Hardware Design Lab

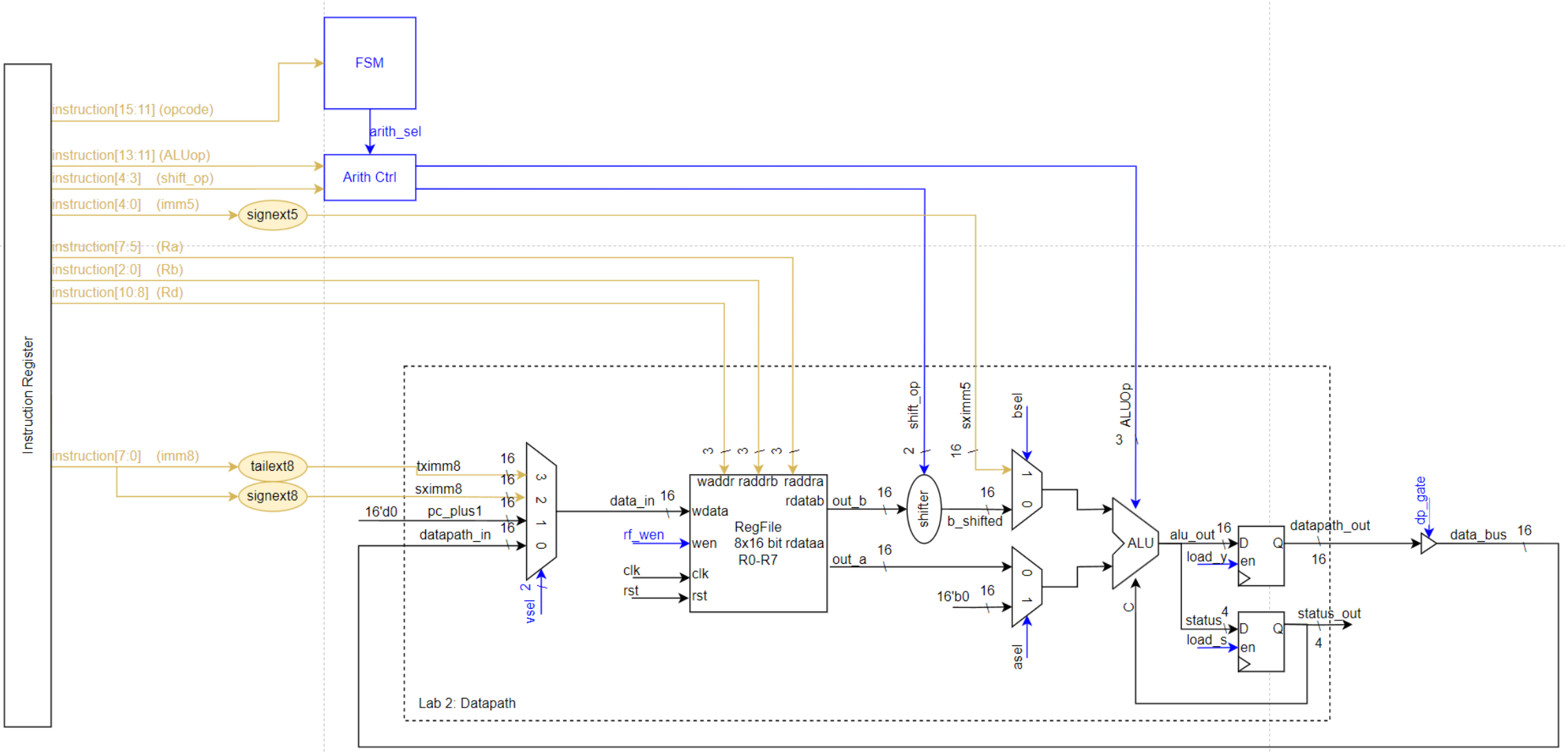
Task 4: Complete CPU

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

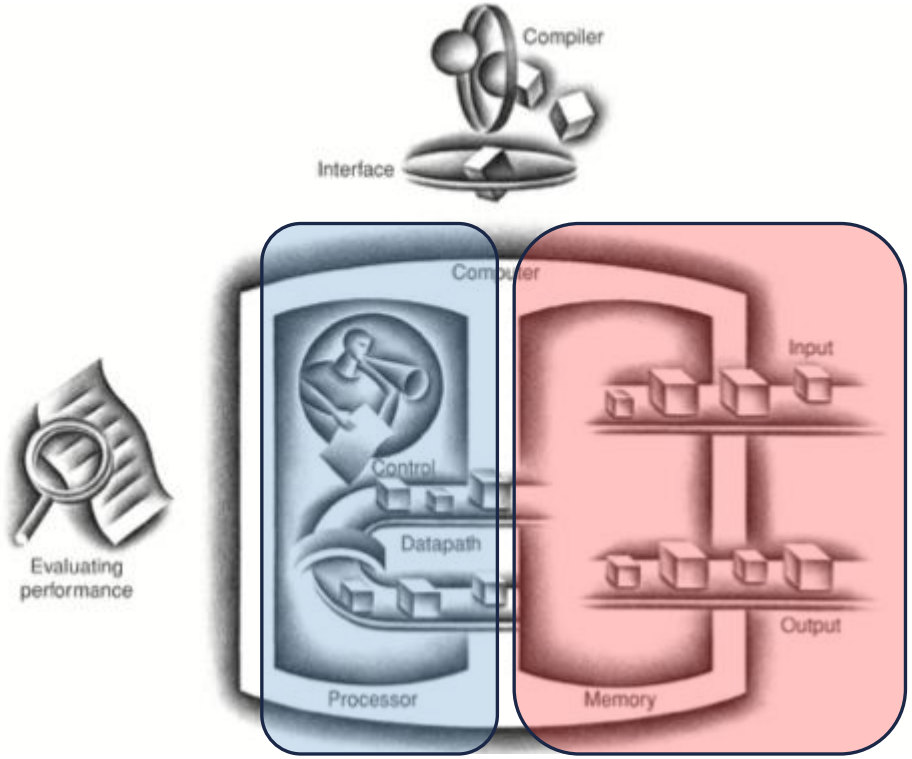


Lab 3 Recap: Supporting Instructions

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

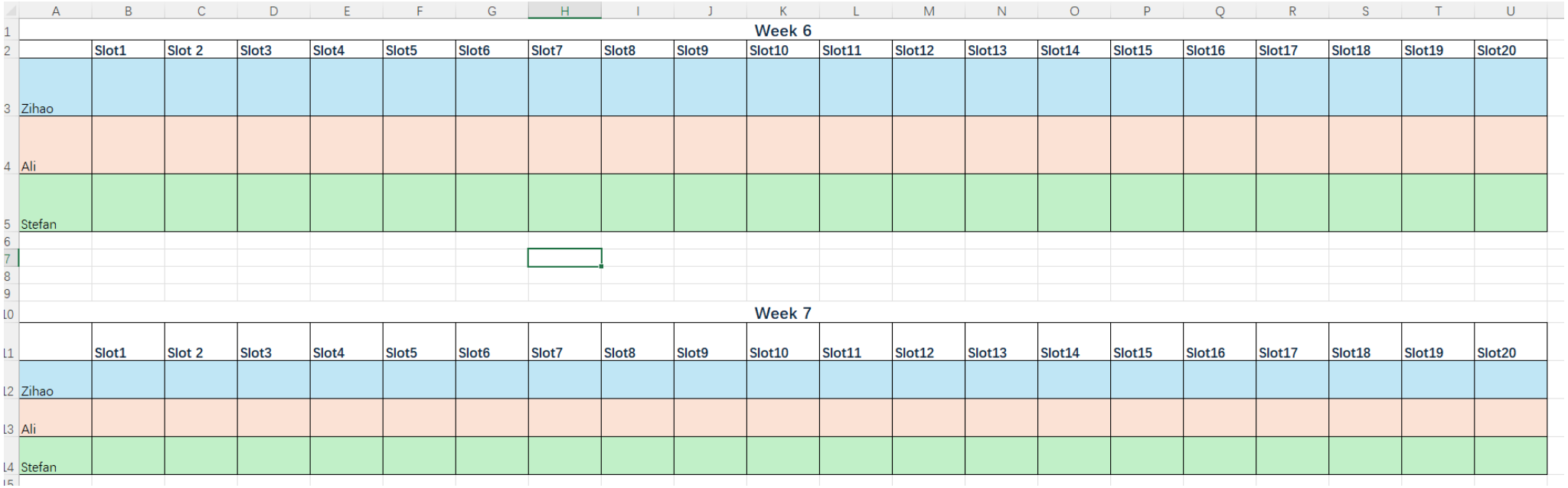
Recap on CPU: A Factory of Data

- 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

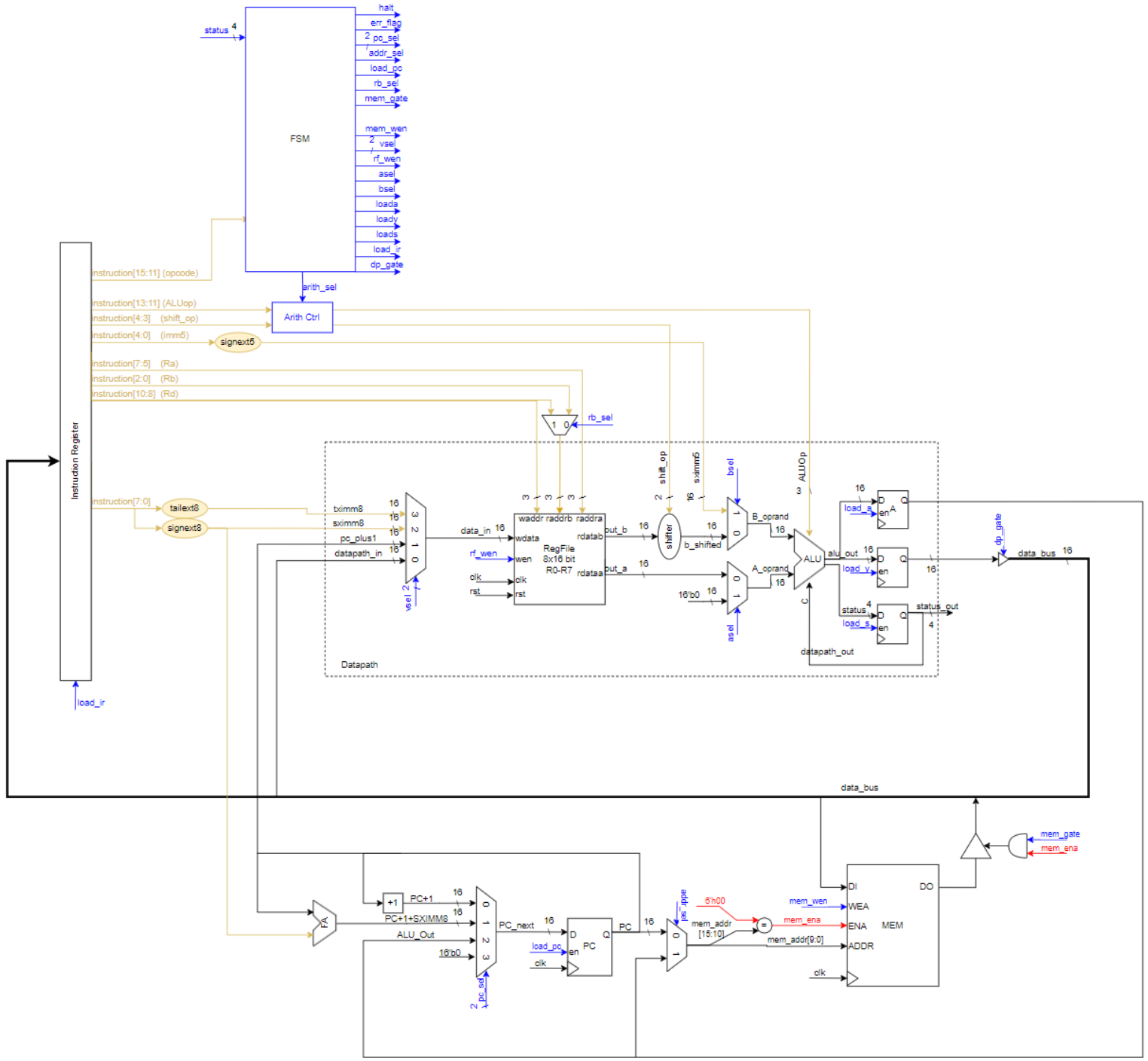


										Week 6										
	Slot1	Slot 2	Slot3	Slot4	Slot5	Slot6	Slot7	Slot8	Slot9	Slot10	Slot11	Slot12	Slot13	Slot14	Slot15	Slot16	Slot17	Slot18	Slot19	Slot20
Zihao																				
Ali																				
Stefan																				
										Week 7										
	Slot1	Slot 2	Slot3	Slot4	Slot5	Slot6	Slot7	Slot8	Slot9	Slot10	Slot11	Slot12	Slot13	Slot14	Slot15	Slot16	Slot17	Slot18	Slot19	Slot20
Zihao																				
Ali																				
Stefan																				

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, them submit to KEATS

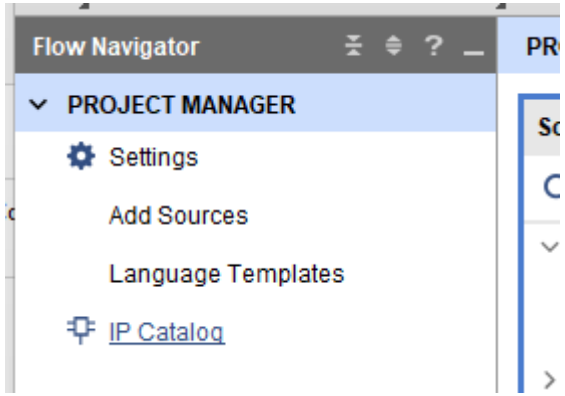
Complete Architecture: Overview

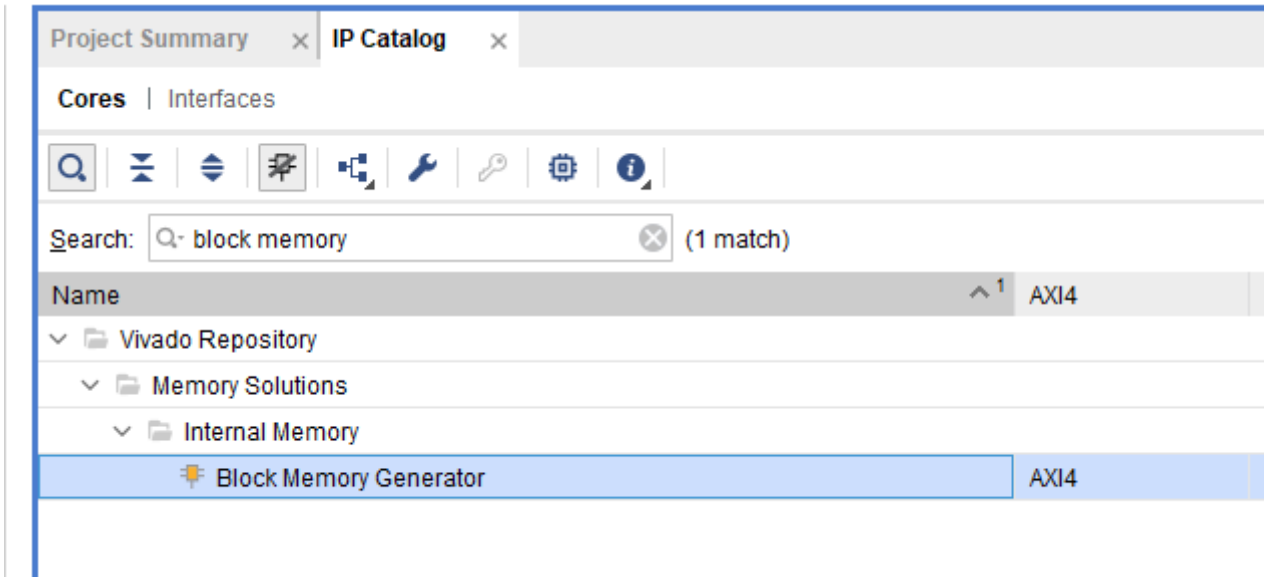


Memory

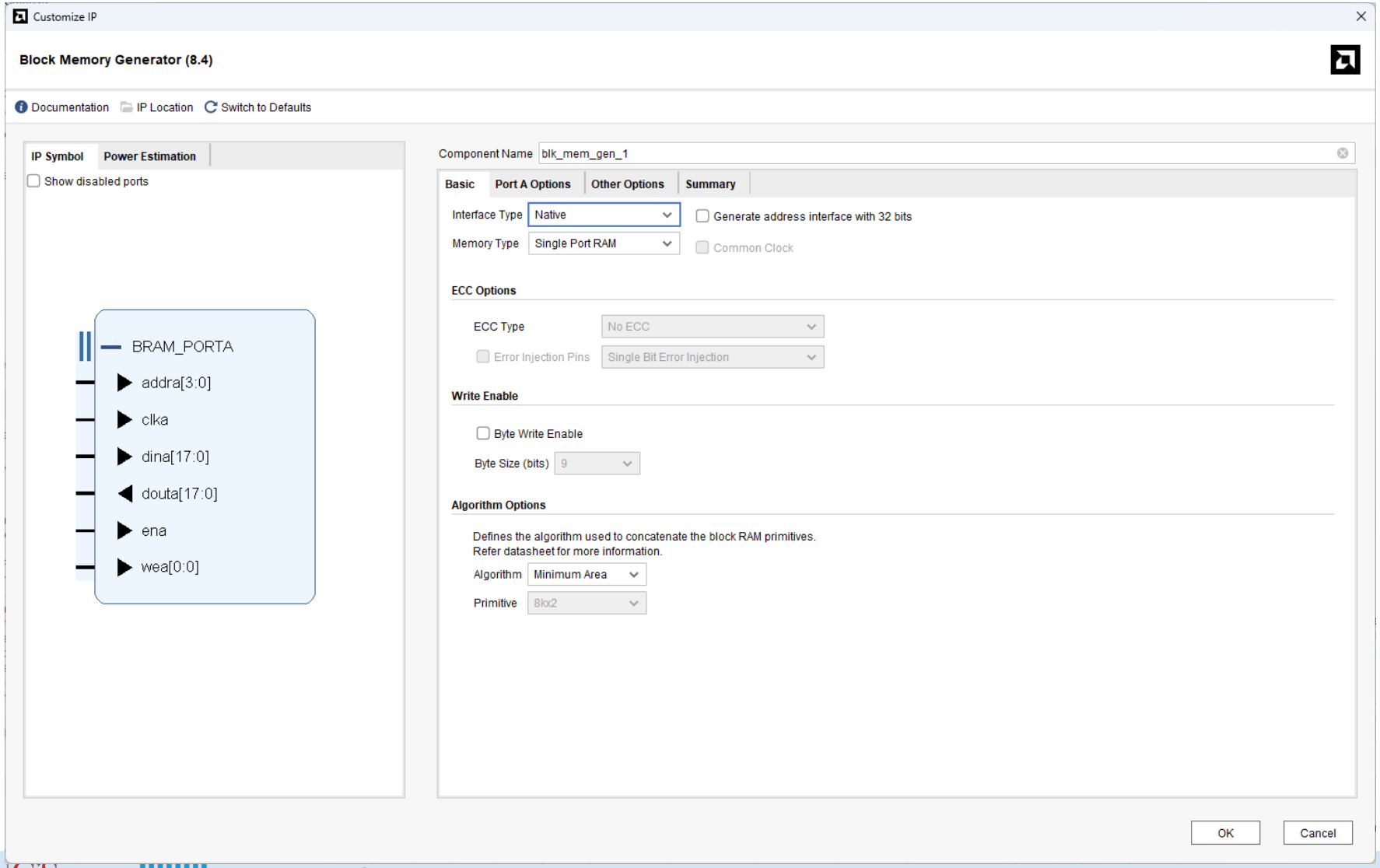
WHERE PROGRAM AND DATA STORE

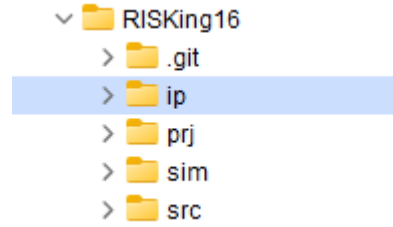
Memory Module: Using Vivado Block Memory



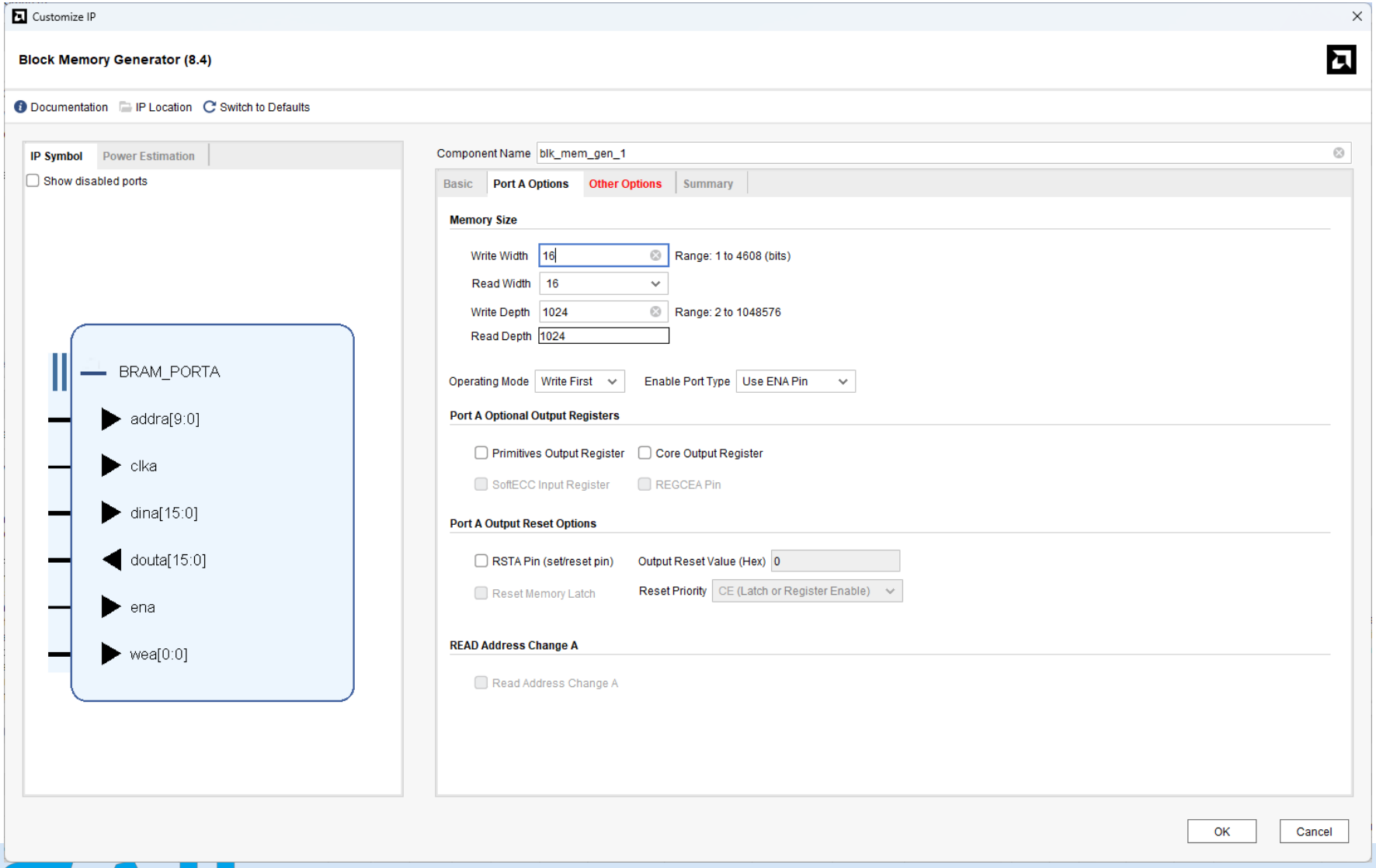


Memory Module: Using Vivado Block Memory





Memory Module: Using Vivado Block Memory



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.

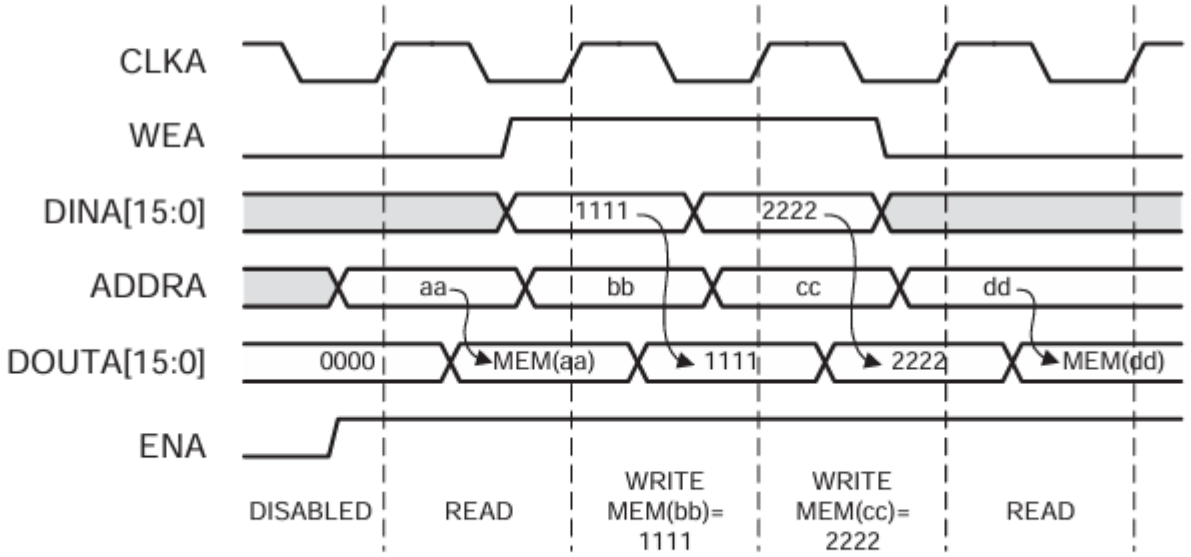
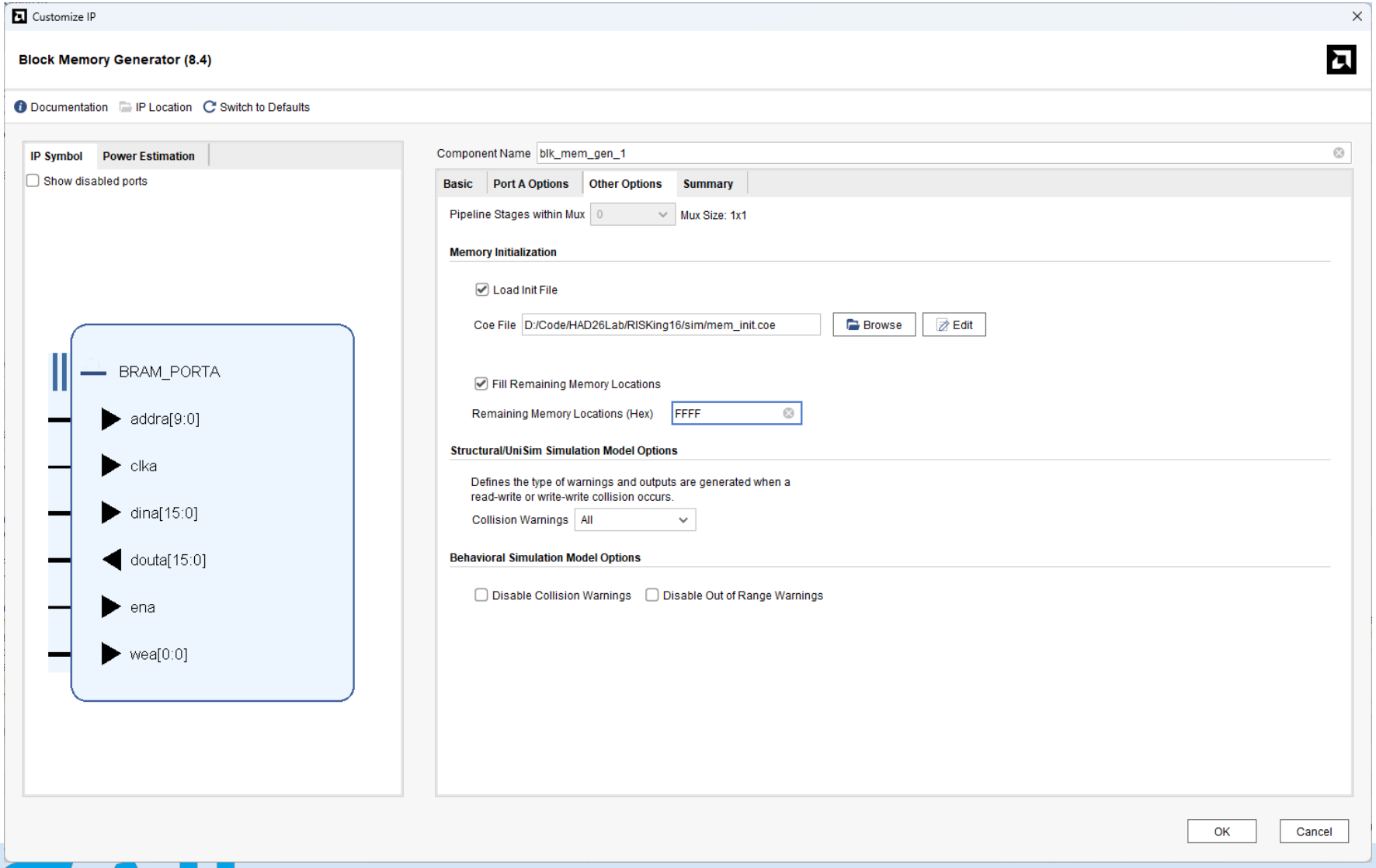


Figure 3-9: **Write First Mode Example**

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

Memory Module: Using Vivado Block Memory

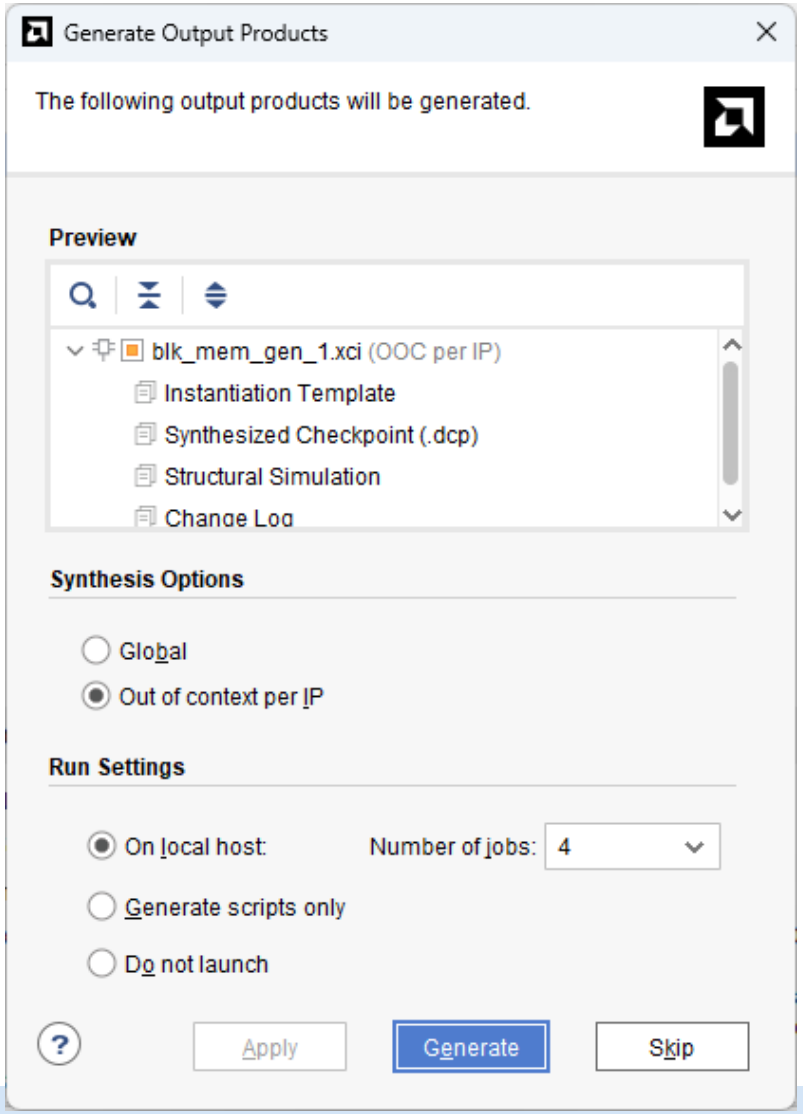


Memory Module: Using Vivado Block Memory

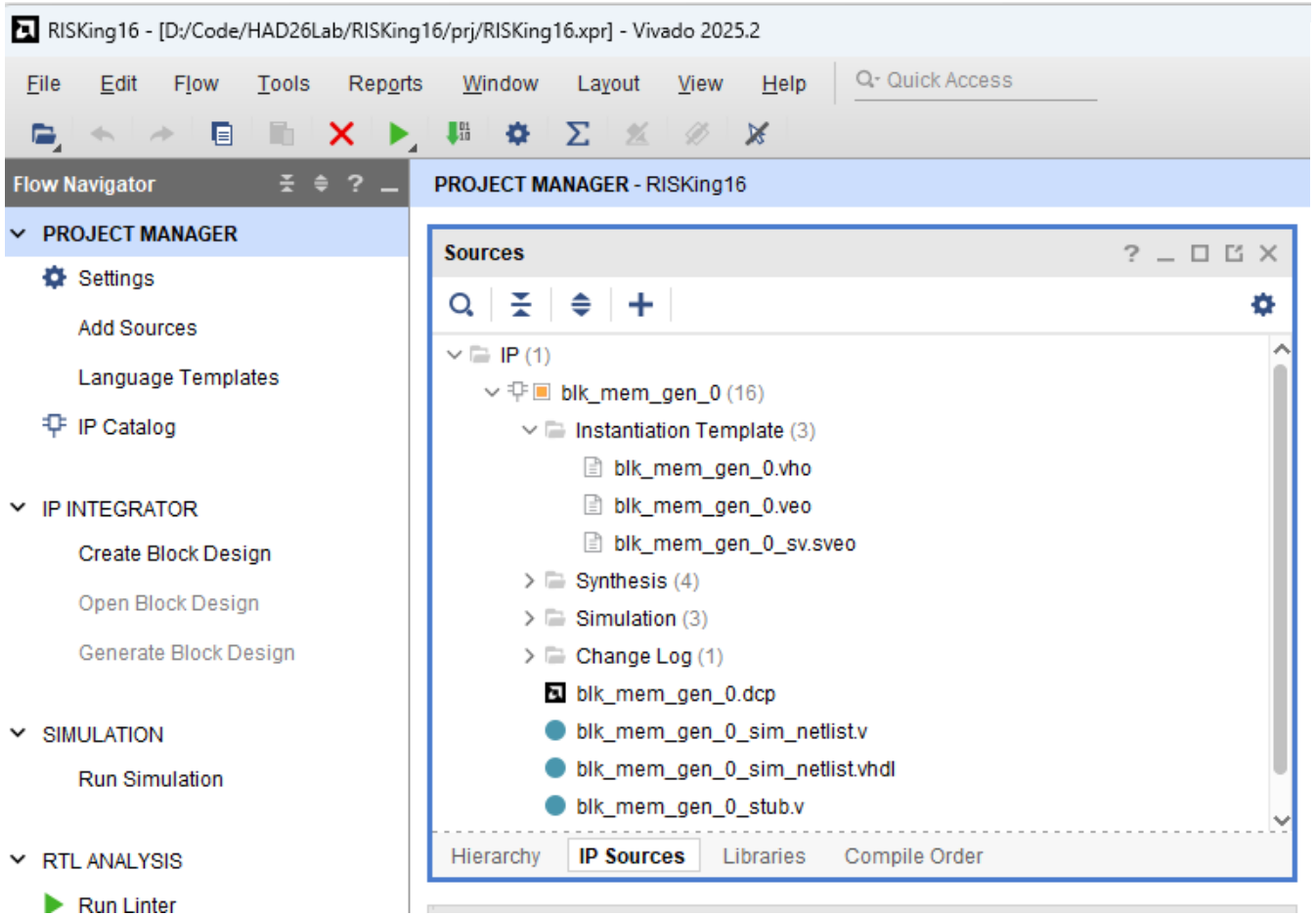
sim > mem_init.coe

```
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,
```

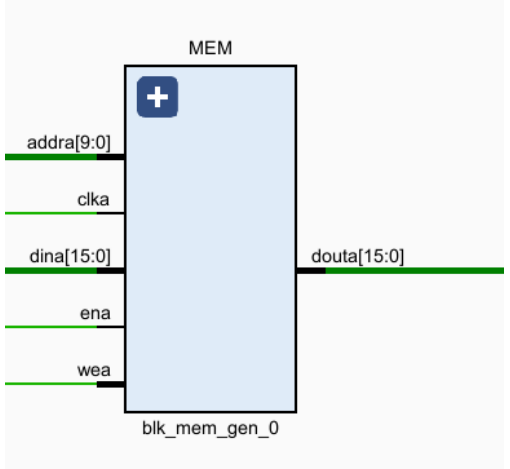
Memory Module: Using Vivado Block Memory



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