

**Department of Electronic & Telecommunication
Engineering
University of Moratuwa**

EN4021 - Advanced Digital Systems



Design Document

System Bus Design Report

AMARATHUNGA D.N.	210037G
PASIRA I.P.M.	210446J
RAJAPAKSHA S.S.D.Z.	210508D

Date - 2025.12.09

Contents

1 Overview	2
2 Single Bus Design	2
2.1 Design Parameters	2
2.2 I/O Port Summary	3
2.3 Bus Architecture	3
2.3.1 Bus Signals	3
2.3.2 Address Decoding Scheme	3
2.4 Bus Arbiter	3
2.5 Address Decoder	5
2.6 Master Interface	7
2.7 Slave Interface	9
3 Bus Bridge Design	11
3.1 Design Parameters	11
3.2 I/O Port Summary	11
3.3 UART Implementation	11
3.4 Bus Bridge Master	12
3.5 Bus Bridge Slave	12
3.6 Memory Map for Bus A	13
3.7 Memory Map for Bus B	13
4 Address Map for Unified Design	13
5 Timing Diagrams	13
6 Timing Characteristics	15
6.1 UART Timing	15
7 Resource Summary	15

1 Overview

The design implements a multi-master, multi-slave serial system bus. It provides a complete framework for communication between multiple devices, featuring:

- Two master interfaces, which can initiate read and write transactions on the bus.
- Three slave interfaces, implemented using local memory blocks (BRAM), which respond to master requests.
- Priority-based bus arbitration, ensuring orderly access to the bus, along with support for split transactions to improve throughput and efficiency.

This architecture also allows integration of bus bridges, enabling communication with external subsystems or other bus domains. The design emphasizes scalability, synchronization, and efficient data transfer between masters and slaves.

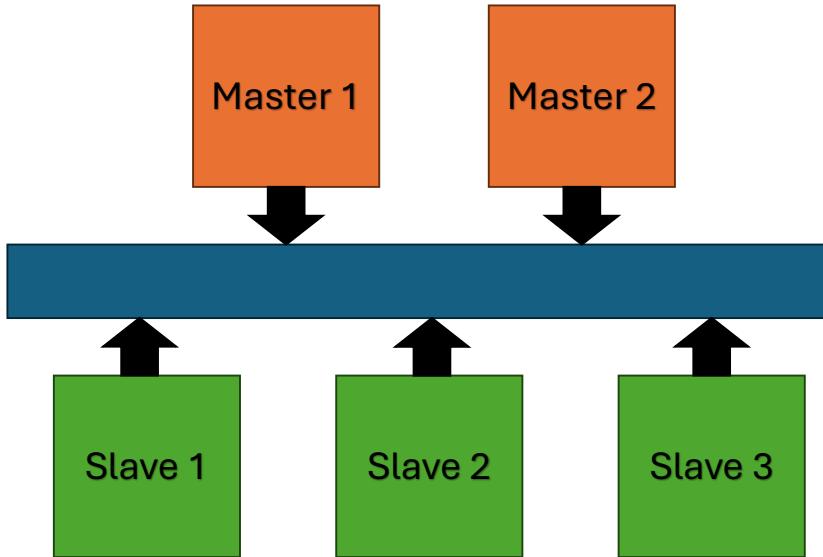


Figure 1: Top-Level Block Diagram of `demo_top_bb`

2 Single Bus Design

2.1 Design Parameters

Table 1: Configurable Design Parameters

Parameter	Default Value	Description
ADDR_WIDTH	16 bits	Total address bus width
DATA_WIDTH	8 bits	Data bus width
SLAVE_MEM_ADDR_WIDTH	12 bits	Slave memory address space
SLAVE_ADDR_WIDTH	4 bits	Slave address width

2.2 I/O Port Summary

Table 2: Top-Level I/O Ports of `demo_top_bb`

Port	Direction	Width	Description
<code>clk</code>	Input	1	System clock
<code>rstn</code>	Input	1	Active-low synchronous reset
<code>start</code>	Input	1	Transaction start trigger (falling edge)
<code>mode1 / mode2</code>	Input	1	0 = Read, 1 = Write
<code>en1 / en2</code>	Input	1	Enable either master 1 or master 2
<code>LED</code>	Output	8	Debug LED output
<code>ready1 / ready2</code>	Output	1	System ready for new transaction

2.3 Bus Architecture

2.3.1 Bus Signals

The bus uses **bit-serial** transfers for all data and addresses, minimizing interconnect complexity.

Table 3: Serial Bus Signals

Signal	Width	Description
<code>wdata</code>	1	Serial write data (address + data)
<code>rdata</code>	1	Serial read data from slave
<code>mode</code>	1	Read (0) / Write (1) mode
<code>mvalid</code>	1	Master data valid
<code>svalid</code>	1	Slave data valid
<code>breq</code>	1	Bus request
<code>bgrant</code>	1	Bus grant
<code>ack</code>	1	Address acknowledge
<code>split</code>	1	Split transaction signal
<code>ready</code>	1	Slave ready for transaction

2.3.2 Address Decoding Scheme

- **Device Address Width:** $16 - 12 = 4$ bits
- **Memory Address Width:** 12 bits per slave

Table 4: Slave Address Mapping

Slave Select	Device Address	Address Range	Slave
2'b00	4'b0000	0x0000 - 0x07FF	Slave 1 (2KB BRAM)
2'b01	4'b0001	0x1000 - 0x1FFF	Slave 2 (4KB BRAM)
2'b10	4'b0010	0x2000 - 0x2FFF	Slave 3 (4KB BRAM)

2.4 Bus Arbiter

The arbiter implements a priority-based arbitration mechanism where Master 1 always has higher priority over Master 2.

Table 5: Arbiter I/O Signal Description

Signal	Dir	Width	Description
clk	In	1	System clock used for synchronous operation.
rstn	In	1	Active-low reset, initializes the arbiter and clears split-related registers.
breq1	In	1	Bus access request from Master 1.
breq2	In	1	Bus access request from Master 2.
sready1	In	1	Indicates Slave 1 is ready for a new transaction.
sready2	In	1	Indicates Slave 2 is ready for a new transaction.
sreadysp	In	1	Indicates the split-capable slave is ready for a transaction.
ssplit	In	1	SPLIT response from the split-capable slave.
bgrant1	Out	1	Bus grant asserted when Master 1 owns the bus.
bgrant2	Out	1	Bus grant asserted when Master 2 owns the bus.
msel	Out	1	Master select output (0 = Master 1, 1 = Master 2).
msplit1	Out	1	Indicates that a split transaction belongs to Master 1.
msplit2	Out	1	Indicates that a split transaction belongs to Master 2.
split_grant	Out	1	Asserted when a previously split transaction may resume.

The arbiter uses a three-state finite state machine governed by master requests, slave readiness, and split events. Master 1 is always given the highest priority unless it is waiting due to a split response.

- **IDLE:** No master currently owns the bus.
- **M1:** Master 1 is granted bus ownership (highest priority).
- **M2:** Master 2 receives the bus if Master 1 is idle or waiting after a split.

Split transactions are tracked internally, allowing the arbiter to resume the transaction using `split_grant` once the slave becomes ready.

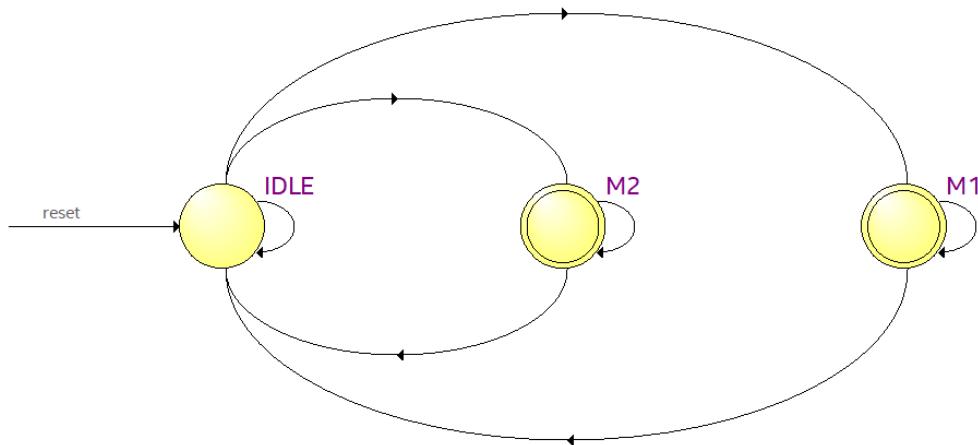


Figure 2: Arbiter State Machine

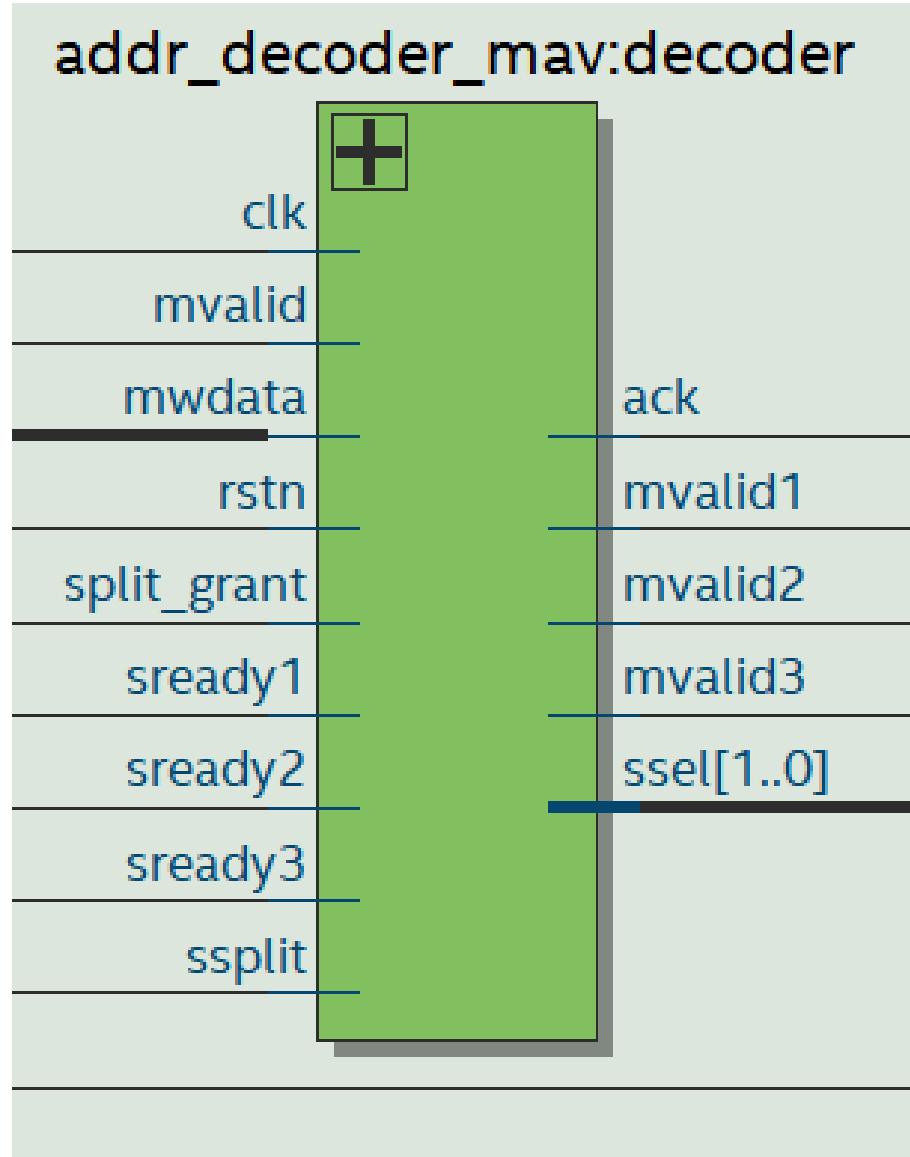


Figure 3: Adress Decoder

Table 6: Arbiter Output Signals per State

State	bgrant1	bgrant2	msel
IDLE	0	0	0
M1	1	0	0
M2	0	1	1

2.5 Address Decoder

The Address Decoder routes serially received device addresses from the master to the appropriate slave. It generates individual valid signals for each slave (`mvalid`), asserts an acknowledgement (`ack`) to the master when a slave is ready, and coordinates with the arbiter to handle and resume split transactions.

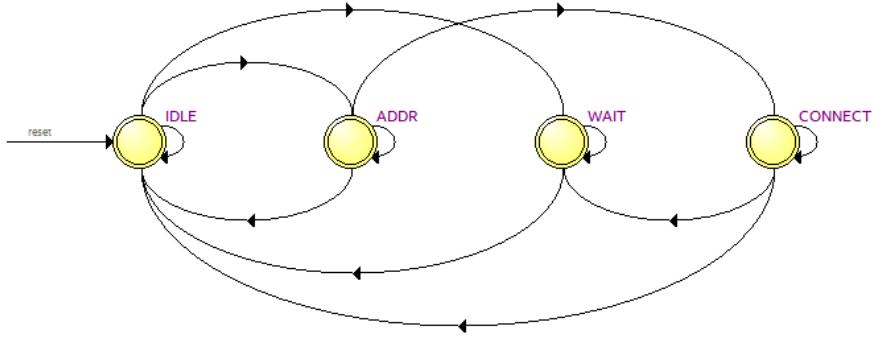


Figure 4: Address Decoder State Machine

Table 7: Address Decoder I/O Signal Description

Signal	Dir	Width	Description
clk	In	1	System clock used for synchronous operation.
rstn	In	1	Active-low reset for initializing the FSM and internal registers.
addr_valid	In	1	Indicates that the serial device address on <code>addr_data</code> is valid.
addr_data	In	1	Serial address bit from the master.
sready	In	3	Each bit indicates readiness of the corresponding slave (Slave 1 to Slave 3).
split	In	1	SPLIT indication from a slave; cannot perform transaction at this moment.
split_grant	In	1	Permission from the arbiter to resume a previously split transaction.
ssel	Out	2	Slave select output (00 = Slave 1, 01 = Slave 2, 10 = Slave 3).
ack	Out	1	Acknowledgement that a valid slave has been selected and is ready for communication.
mvalid	Out	3	Valid signals for each slave (combinational), asserted during <code>S_CONNECT</code> and <code>WAIT_TXN</code> when <code>addr_valid</code> = 1.

The Address Decoder operates using a four-state FSM:

- **IDLE** – Waits for a new master transaction. If `addr_valid` = 1, the first address bit is captured and the FSM moves to **S_ADDR_RECEIVE**. If `split_grant` = 1, the previously saved split address is restored and the FSM moves to **WAIT_TXN**.
- **S_ADDR_RECEIVE** – Serial address bits are shifted into an internal register (`slave_addr`). When all address bits are received, the FSM transitions to **S_CONNECT**.
- **S_CONNECT** – The selected slave is enabled and `ssel` is set. If the slave is ready (`sready[slave_addr]` = 1), an `ack` is sent to the master. The combinational `mvalid` corresponding to the slave is asserted when `addr_valid` = 1. If the master has started the transaction, FSM moves to **WAIT_TXN**, otherwise it stays in **S_CONNECT**.

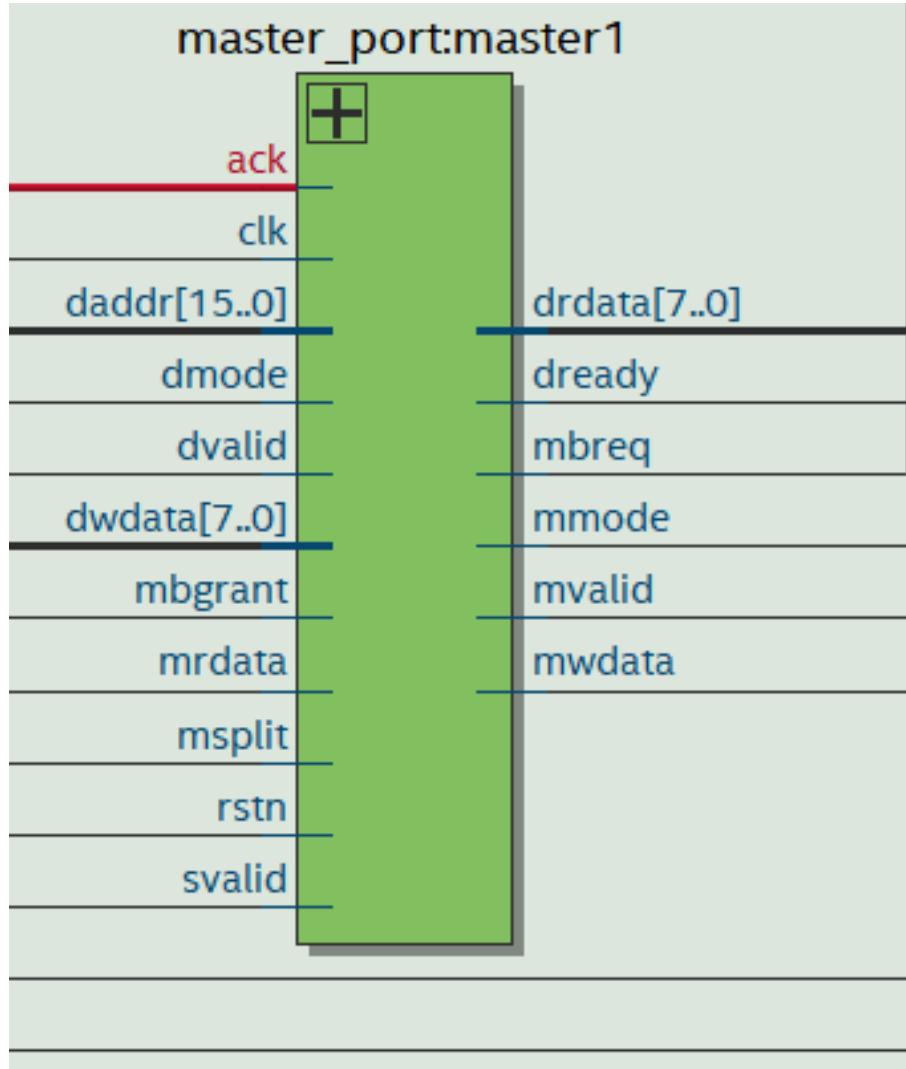


Figure 5: Master

- **WAIT_TXN** – Waits until the targeted slave becomes ready. If a split occurs (`split = 1`), the slave address is saved internally to resume later when `split_grant = 1`. Non-split transactions can continue even if a split is pending. Once the slave is ready or after split resolution, FSM returns to **IDLE**.

2.6 Master Interface

The Master Interface module converts parallel address and data from the master into a serial stream suitable for the shared bus. It communicates with the arbiter to request bus access, sends the slave device address and memory address serially, and handles read/write operations. It also manages split transactions and ensures proper handshaking with both the arbiter and the address decoder.

Table 8: Master Interface I/O Signal Description

Signal	Dir	Width	Description
clk	In	1	System clock for synchronous operation.
rstn	In	1	Active-low reset.
mwdata	In	DATA_WIDTH	Parallel write data from master.
maddr	In	ADDR_WIDTH	Parallel address from master.
mwvalid	In	1	Indicates that <code>mwdata</code> and <code>maddr</code> are valid.
mrdata	Out	DATA_WIDTH	Read data from the slave returned to the master.
mrvalid	Out	1	Indicates that <code>mrdata</code> is valid.
mready	Out	1	Ready signal to master indicating the interface can accept new transaction.
wen	In	1	Write enable signal from master.
bwdata	Out	1	Serial data stream (address/data) sent to the bus.
brdata	In	1	Serial read data received from the bus.
bmode	Out	1	Bus mode (0 = read, 1 = write).
bwvalid	Out	1	Indicates that <code>bwdata</code> is valid.
brvalid	In	1	Indicates that <code>brdata</code> is valid.
bus_busy	In	1	Indicates that the bus is currently busy.
mbreq	Out	1	Request signal to the arbiter to gain bus access.
mbgrant	In	1	Bus grant signal from the arbiter.
msplit	In	1	Indicates a split transaction from the arbiter.
ack	In	1	Acknowledgement from the address decoder that the slave address is valid.

The Master Interface operates using a seven-state FSM:

- **IDLE** – Waits for a valid transaction from the master (`mwvalid`). Captures the parallel address and data, saves the mode (`wen`), and moves to the **REQ** state. While in IDLE, `mready` is asserted to indicate the interface is ready.
- **REQ** – Asserts `mbreq` to request bus access from the arbiter. Waits for `mbgrant` before proceeding to **SLAVE_ADDR**.
- **SLAVE_ADDR** – Serially sends the slave device address (MSBs of the master address) over the bus (`bwdata`). Waits for acknowledgment (`ack`) from the address decoder before proceeding to **ADDR**.
- **ADDR** – Serially sends the memory address portion of the master address. After sending all bits, transitions to either **WDATA** (for write) or **RDATA** (for read) based on the mode.
- **WDATA** – Serially sends the write data to the bus. After completion, returns to **IDLE**.
- **RDATA** – Receives serial read data from the bus (`brdata`) and stores it in an internal register. If a split occurs (`msplit = 1`), moves to the **SPLIT** state. Once all bits are received, asserts `mrvalid` and returns to **IDLE**.
- **SPLIT** – Waits for the arbiter to grant access again for a previously split transaction (`msplit = 0` and `mbgrant = 1`). Resumes read transaction by returning to **RDATA**.

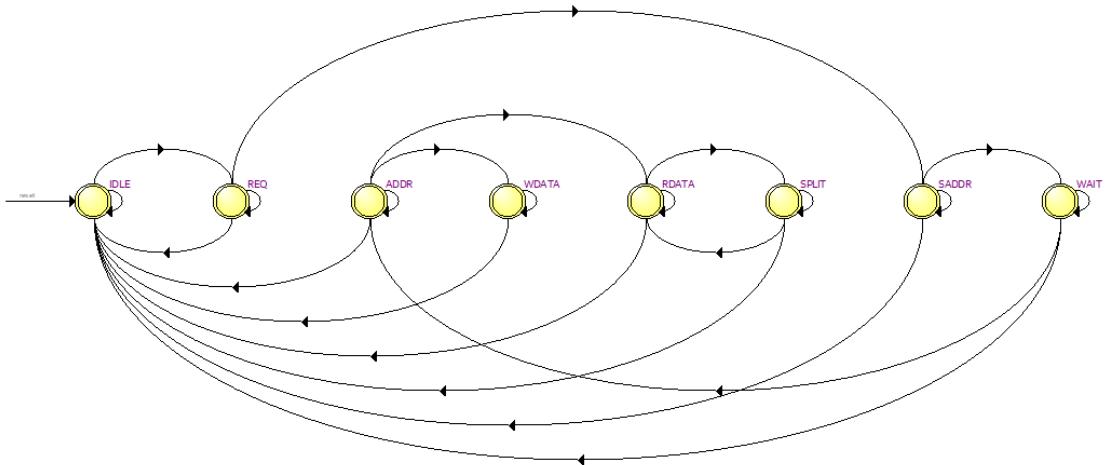


Figure 6: Master interface State Machine

2.7 Slave Interface

The Slave Interface module converts serial data from the bus into parallel signals for the slave memory and vice versa. It manages read and write operations, communicates readiness to the bus, and handles split transactions when the slave cannot immediately service a request. For split-capable slaves, it asserts a split signal and resumes the transaction once the bus grants access.

Table 9: Slave Interface I/O Signal Description

Signal	Dir	Width	Description
clk	In	1	System clock for synchronous operation.
rstn	In	1	Active-low reset.
mem_addr	Out	ADDR_WIDTH	Parallel memory address sent to the slave memory.
mem_wen	Out	1	Write enable for slave memory.
mem_ren	Out	1	Read enable for slave memory.
mem_rdata	In	DATA_WIDTH	Data read from the slave memory.
mem_rvalid	In	1	Indicates that mem_rdata is valid.
mem_wdata	Out	DATA_WIDTH	Parallel write data sent to the slave memory.
mem_wvalid	Out	1	Indicates that mem_wdata is valid.
bwdata	In	1	Serial data from the bus (address or write data).
brdata	Out	1	Serial data sent back to the bus (read data).
bmode	In	1	Bus mode signal (0 = read, 1 = write).
bwvalid	In	1	Indicates that bwdata is valid.
brvalid	Out	1	Indicates that brdata is valid.
sready	Out	1	Indicates the slave is ready to accept a transaction.
split_grant	In	1	Indicates that a previously split transaction can resume.
ssplit	Out	1	Asserted when a split transaction is in progress.

The Slave Interface operates using a six-state FSM:

- **IDLE** – Waits for a valid serial transaction from the bus (**bwvalid**). If a valid bit is received, captures the first address bit and transitions to **ADDR**. Asserts **sready** when ready to accept a new transaction.

System Bus Design Report

- **ADDR** – Receives the full slave memory address serially from the bus. Once all address bits are received, determines the mode from `bmode` and transitions to **WDATA** for write or **RDATA** for read operations.
- **WDATA** – Receives serial write data from the bus. Accumulates data in an internal register until a full word is received, then moves to **WDATA_MEM**.
- **WDATA_MEM** – Writes the accumulated data to the slave memory. Asserts `mem_wvalid` and `mem_wen` during the write, then returns to **IDLE**.
- **RDATA** – Reads data from the slave memory. If split is enabled (`SPLIT_EN`) and a split is required, asserts `ssplit` and waits until the bus grants access (`split_grant`) before proceeding. Once data is ready, transitions to **RDATA_BUS**.
- **RDATA_BUS** – Sends the read data serially back to the bus (`brdata`), asserting `brvalid`. After all bits are transmitted, returns to **IDLE**.

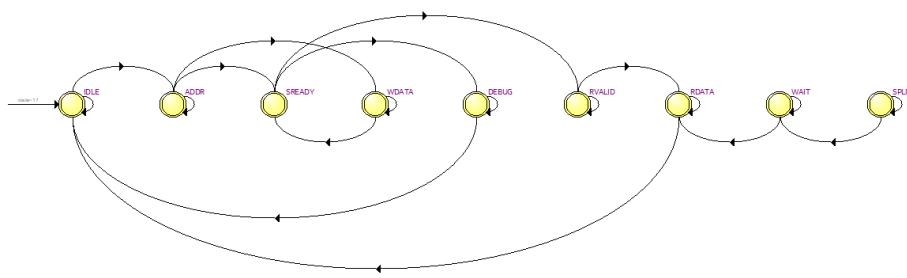


Figure 7: Slave interface State Machine

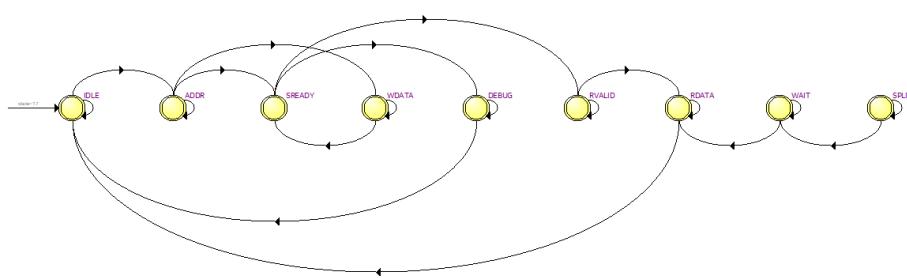


Figure 8: Slave Port State Machine

3 Bus Bridge Design

3.1 Design Parameters

Table 10: Configurable Design Parameters

Parameter	Default Value	Description
ADDR_WIDTH	16 bits	Total address bus width
DATA_WIDTH	8 bits	Data bus width
SLAVE_MEM_ADDR_WIDTH	14 bits	Slave memory address space
BB_ADDR_WIDTH	16 bits	Bus bridge address width

3.2 I/O Port Summary

Table 11: Top-Level I/O Ports of `demo_top_bb`

Port	Direction	Width	Description
clk	Input	1	System clock
rstn	Input	1	Active-low synchronous reset
start	Input	1	Transaction start trigger (falling edge)
mode	Input	1	0 = Read, 1 = Write
m_u_rx	Input	1	Master UART receive line
s_u_rx	Input	1	Slave UART receive line
m_u_tx	Output	1	Master UART transmit line
s_u_tx	Output	1	Slave UART transmit line
LED	Output	8	Debug LED output
ready	Output	1	System ready for new transaction

3.3 UART Implementation

The UART interface is used as the communication channel between Bus A and Bus B through the bridge. Operating with a system clock of 50 MHz and a clock divider value of 5208, the UART achieves an approximate baud rate of 9600 bps:

$$\text{Baud Rate} = \frac{f_{clk}}{\text{CLOCKS_PER_PULSE}} = \frac{50 \text{ MHz}}{5208} \approx 9600 \text{ bps} \quad (1)$$

During a transaction, the UART transfers two distinct message types: a REQUEST message from Bus A to Bus B, and a RESPONSE message returned by Bus B after completing the operation. The REQUEST frame is 32 bits wide and carries the full command information, while the RESPONSE frame is 16 bits wide and returns the result of the transaction. To ensure smooth operation and accommodate burst command sequences, an 8-entry FIFO is used for buffering incoming REQUEST messages.

Message Encoding

- **REQUEST (RX) Payload – 32 bits:** Consists of `padding[7]` + `mode[1]` + `data[8]` + `addr[16]`. This frame communicates the target address, transaction type (read/write), and write data (if applicable).

- **RESPONSE (TX) Payload – 16 bits:** Contains the read data returned from Bus B or status information for write operations.

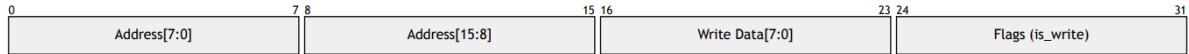


Figure 9: Request Message Format

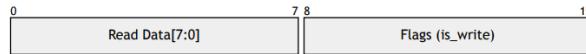


Figure 10: Response Message Format

3.4 Bus Bridge Master

The bus bridge master receives UART commands from an external source and converts them to bus transactions. This allows temporary storage of multiple REQUEST frames to avoid stalling the initiator side.

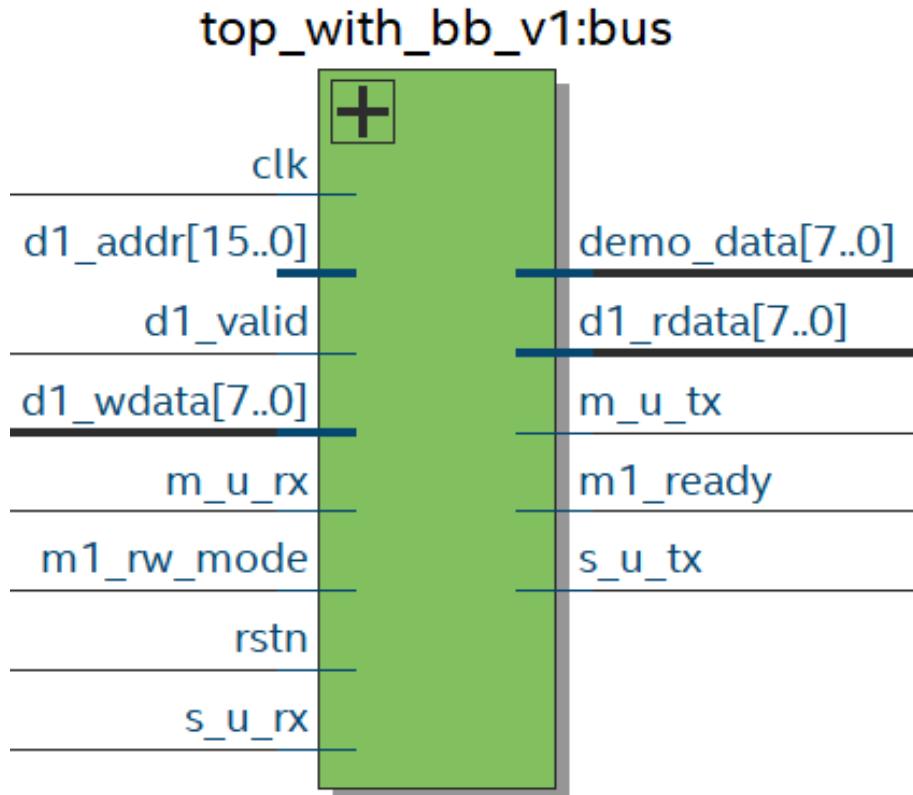


Figure 11: Bus system top module

3.5 Bus Bridge Slave

The bus bridge slave converts bus transactions to UART commands for remote memory access.

Table 12: Bus Bridge Slave States

State	Encoding	Description
IDLE	2'b00	Wait for memory access request
WSEND	2'b01	Transmit write data via UART
RSEND	2'b10	Transmit read address via UART
RDATA	2'b11	Wait for read data from UART

3.6 Memory Map for Bus A

Table 13: Slave Memory Map for Bus A

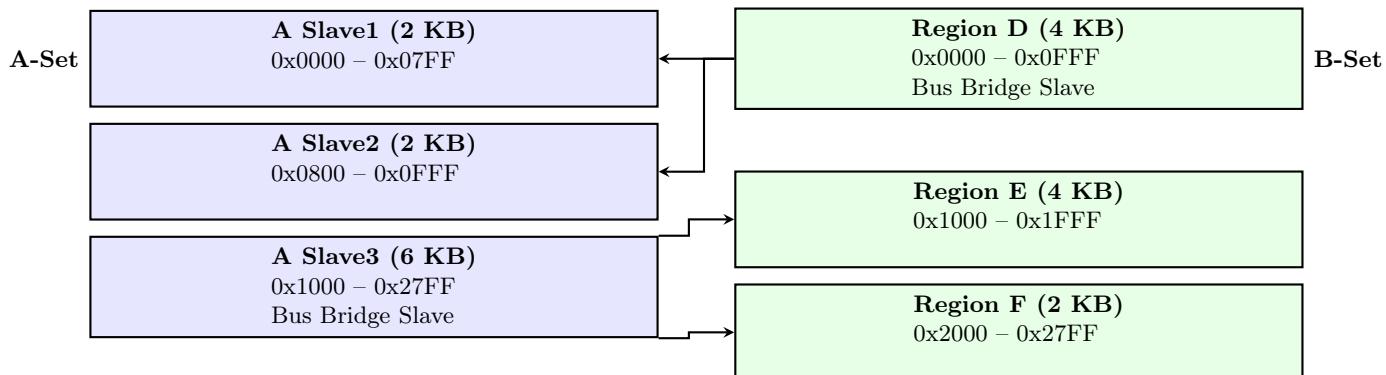
Slave	Base Address	End Address	Size	Type
Slave 1	0x0000	0x07FF	2 KB	Local BRAM
Slave 2	0x4000	0x4FFF	4 KB	Local BRAM
Slave 3	0x8000	0xA7FF	10 KB	UART Bridge

3.7 Memory Map for Bus B

Table 14: Slave Memory Map for Bus B

Slave	Base Address	End Address	Size	Type
Slave 1	0x8000	0x8FFF	4 KB	Local BRAM
Slave 2	0x9000	0x9FFF	4 KB	Local BRAM
Slave 3	0xA000	0xA7FF	2 KB	Local BRAM

4 Address Map for Unified Design



5 Timing Diagrams

System Bus Design Report

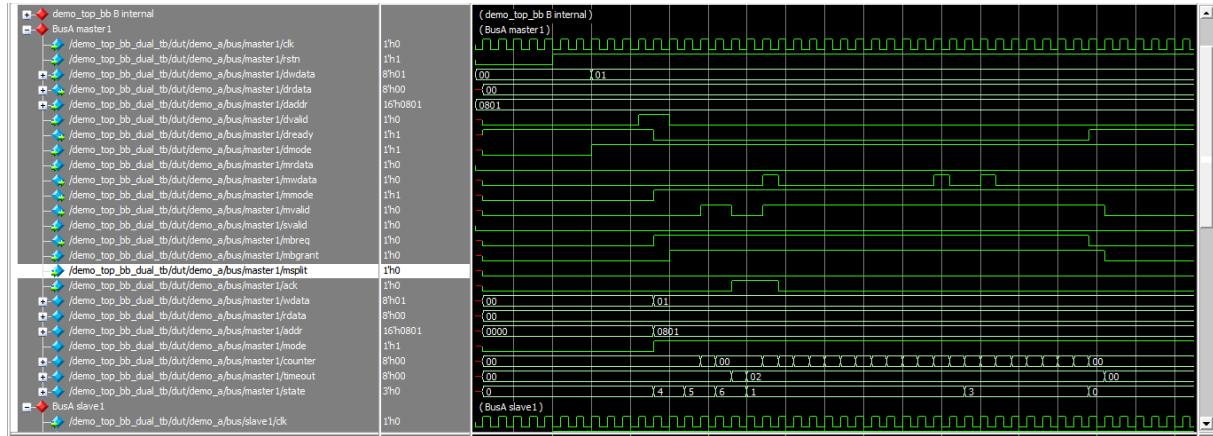


Figure 12: Master 1 write data to slave

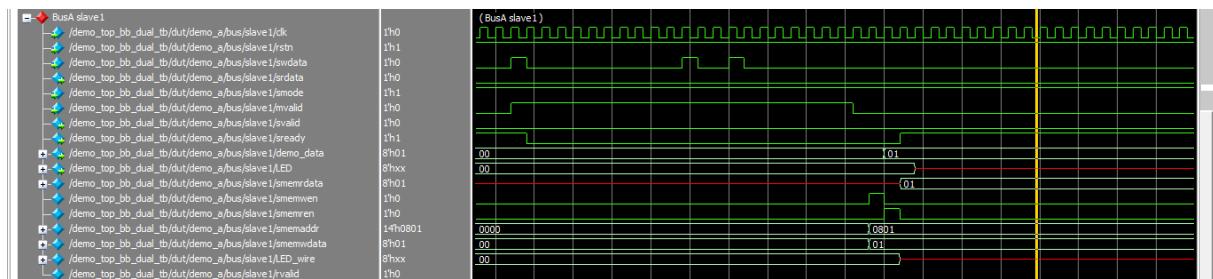


Figure 13: Slave 1 receives data

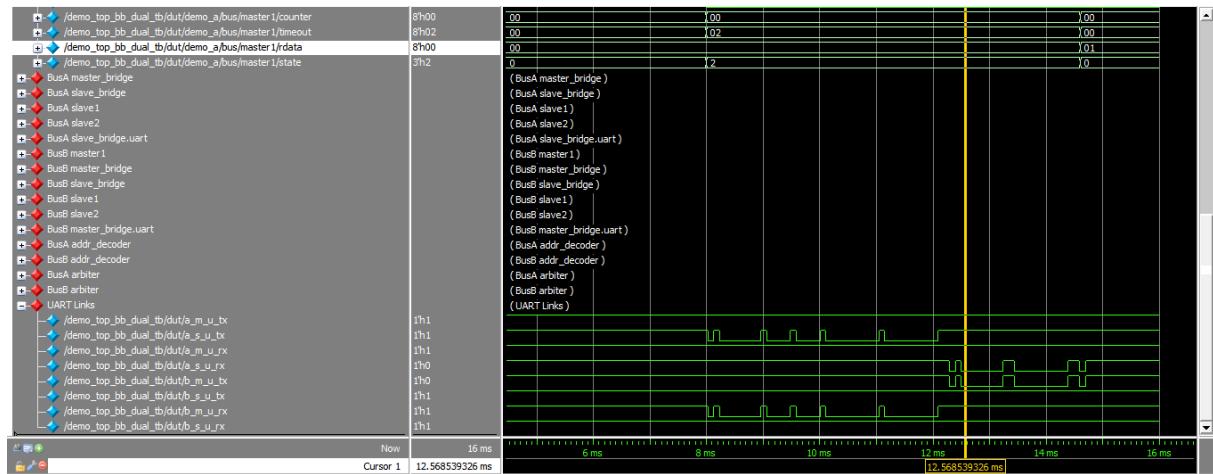


Figure 14: Timing Diagram for bus A writing and reading from bus B

6 Timing Characteristics

Table 15: Transaction Timing Parameters

Parameter	Value	Notes
Device Address Transfer	4 cycles	Serial, MSB first
Memory Address Transfer	12 cycles	Serial, LSB first
Data Transfer	8 cycles	Serial, LSB first
Total Write Transaction	~24 cycles	Without arbitration wait
Total Read Transaction	~32 cycles	Without arbitration wait

6.1 UART Timing

$$\text{Baud Rate} = \frac{50 \text{ MHz}}{5208} \approx 9600 \text{ bps} \quad (2)$$

$$\text{Bit Period} = \frac{1}{9600} \approx 104 \mu\text{s} \quad (3)$$

$$32\text{-bit Frame Time} = 32 \times 104 \mu\text{s} \approx 3.33 \text{ ms} \quad (4)$$

7 Resource Summary

Table 16: Design Resource Utilization Summary

Component Type	Count	Description
Master Ports	2	1 local + 1 bridge
Slave Ports	3	2 local + 1 bridge
Block RAMs	3	32B demo + 2KB + 4KB
FIFOs	1	8-deep, 32-bit width
UART Modules	2	TX/RX pairs (asymmetric widths)
Multiplexers	4	2×MUX2 + 2×MUX3
Decoders	2	Address decoder + 3-to-1 enable
State Machines	6+	Arbiter, masters, slaves, bridges