AHB-Lite to SPI Bridge Validation Tests

1. Hardware Validation Test Using Jupyter notebook + PYNQ-Z2 board

TEST-0: AHB_CLK & SPI_CLK

Code:

```
from pynq import Overlay, MMIO, Clocks
ol = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
ol.download()
spi = MMIO(0x40000000, 0x1000)
print("FCLK0 =", Clocks.fclk0_mhz)
```

Jupyter Output: FCLK0 = 100

DSO Output:(Yellow trace: spi_clk)

(spi_clk: 6.25MHz)

TEST-1: Continuous 0xAA pattern

Code:

```
from pynq import Overlay, MMIO
ol = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
ol.download()
spi = MMIO(0x40000000, 0x1000)
spi.write(0x08, 1) # DC write data mode

while True:
    spi.write(0x00, 0xAA) # 10101010
```

DSO Output:(Yellow trace: spi_mosi, Green trace: spi_clk)

(spi_mosi: 0xAA)

TEST-2: Continuous 0x55 pattern

Code:

```
from pynq import Overlay, MMIO
ol = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
ol.download()

spi = MMIO(0x40000000, 0x1000)

spi.write(0x08, 1) # DC write data mode

while True:
    spi.write(0x00, 0x55) # 01010101
```

DSO Output:(Yellow trace: spi_mosi, Green trace: spi_clk)

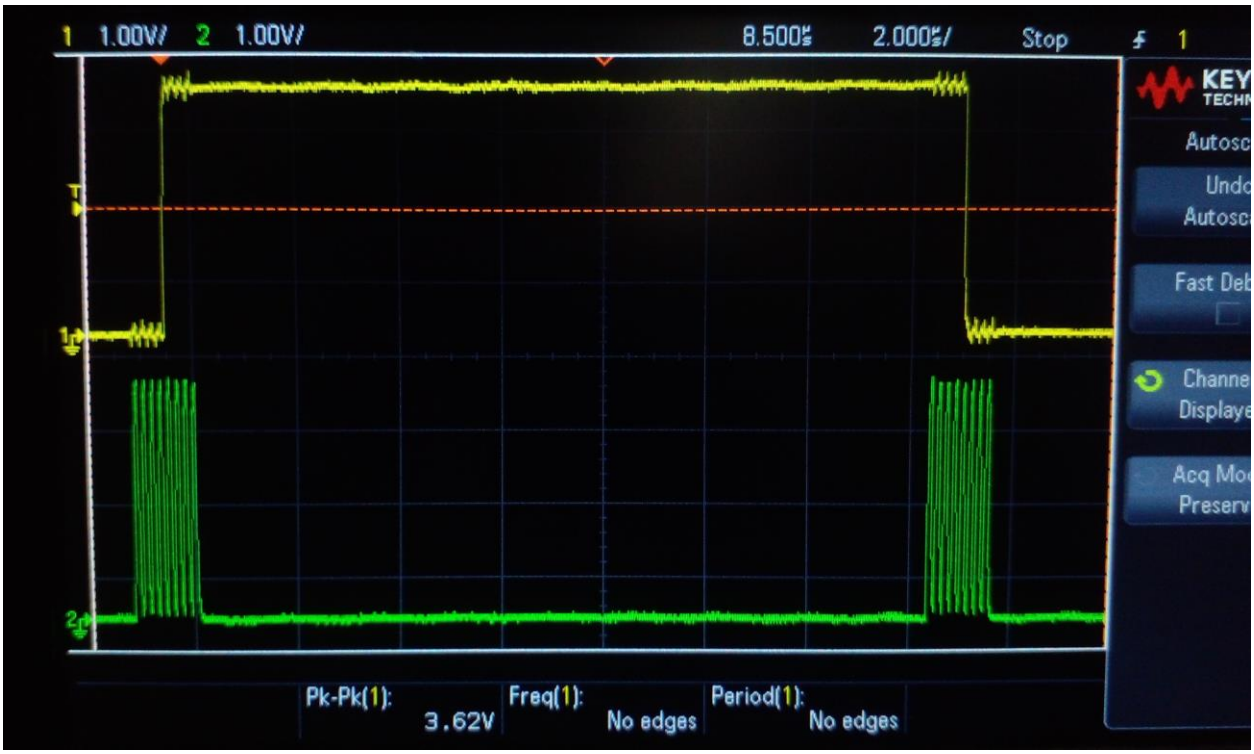
(spi_mosi: 0x55)

TEST-3: Toggle between two bytes (checking shift accuracy)

Code:

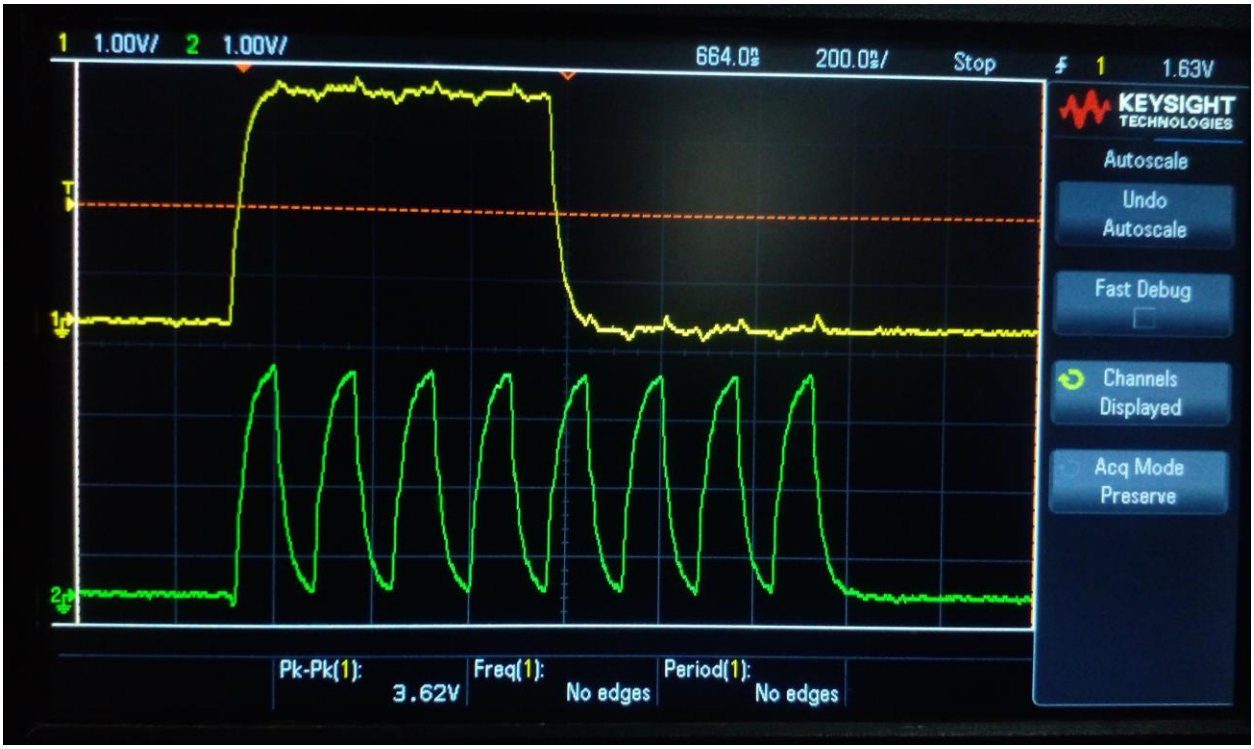
```
from pynq import Overlay, MMIO
ol = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
ol.download()

spi = MMIO(0x40000000, 0x1000)

spi.write(0x08, 1) # DC write data mode

while True:
    spi.write(0x00, 0x0F) # 00001111
    spi.write(0x00, 0xF0) # 11110000
```

DSO Output:(Yellow trace: spi_mosi, Green trace: spi_clk)



(spi_mosi: 0x0F_0xF0_Combined)

(spi_mosi: 0xF0_Separate)



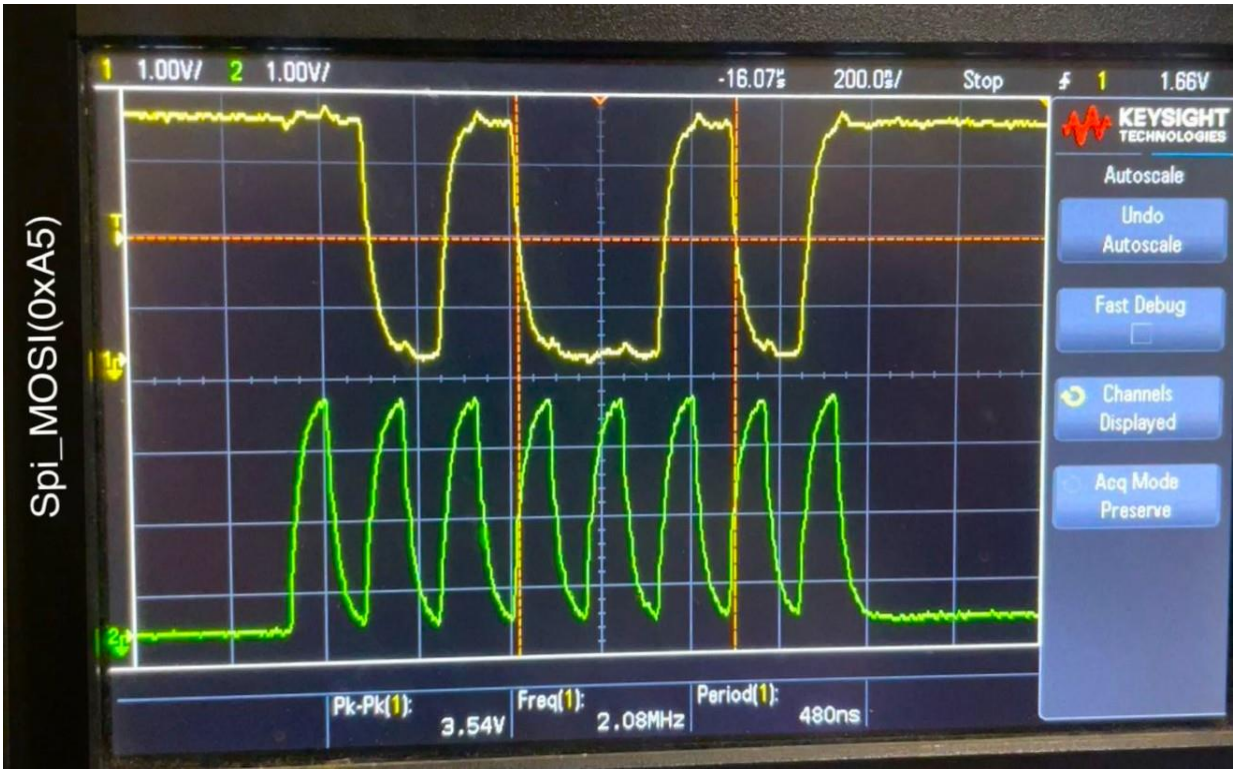
(spi_mosi: 0xF0_Separate)

TEST-4: Long burst pattern (for persistent waveform)

Code:

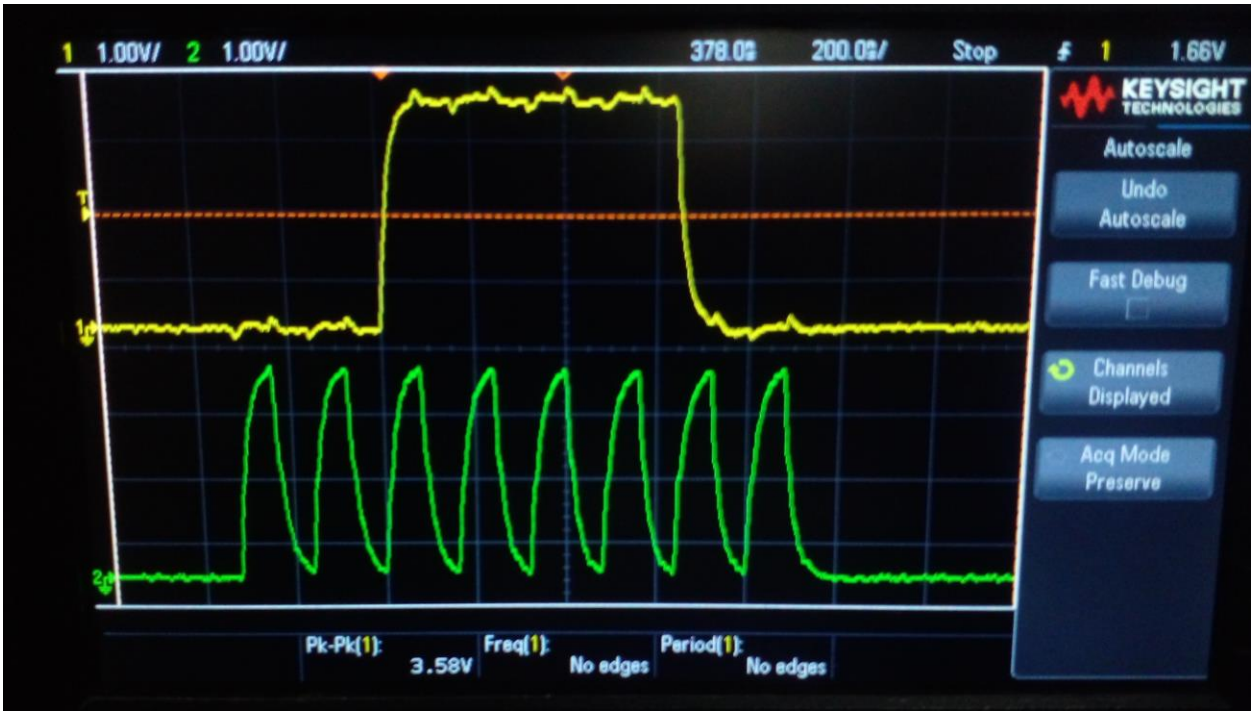
```
from pynq import Overlay, MMIO
ol = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
ol.download()

spi = MMIO(0x40000000, 0x1000)

spi.write(0x08, 1) # DC write data mode

while True:
    spi.write(0x00, 0xA5) # 10100101
    spi.write(0x00, 0x5A) # 01011010
    spi.write(0x00, 0x3C) # 00111100
    spi.write(0x00, 0xC3) # 11000011
```

DSO Output:(Yellow trace: spi_mosi, Green trace: spi_clk)

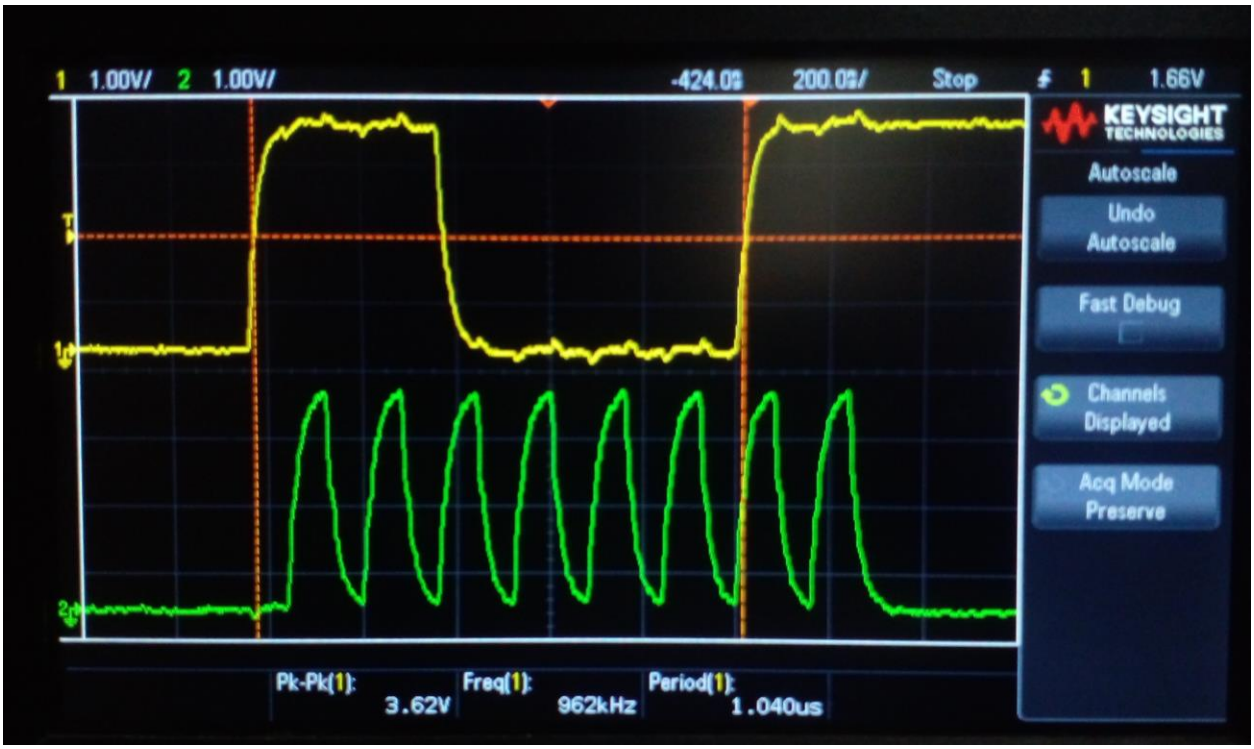


(spi_mosi: 0xA5_Separate)

(spi_mosi: 0x5A_Separate)



(spi_mosi: 0x3C_Separate)



(spi_mosi: 0xC3_Separate)

TEST-5: CS verification loop

Code:

```
from pynq import Overlay, MMIO
import time
ol = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
ol.download()

spi = MMIO(0x40000000, 0x1000)

spi.write(0x08, 1) # DC write data mode

while True:
    spi.write(0x00, 0xF0)
    time.sleep(0.0001)
```

DSO Output:(Yellow trace: spi_cs, Green trace: spi_clk)

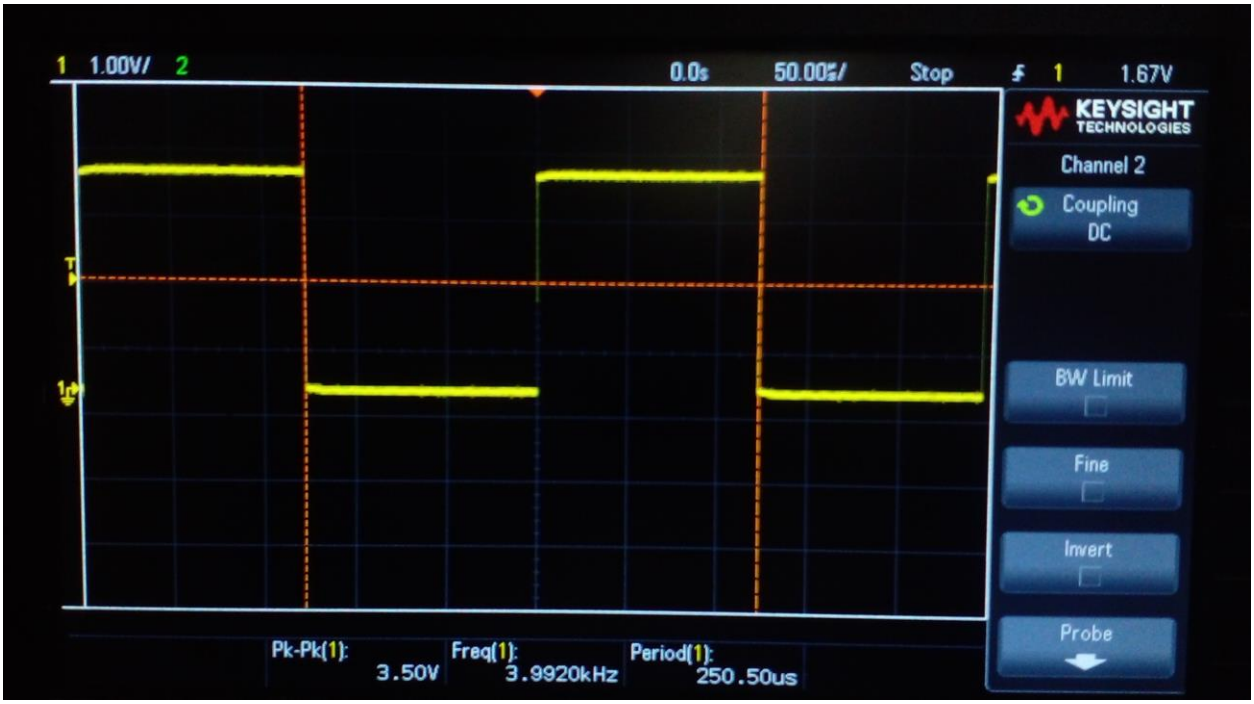
(spi_cs)

TEST-6: DC pin toggle at human-visible rate

Code:

```
from pynq import Overlay, MMIO
import time
ol = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
ol.download()

spi = MMIO(0x40000000, 0x1000)

while True:
    spi.write(0x08, 0) # DC write command mode
    time.sleep(0.000125)
    spi.write(0x08, 1) # DC write data mode
    time.sleep(0.000125)
```

DSO Output:(Yellow trace: spi_dc)



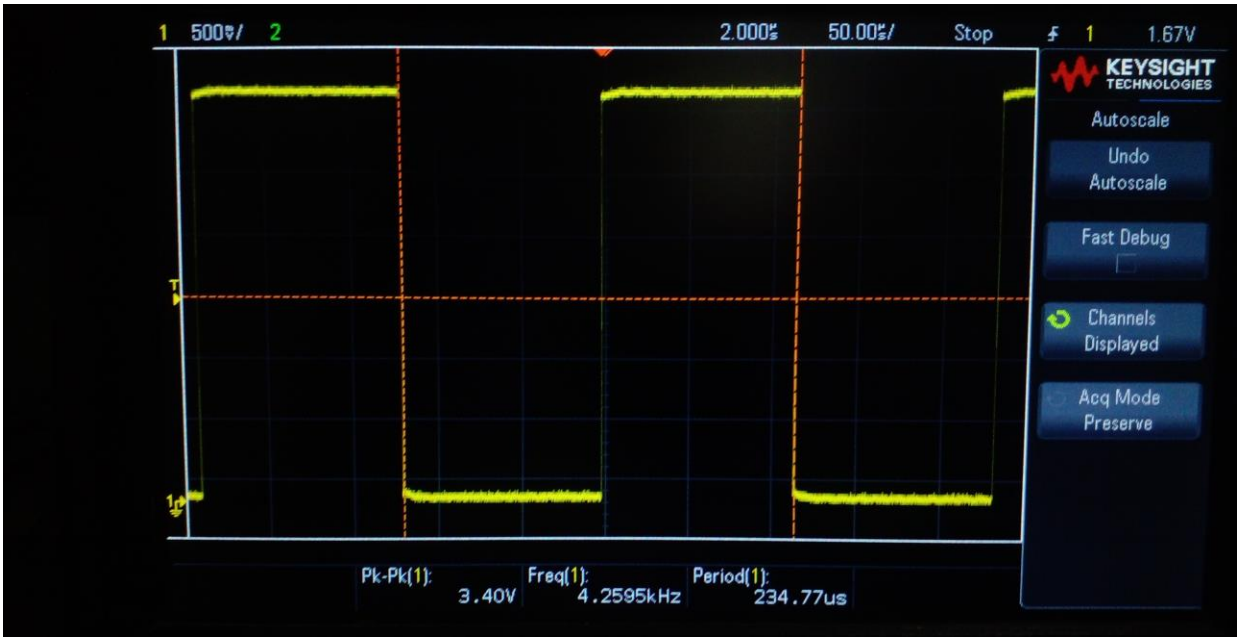
(spi_dc: Toggling at 4kHz)

TEST-7: Reset pin toggle burst

Code:

```
from pynq import Overlay, MMIO
import time
ol = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
ol.download()

spi = MMIO(0x40000000, 0x1000)

while True:
    spi.write(0x18, 0) # write spi slave reset
    time.sleep(0.000125)
    spi.write(0x18, 1) # write spi slave reset
    time.sleep(0.000125)
```

DSO Output:(Yellow trace: spi_rst)



(spi_rst: Toggling at 4kHz)

2. Software Validation Test Using Jupyter notebook + PYNQ-Z2 board

Test 1: Register Read/Write Sanity Check (DC Register)

Code:

```
from pynq import Overlay, MMIO
import time

overlay = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
base_addr = overlay.ip_dict['top_pynq_0']['phys_addr']
spi = MMIO(base_addr, 0x1000)

print("Base address =", hex(base_addr))
```

Jupyter Output: Base address = 0x40000000

```
print("DC initial =", spi.read(0x08))

spi.write(0x08, 0) # DC write command mode
print("DC after write 0 =", spi.read(0x08)) # DC read

spi.write(0x08, 1) # DC write data mode
print("DC after write 1 =", spi.read(0x08)) # DC read
```

Jupyter Output:

```
DC initial = 0
DC after write 0 = 0
DC after write 1 = 1
```

Test 2: Slave Reset Register Validation

Code:

```
from pynq import Overlay, MMIO
import time

overlay = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
base_addr = overlay.ip_dict['top_pynq_0']['phys_addr']
spi = MMIO(base_addr, 0x1000)

print("Base address =", hex(base_addr))
```

Jupyter Output: Base address = 0x40000000

```
spi.write(0x18, 0) # slave reset write
time.sleep(0.05)
print("RESET =", spi.read(0x18)) # slave reset read

spi.write(0x18, 1) # slave reset write
time.sleep(0.05)
print("RESET =", spi.read(0x18)) # slave reset read
```

Jupyter Output:
RESET = 0
RESET = 1

Test 3: FIFO Write & TX Counter Validation

Code:

```
from pynq import Overlay, MMIO
import time

overlay = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
base_addr = overlay.ip_dict['top_pynq_0']['phys_addr']
spi = MMIO(base_addr, 0x1000)

print("Base address =", hex(base_addr))
```

Jupyter Output: Base address = 0x40000000

```
start = spi.read(0x10) # Initial Read TX_CNT

for i in range(50):
    spi.write(0x00, i) # TX FIFO write
    time.sleep(0.001)

end = spi.read(0x10) # Final Read TX_CNT
print("Expected ~50, Got =", end - start)
```

Jupyter Output: Expected ~50, Got = 50

Test 4: FIFO Status Flags Check

Code:

```
from pynq import Overlay, MMIO
import time

overlay = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
base_addr = overlay.ip_dict['top_pynq_0']['phys_addr']
spi = MMIO(base_addr, 0x1000)

print("Base address =", hex(base_addr))
```

Jupyter Output: Base address = 0x40000000

```
flags = spi.read(0x0C) # Read FLAGS dc, fifo full, fifo empty

fifo_empty = flags & 0x1
fifo_full  = (flags >> 1) & 0x1
dc_state   = (flags >> 2) & 0x1

print("FIFO_EMPTY =", fifo_empty)
print("FIFO_FULL  =", fifo_full)
print("DC_STATE   =", dc_state)
```

Jupyter Output:

```
FIFO_EMPTY = 1
FIFO_FULL  = 0
DC_STATE   = 0
```

Test 5: SPI FSM Activity Verification

Code:

```
from pynq import Overlay, MMIO
import time

overlay = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
base_addr = overlay.ip_dict['top_pynq_0']['phys_addr']
spi = MMIO(base_addr, 0x1000)

print("Base address =", hex(base_addr))
```

Jupyter Output: Base address = 0x40000000

```
for _ in range(5):
    spi.write(0x00, 0xAA)
    time.sleep(0.01)
    print("FSM state =", spi.read(0x14)) # Read FSM State 0 – IDLE, 1 – POP FIFO, 2 -
LAUNCH SPI, 3 – WAIT
```

Jupyter Output:

```
FSM state = 0
FSM state = 0
FSM state = 0
FSM state = 0
FSM state = 0
```

Test 6: Continuous Streaming Stress Test

Code:

```
from pynq import Overlay, MMIO
import time

overlay = Overlay("/home/xilinx/jupyter_notebooks/ahb_spi_9dec/system_wrapper.bit")
base_addr = overlay.ip_dict['top_pynq_0']['phys_addr']
spi = MMIO(base_addr, 0x1000)

print("Base address =", hex(base_addr))
```

Jupyter Output: Base address = 0x40000000

```
start = spi.read(0x10) TX_CNT

for _ in range(200):
    spi.write(0x00, 0x55) # Write to TX FIFO

time.sleep(0.2)
end = spi.read(0x10) # Read total transmitted bytes

print("Bytes transmitted =", end - start)
```

Jupyter Output: Bytes transmitted = 200

Register Map and Functional Description

Table 1: Memory-Mapped Register Specification (Base Address: `0x4000_0000`)

Register Name	Offset	Access	Bits / Width	No. of States	Description
TX_FIFO	`0x00`	Write-only	`[7:0]`	256	Writing an 8-bit value pushes one byte into the transmit FIFO for SPI transmission
DC	`0x08`	Read/Write	`[0]`	2	Selects SPI transfer type: Command or Data
FLAGS	`0x0C`	Read-only	`[2:0]`	4	Indicates FIFO and DC status
TX_CNT	`0x10`	Read-only	`[31:0]`	2^{32}	Counts total number of bytes transmitted over SPI
FSM	`0x14`	Read-only	`[1:0]`	4	Indicates current SPI Master FSM state
RESET	`0x18`	Read/Write	`[0]`	2	Controls hardware reset of SPI slave device

Detailed Bit-Level Description

1. TX_FIFO Register

Field	Bits	States	Meaning
TX_DATA	`[7:0]`	0–255	8-bit SPI data byte to be transmitted

Notes:

- Each write enqueues **one byte** into FIFO
- FIFO depth = 16 bytes
- Writes are stalled when FIFO is full (`HREADY = 0`)

2. DC (Data / Command) Register

Bit	Name	Value	Meaning
0	DC	`0`	Command mode (TFT command)
0	DC	1	Data mode (TFT pixel / data)

Number of states: 2

3. FLAGS Register

Bit	Name	Value	Meaning
0	FIFO_EMPTY	1	FIFO is empty
1	FIFO_FULL	1	FIFO is full
2	DC_STATE	0/1	Current DC register value
[31:3]	Reserved	—	Reserved (read as 0)

Number of states:

- FIFO_EMPTY: 2
- FIFO_FULL: 2
- DC_STATE: 2
- Total observable combinations: **4 meaningful states**

4. TX_CNT (Transmit Counter) Register

Field	Bits	States	Meaning
TX_COUNT	[31:0]	$0 \rightarrow 2^{32}-1$	Counts number of bytes successfully transmitted

Notes:

- Incremented after each completed SPI byte
- Used for throughput and correctness validation

5. FSM (SPI FSM Debug) Register

Value	FSM State Name	Description
00	IDLE	SPI idle, waiting for FIFO data
01	POP	Byte popped from FIFO
10	LAUNCH	SPI transmission started
11	WAIT	Waiting for SPI byte completion

Bits: [1:0]
Number of states: 4

6. RESET Register

Bit	Name	Value	Meaning
0	RESET_N	`0`	SPI slave held in reset
0	RESET_N	`1`	SPI slave active (normal operation)

Number of states: 2

Polarity: Active-Low (internally)

Summary Table

Register	Width	R/W	States
TX_FIFO	8-bit	W	256
DC	1-bit	R/W	2
FLAGS	3-bit	R	4
TX_CNT	32-bit	R	2^{32}
FSM	2-bit	R	4
RESET	1-bit	R/W	2