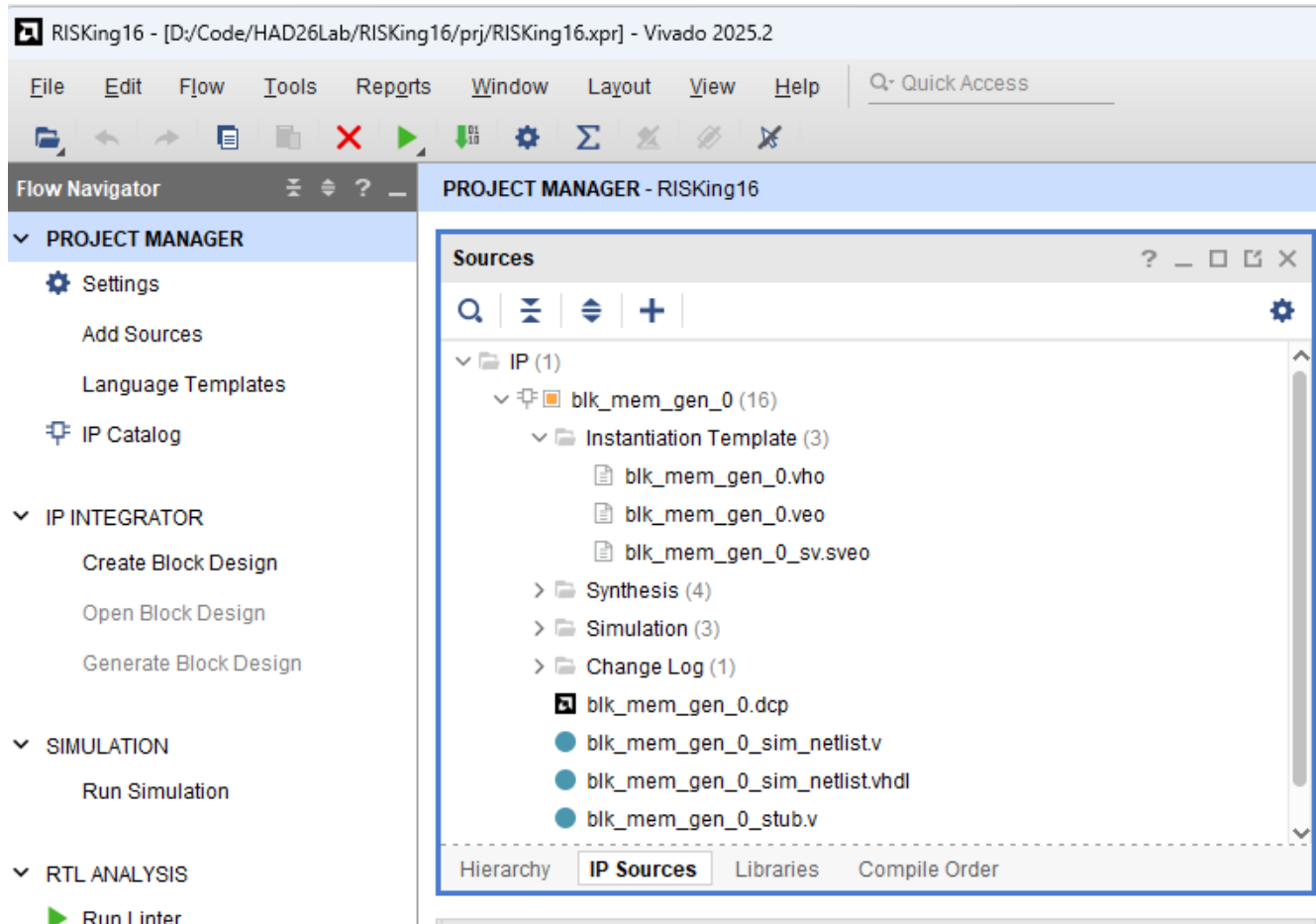
Memory Module: Using Vivado Block Memory

```
sim >  mem_init.coe  
1  memory_initialization_radix=16;  
2  memory_initialization_vector=  
3  810A,  
4  820B,  
5  C007,  
6  FFFF,  
7  FFFF,  
 8  FFFF,  
9  FFFF,  
10 FFFF,  
11 FFFF,  
12 FFFF,  
13 0022,  
14 FFFF,  
15 FFFF,  
16 FFFF,
```

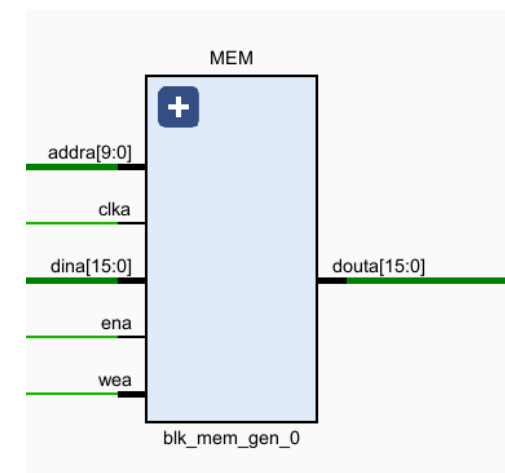
Memory Module: Using Vivado Block Memory



Memory Module: Using Vivado Block Memory



```
55
56 //----- Begin Cut here for INSTANTIATION Template ---// INST_TAG
57 blk_mem_gen_0 your_instance_name (
58     .clka(clka), // input wire clka
59     .ena(ena), // input wire ena
60     .wea(wea), // input wire [0 : 0] wea
61     .addra(addra), // input wire [9 : 0] addra
62     .dina(dina), // input wire [15 : 0] dina
63     .douta(douta) // output wire [15 : 0] douta
64 );
```



Program Counter and Memory

