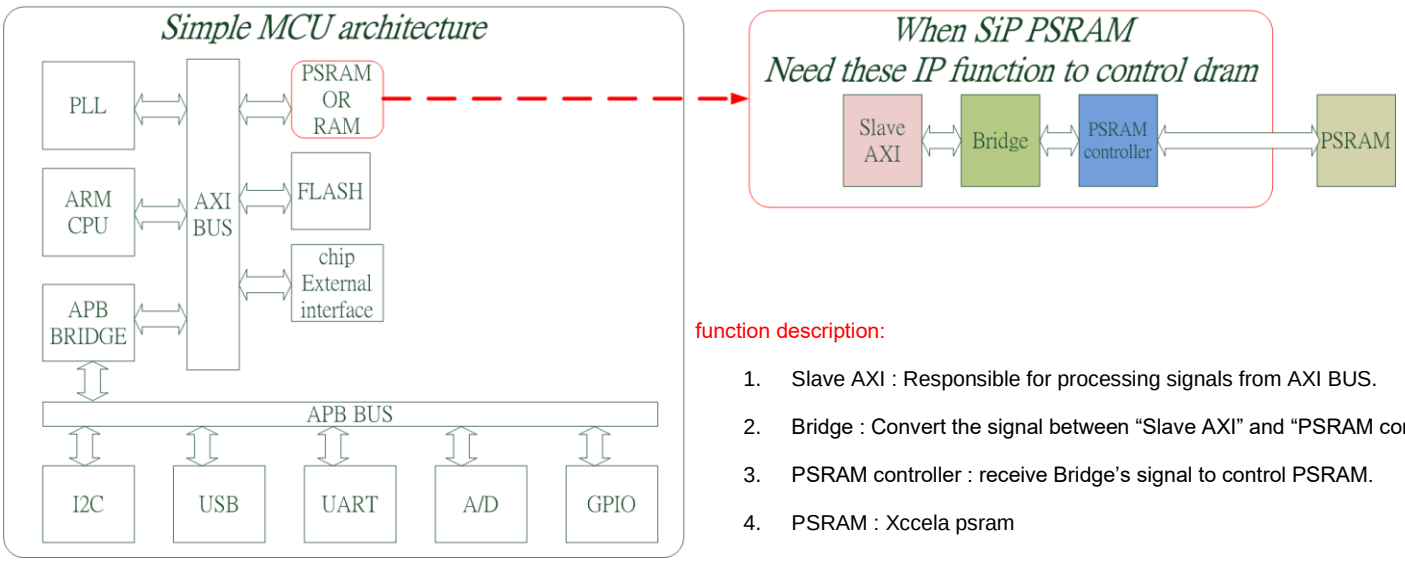
Catalog

1 Purpose and IP features introduction

1.1 Purpose introduction

There are many reduced pin count (RPC) memory on the market, but they are all special protocols. This example finds and uses **Xccela PSRAM** to do IP development.

Usually, the MCU controls the PSRAM controller through the "AXI BUS" protocol, and the IP between the two still lacks some functions as shown in the following figure (Slave AXI + Bridge + PSRAM controller), the **open source** sample code is provided, for your reference and use.



function description:

1. Slave AXI : Responsible for processing signals from AXI BUS.
2. Bridge : Convert the signal between “Slave AXI” and “PSRAM controller”.
3. PSRAM controller : receive Bridge’s signal to control PSRAM.
4. PSRAM : Xccela psram

1.2 IP features introduction

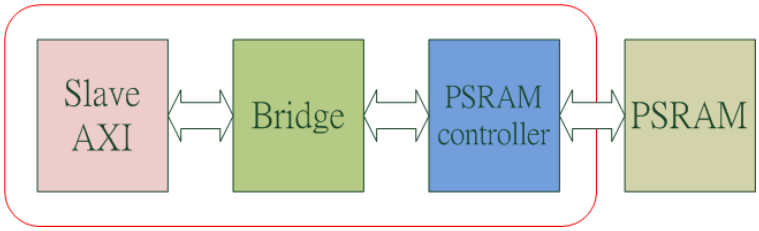
This IP (Slave AXI + Bridge + PSRAM controller) supports the functions listed below:

- Supports Xccela x8 IO psram protocol.
- Supports Xccela psram maximum frequency 200 MHz
- Supports Xccela psram mode register write and read, MR0~MR8.
- Supports Xccela psram array write and read.
- Supports AXI4 maximum frequency 250 MHz
- Supports AXI4 data width of 16/32-bit
- Supports AXI4 address width of 32-bit
- Supports AXI4 INCR burst length of 256.
- Supports AXI4 Narrow Transaction.

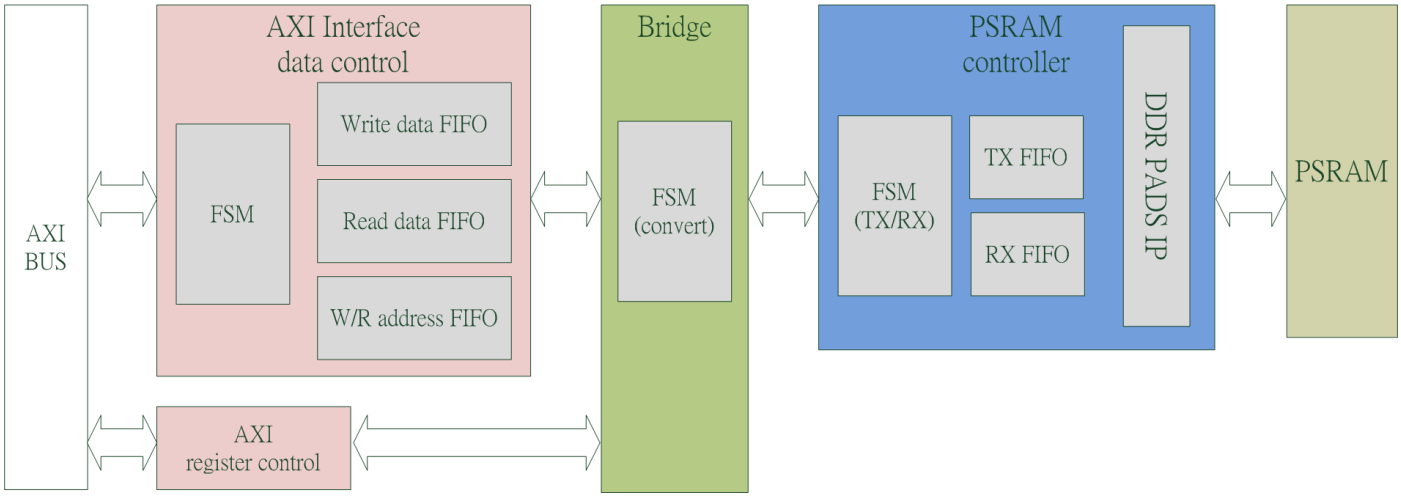
2 IP block diagram

The following diagram uses a detailed block diagram to illustrate the functionality required for each block.

Simple block diagram



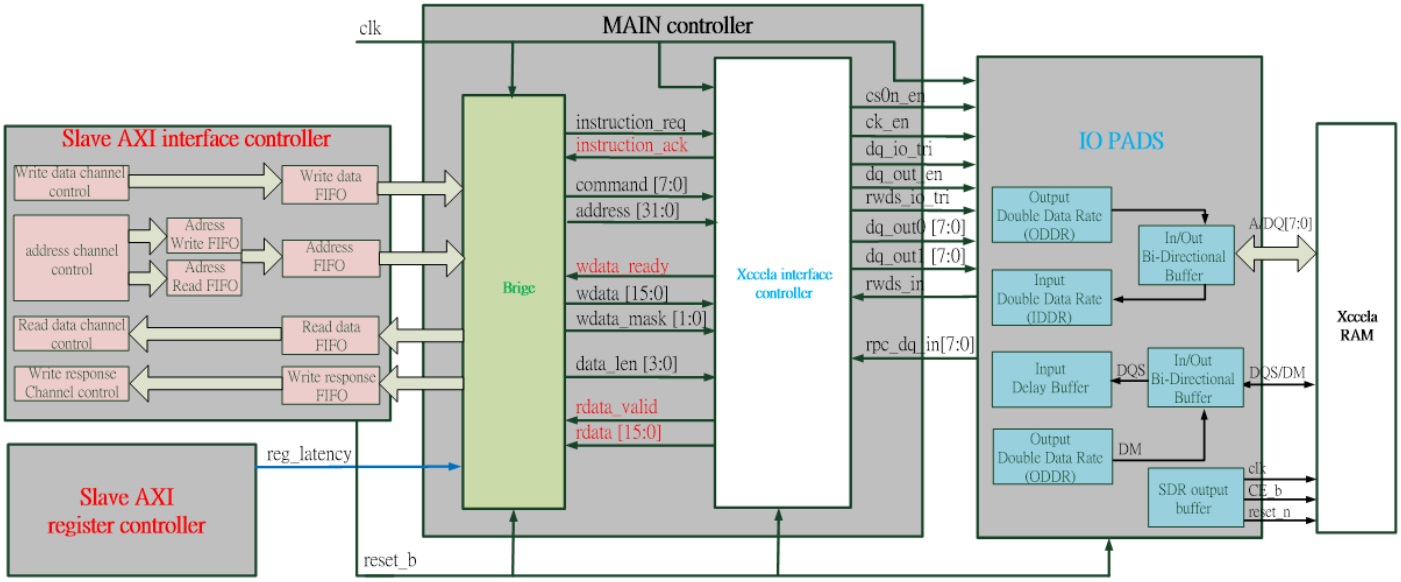
Detailed block diagram



3 IP finite-state machine(FSM) introduction

3.1 IP block include signal name

The following figure shows the wire between each ip block, wire signal name must be defined for Brige handshake controller.



3.2 Signal name introduction

Signals of bridge-to-controller

Name	Bit	IO	Description
clk	1	Input	It from clock generator IP Clock 10MHZ~200MHZ
reset_b	1	Input	Reset signal. (active low) For initial or timeout or stuck use, it from AXI controller.
Instruction_req	1	Output	Instruction include **command and address** Bridge request controller for instruction. (active High)
Instruction_ready	1	Input	Controller ready to accept Instruction. (active High)
command	7	Output	Command
address	32	Output	Start address of the memory
wdata_ready	1	Input	Write data ready from controller. (active High)
wdata	16	Output	Bridge transmits data to controller
wdata_mask	2	Output	1: mask data 0: not mask data
data_len	4	Output	Setting data burst length
rdata_valid	1	Input	Read data valid from controller. (active High)
rdata	16	Input	Controller transmits data to bridge

Signals of controller-to-PSRAM

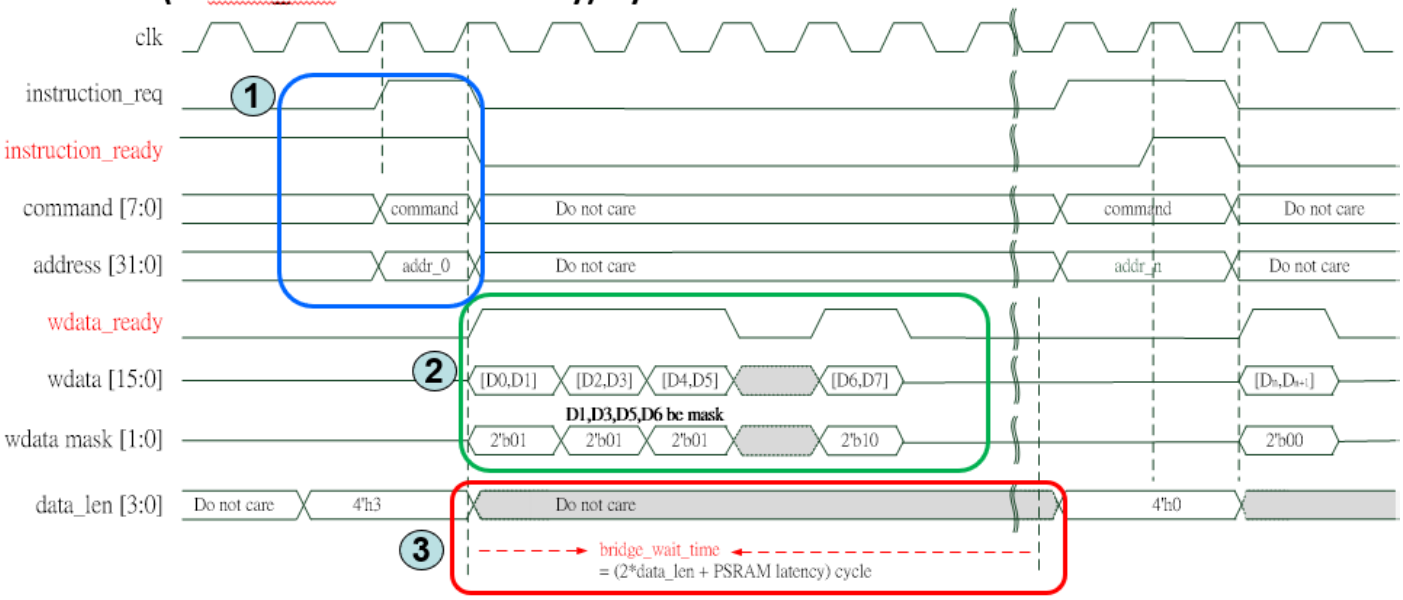
Name	Bit	IO	Description
clk	1	Output	Clock 10MHZ~200MHZ
A/DQ[7:0]	7	bidirection	Address/DQ bus [7:0]
DQS/DM	1	bidirection	DQ strobe clock during reads Data mask during writes. 1: mask data 0: not mask data
CE_b	1	Output	Chip select, active low. When CE#=1, chip is in standby state.

3.3 Command table of bridge-to-controller

NO	Command	Function	Note
1	8'b0000_0000 = 0x00	INIT	
2	8'b0000_0001 = 0x01	reg_w	Mode register write
3	8'b0000_0010 = 0x02	reg_r	Mode register read
4	8'b0000_0100 = 0x04	data_w	Data write
5	8'b0000_1000 = 0x08	data_r	Data read
6	8'b1000_0000 = 0x80	gb_rst	Global reset

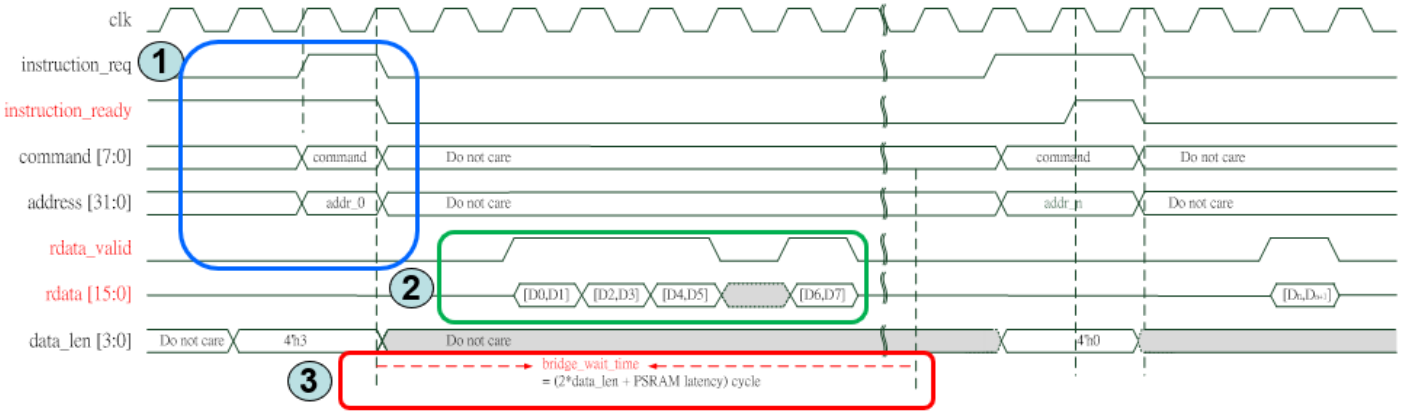
3.4 Write operation waveform of bridge-to-controller

1. instruction_ready before instruction_req handshake
2. wdata_ready (high -> low -> high)
3. add bridge_wait_time when read and write.
   - receive ack to start the calculation, if timeout, bridge will feedback to AXI
   - (2*data_len + PSRAM latency) cycle

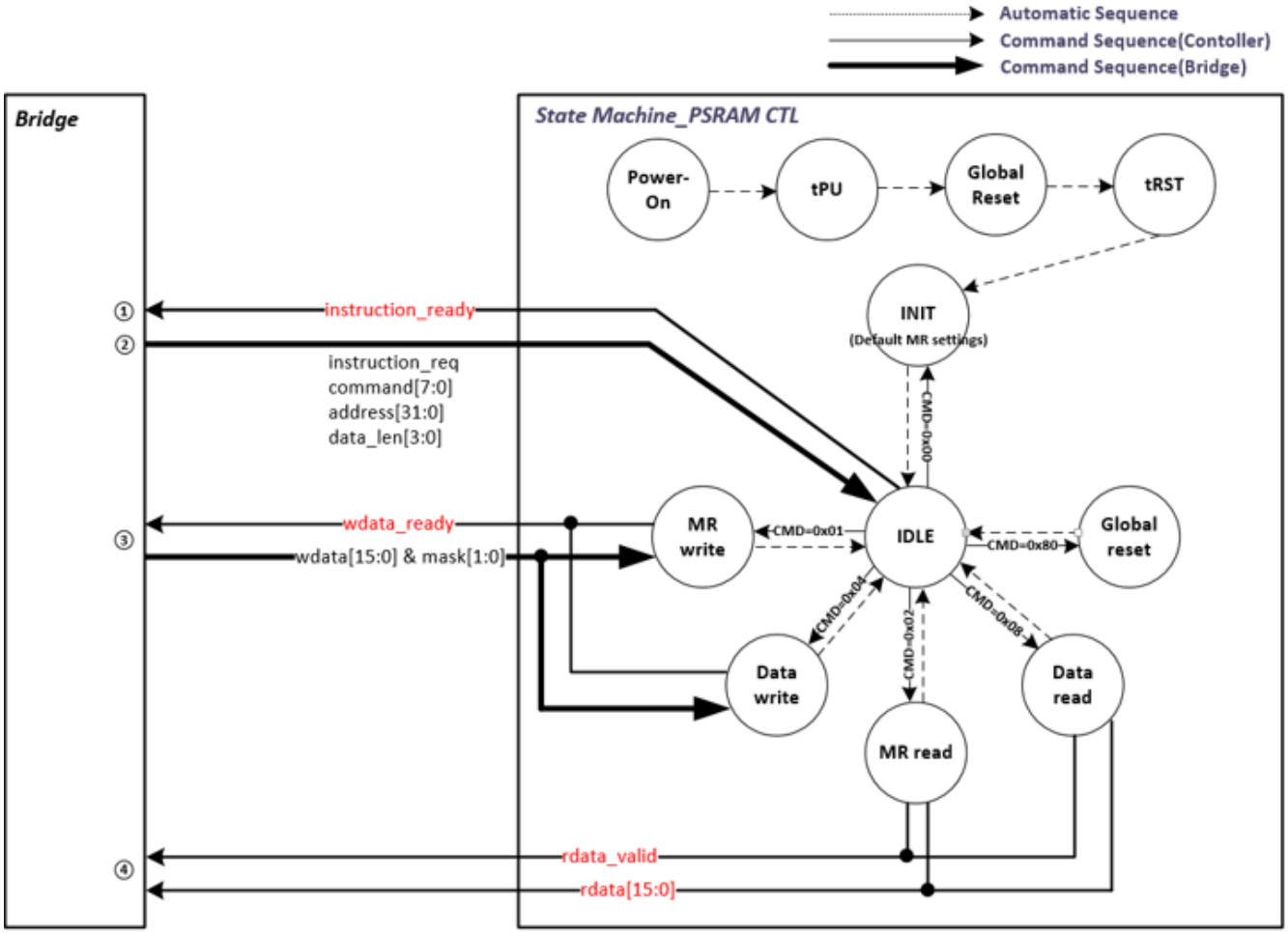


3.5 Read operation waveform of bridge-to-controller

1. instruction_ready before instruction_req handshake
2. rdata_valid (high -> low -> high)
3. add bridge_wait_time when read and write.
   - receive ack to start the calculation, if timeout, bridge will feedback to AXI
   - (2*data_len + PSRAM latency) cycle



3.6 IP PSRAM controller(FSM) introduction



1. After Power-on, state machine will perform automatically the complete PSRAM initialization process, and standby in IDLE state with instruction_ready high.
2. When getting the instruction_req from Bridge, state machine will check command from bridge and jump to the corresponding work task.
3. When a task is completed, state machine will automatically jump to IDLE state for waiting next command.

Communication Example

MR write after Initialization:
Power-On -> ...-> INIT -> IDLE -> ① -> ② -> MR write -> ③ -> IDLE

Data write, Data read after Initialization:
Power-On -> ...-> INIT -> IDLE ->① -> ② -> Data write, ③ -> IDLE -> ① -> ② -> Data read, ④ -> IDLE

4 Testbench description

4.1 Operating frequency setting

1. Set AXI frequency, please refer to the description in columns 8 to 23.

```
 8    //Slave AXI clock
 9    `define AXI_clock_200M
10
11    `ifdef AXI_clock_250M
12        parameter AXI_CLK_PERIOD = 4000;
13        parameter AXI_CLOCK_MHZ = 250;
14    `else
15    `ifdef AXI_clock_200M
16        parameter AXI_CLK_PERIOD = 5000;
17        parameter AXI_CLOCK_MHZ = 200;
18
19    `else
20        parameter AXI_CLK_PERIOD = 4000;
21        parameter AXI_CLOCK_MHZ = 250;
22    `endif
23    `endif
```

2. Set PSRAM controller frequency, please refer to the description in columns 28 to 48.

```
28    //clock define for RAM_Controller AND PSRAM
29    `define tCLK5000  // Note :psram verilog module "most" define same word
30
31    `ifdef tCLK5000 //0.5ns @200M
32      parameter   integer INIT_CLOCK_HZ = 200_000000; // controller mode
33      parameter RAM_CLK_PERIOD = 5000;                // PLL clock for RAM_Controller AND PSRAM
34    `else
35    `ifdef tCLK6000 //0.6ns @166M
36      parameter   integer INIT_CLOCK_HZ = 166_000000; // controller mode
37      parameter RAM_CLK_PERIOD = 6000;                // PLL clock for RAM_Controller AND PSRAM
38
39    `else
40    `ifdef tCLK7500 //0.75ns @133M
41      parameter   integer INIT_CLOCK_HZ = 133_000000; // controller mode
42      parameter RAM_CLK_PERIOD = 7500;                // PLL clock for RAM_Controller AND PSRAM
43    `else
44      parameter   integer INIT_CLOCK_HZ = 200_000000; // controller mode
45      parameter RAM_CLK_PERIOD = 5000;                // PLL clock for RAM_Controller AND PSRAM
46    `endif
47    `endif
48    `endif
```

3. Set PSRAM verilog module frequency, it use define to select operating frequency, It must be modified in PSRAM verilog.

```
93  `define tCLK5000
94
95
96
97  `define tDQSCKmin   2   // fixed 2
98
99  `define tHZ         4
100 `define tDQSCK      3   // range 2-5.5ns (should use max for model delivery)
101 `define tDQSQ       0.4 // max 0.4ns @200, 0.5ns @166, 0.6ns @133
102 `define tRC     60
103
104 module psram_model (xDQ, xDQSDM, xCEn, xCLK, xRESETn);
105
106 `ifdef tCLK5000
107     parameter CLOCK_MHZ = 200;
108     parameter tCLKmin   = 5;
109     parameter tDQSCKmax = 5.5;          // should be used for timing checks only, and not model output behaviour
110     parameter tDQSQmax  = 0.4;          // should be used for timing checks only, and not model output behaviour
111 `else
112 `ifdef tCLK6000
113     parameter CLOCK_MHZ = 166;
114     parameter tCLKmin   = 6;
115     parameter tDQSCKmax = 5.5;          // should be used for timing checks only, and not model output behaviour
116     parameter tDQSQmax  = 0.5;          // should be used for timing checks only, and not model output behaviour
117 //`else `define tCLK7500
118 `else
119     parameter CLOCK_MHZ = 133;
120     parameter tCLKmin   = 7.5;
121     parameter tDQSCKmax = 5.5;          // should be used for timing checks only, and not model output behaviour
122     parameter tDQSQmax  = 0.6;          // should be used for timing checks only, and not model output behaviour
123 `endif
124 `endif
```

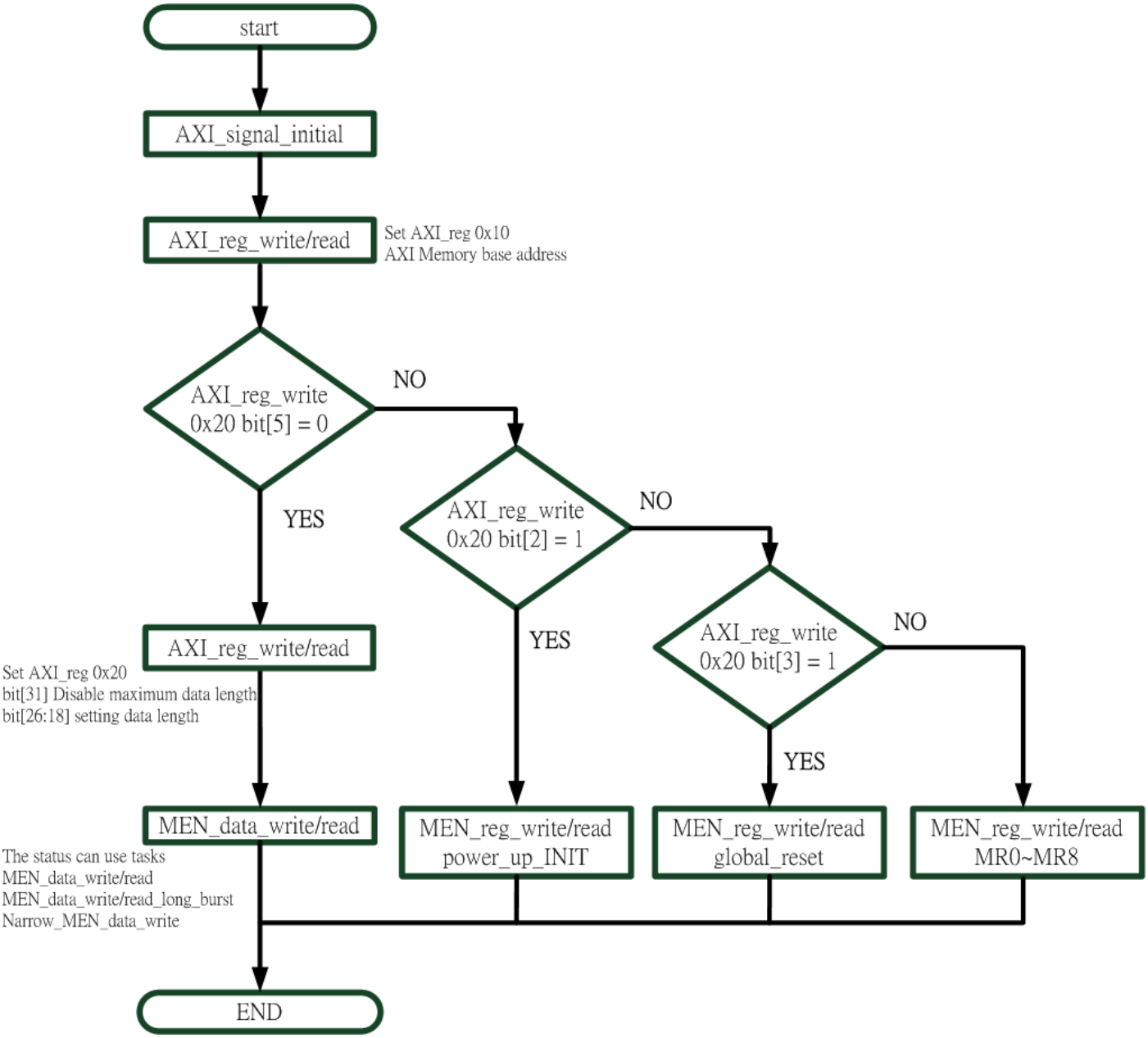
4.2 Testbench task table

Testbench task is a test pattern written according to the AXI protocol, all tasks are shown below:

Task item	Task description
AXI_signal_initial	Initializes AXI for each signal.
AXI_reg_write	Write to AXI's register.
AXI_reg_read	Read from AXI's register.
MEN_reg_write	Write to memory's register.
MEN_reg_read	Read from memory's register.
MEN_data_write	Write data to memory.
MEN_data_read	Read data from memory.
Narrow_MEN_data_write	Use AXI Narrow Transfer to Write data to memory.
MEN_data_write_long_burst	Set the burst length 0~255 and then write data to memory.
MEN_data_read_long_burst	Set the burst length 0~255 and then read data from memory.
global_reset	Memory global reset.
power_up_INIT_MR0_MR4	Memory power up Initializes.

4.3 IP task switched flow

When a task needs to be switched, it needs to meet the following setup flow.

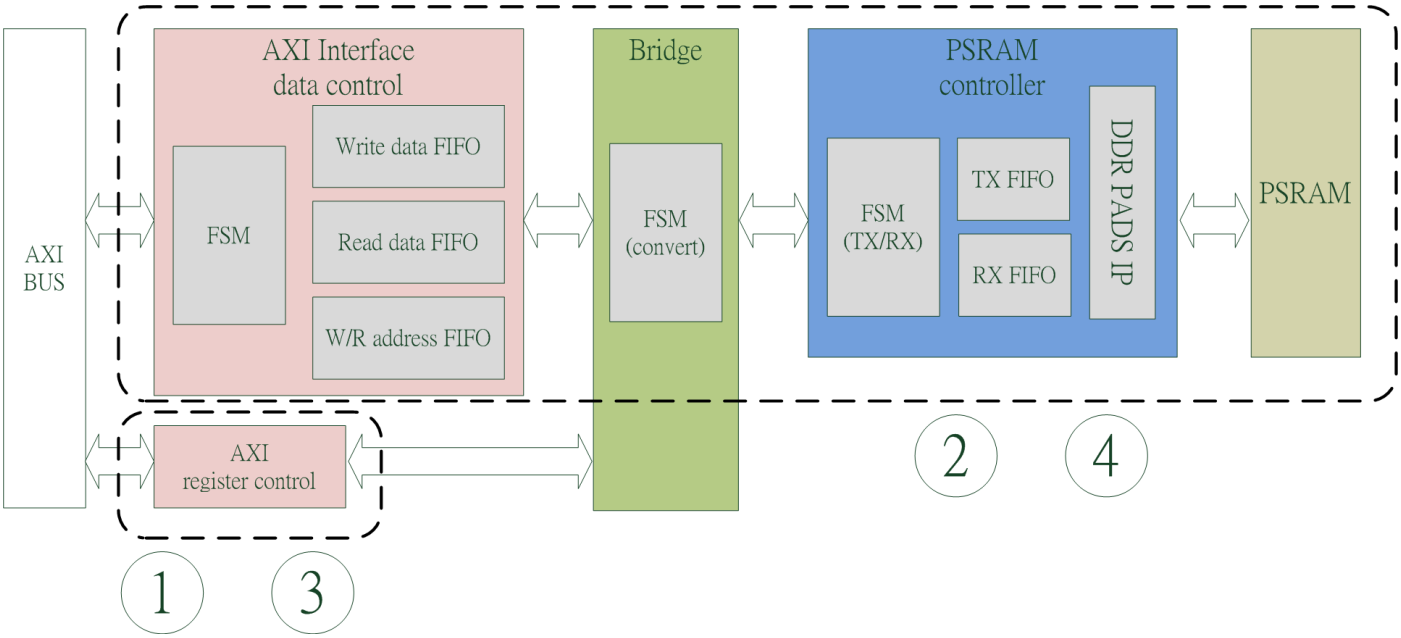


4.4 The IP setting sequence

The axi register needs to be set before the IP is used

1. Setting AXI4 register 0x10 :
   set up base address for each memory.
2. Setting memory register :
   set up memory register. Ex : MR0 and MR4
3. Setting AXI4 register 0x20 :
   change memory register mode to array write and array read.
4. Start using memory read and write functions.

Note: AXI4 register 0x00, can monitoring IP status any time.



5 IP verify items

Verification item list

Item	Function test
1	AXI4 register write and read test
2	Memory register (MRW) write and (MRR) read test Note: (Narrow Transfer) : Data bus is 32 bit , but only 16 bits of data are transferred
3	Write + mask & read axlen=1 (burst len=2) , axsize=2 (4 Byte)
4	Write & read axlen=1 (burst len=2) , axsize=2 (4 Byte)
5	Write & read axlen=1 (burst len=4) , axsize=1 (2 Byte) Note: (Narrow Transfer) : Data bus is 32 bit , but only 16 bits of data are transferred
6	Write + mask & read axlen=1 (burst len=4) , axsize=2 (4 Byte) Data bus is 32 bit , 32 bits of data are transferred
7	AXI4 Max Burst length 256 of AXI+PSRAM array write and read test
8	Write and read sequence test (W->R->W->R) or (W->W->R->R)
9	Maximum frequency of AXI + minimum frequency of PSRAM.