Design Verification Project

July-Aug'25

Chen Yong Seow

DV Project Agenda

Day	Agenda
Day1	Revise and complete sample codes and labs - submit SV and UVM mini projects - review the SV questionnaires answer
Day2 – Day6	DV Project – planning and execution a. SPI Controller
Day7	Present DV project – Test Plan, Test bench, test reports

Project Team Formation

10 mins

What you will learn

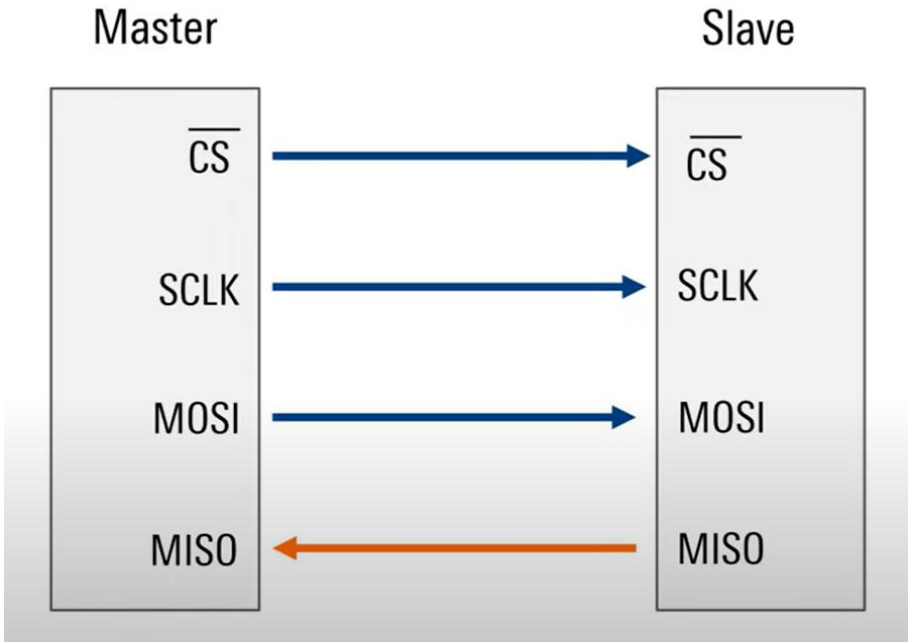
- Apply the knowledge and skills learnt throughout the training on the project.
- Participants undertake a verification project from start to finish.

SPI: Serial Peripheral Interface

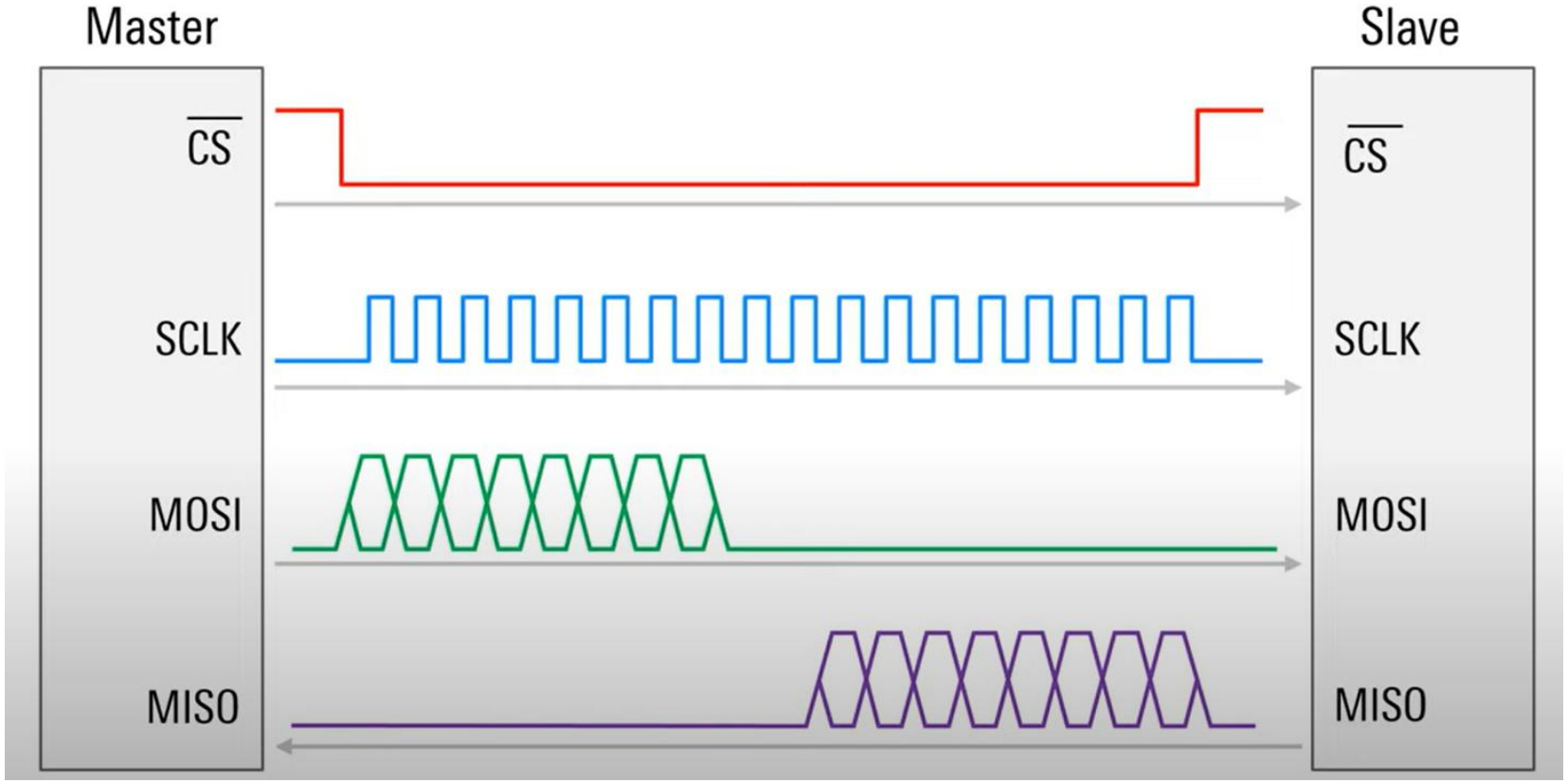
- SPI specifications: Serial Peripheral Interface – Wikipedia
- Four wire serial interface
- Developed by Motorola in the 1980s
  - Speed improvement over UART/I2C
  - Supports full-duplex communications
- Very loosely "defined"
- Often used for data transfer between a (smart) controller and a less smart) peripheral device
  - Common applications include SD cards, LCD, ADC/DAC, Flash memory, EEPROM, RTC, game controllers, sensors, etc.

SPI: Basic components/nomenclature

- One master and one or more slaves
  - (Up to) 4 wires between them
- CSn: Chip Select
  - Chooses communication target
- SCLK: Synchronous Clock
  - Provides timing/synchronization
- MOSI: Master Out Slave In
  - Data transmitted by master
- MISO: Master In Slave Out
  - Data received by master
- Many different names for these

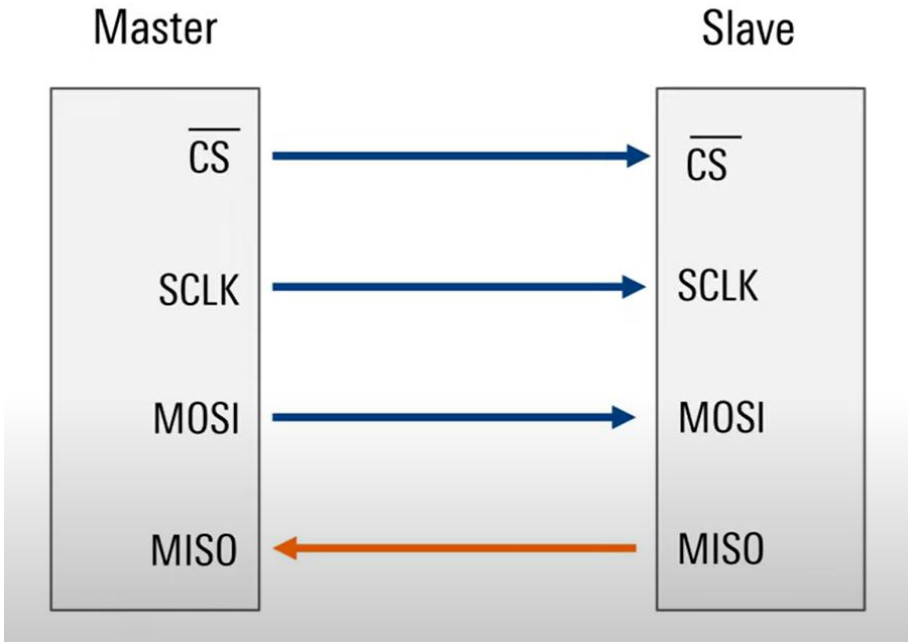


Overview of SPI protocol



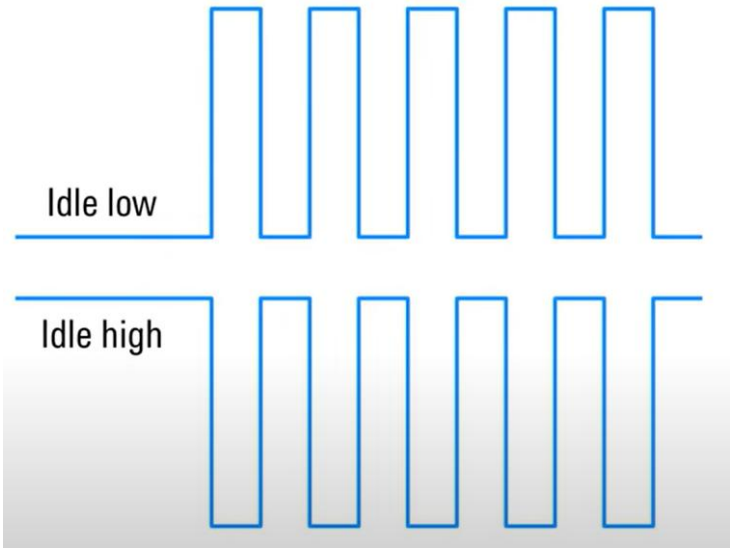
SPI: Chip Select (CSn)

- Also commonly called “Slave Select” (SSn)
- Typically, active low
- Pulled low to select the target (slave) for communication
  - Slave then listens for SCLK and MOSI
  - Kept low until communication is complete
- Simple way to address targets
  - Unlike I2C which uses addresses
- A single or multiple CSn lines can be used to address multiple devices



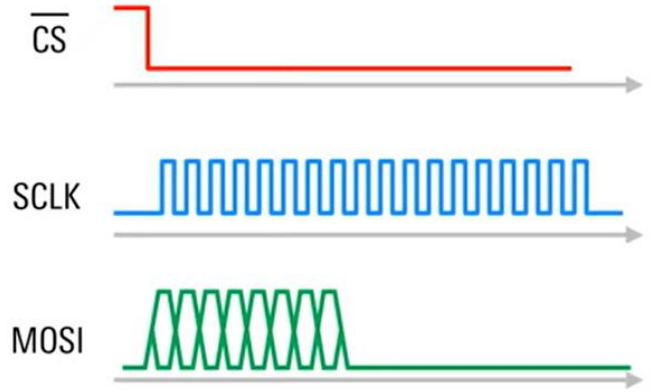
SPI: Synchronous Clock (SCLK)

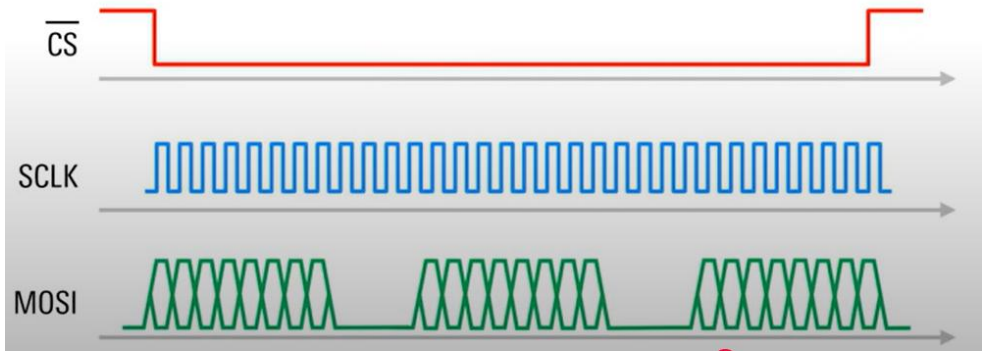
- Clock signal is generated by the master
  - Slaves do not require their own clocks, even when they are transmitting data
- Speed usually into the MHz range
  - Faster then UART or I2C
- Clock indicates when data should be sampled
  - One bit is read per clock signal
- Clock can be idle low or idle high (Clock polarity)
- Data can be sampled on rising or falling edge (Clock phase)



SPI: Master Out Slave In (MOSI)

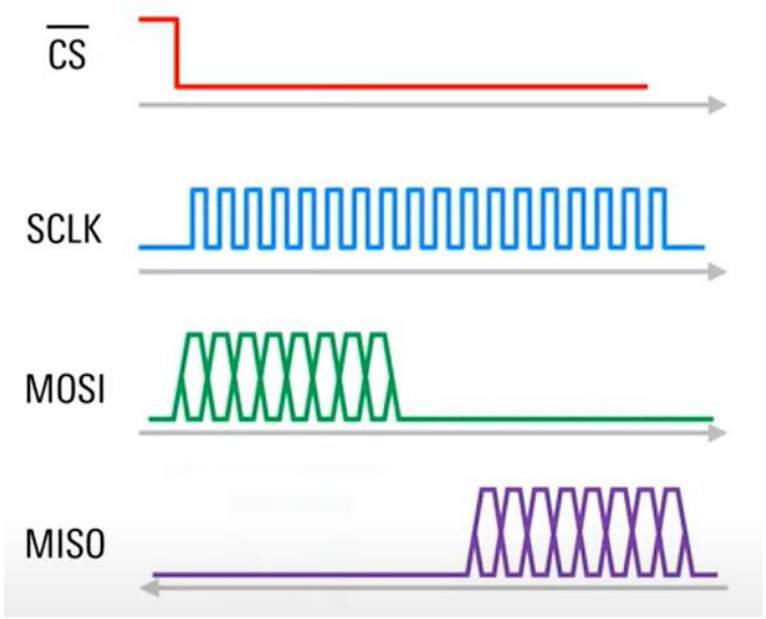
- Used to send data from master to slave(s)
  - Data is usually sent as bytes (LSB or MSB first)
- Number of bytes depends on implementation
  - Multiple bytes can be sent sequentially
  - CSn may be held low between bytes



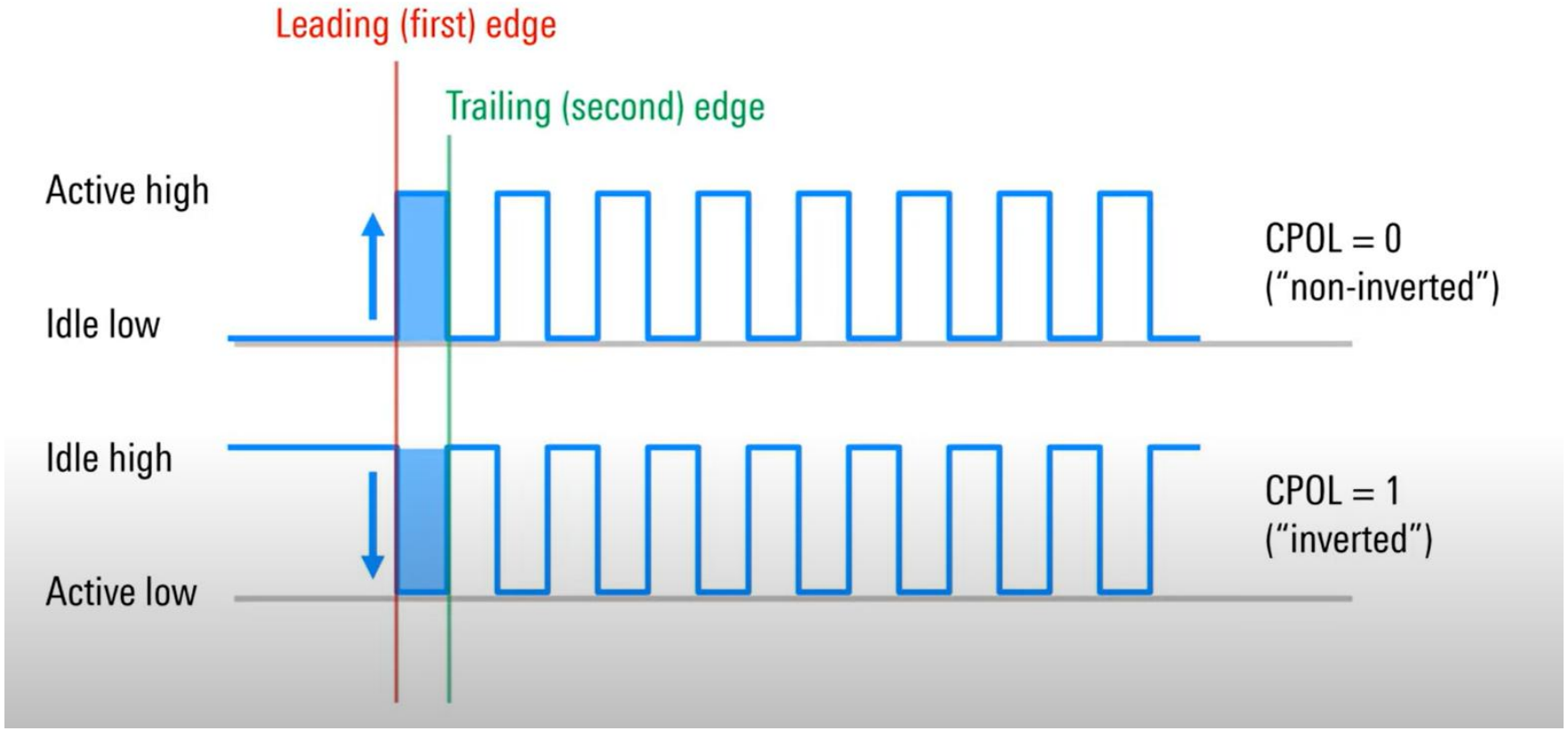


SPI: Master In Slave Out (MISO)

- Used to send data from slave(s) to master
- Not all SPI implementations use MISO
  - Some devices only receive data from master
- MISO sent as a response to data on MOSI
  - Commands, queries, etc.
- Master must know how many clock cycles to generate so the slave can send all of its data

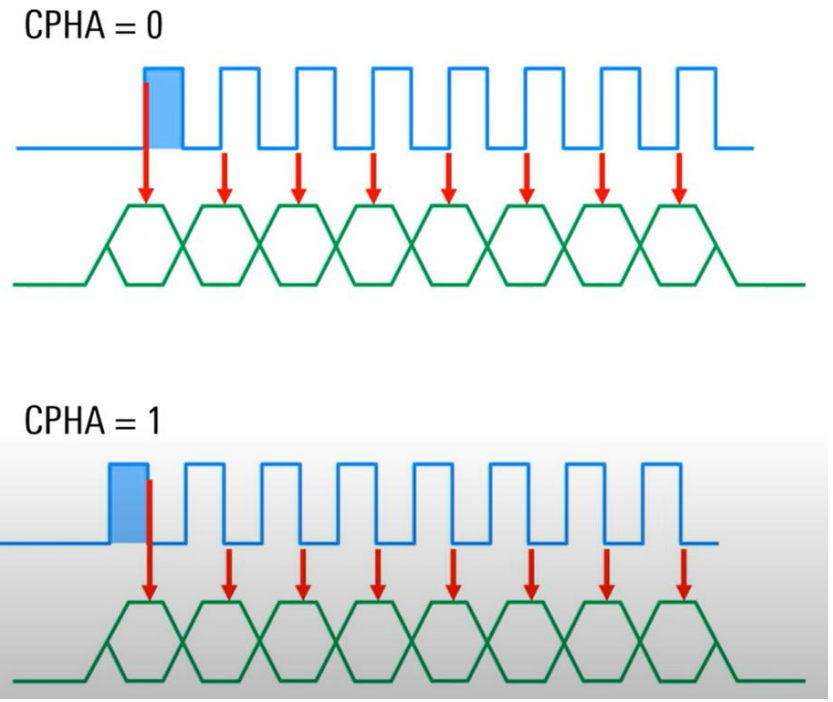


SPI: Clock Polarity (CPOL)



SPI: Clock Phase (CPHA)

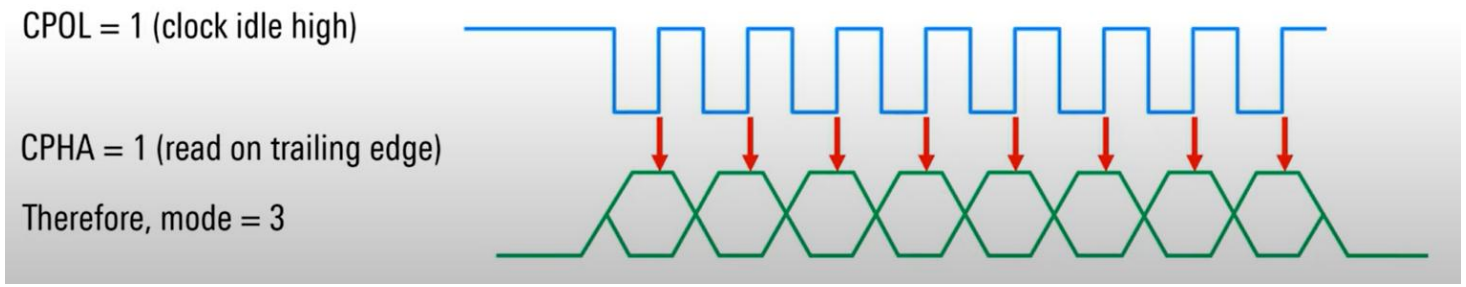
- SPI receivers may sample data on either leading or trailing clock edge
  - This is called clock phase (CPHA)
- CPHA = 0
  - Data sampled on leading (or first) edge of each clock pulse
- CPHA = 1
  - Data sampled on trailing (or second) edge of each clock pulse
- In both of these examples, CPOL = 0, that is, clock idle low



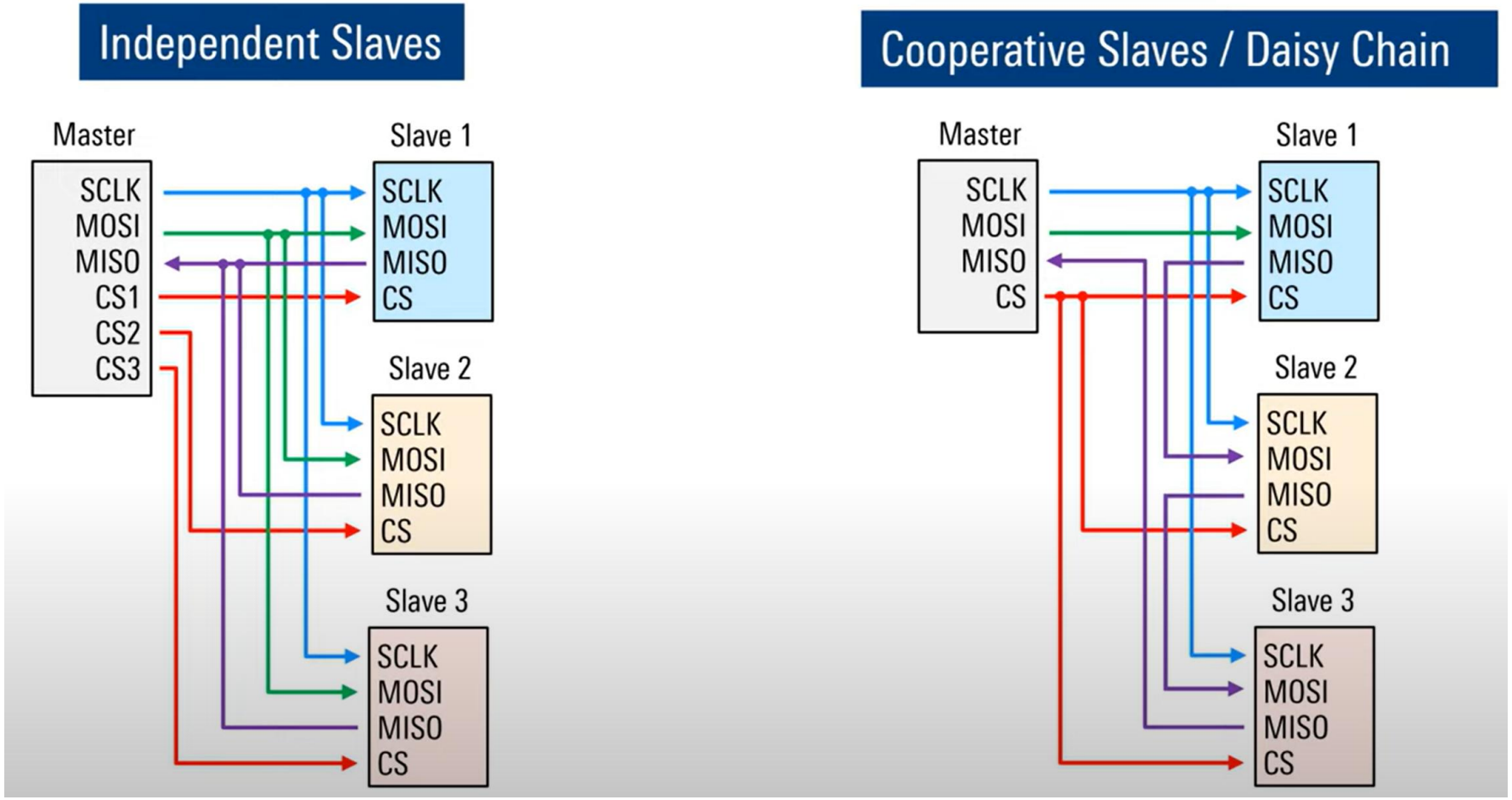
SPI: SPI modes

- Four possible combinations of CPOL and CPHA, aka SPI mode number (0-3)
- All SPI nodes must use the same mode
  - May be fixed or configurable
- Mode can often be determined by inspection

SPI Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

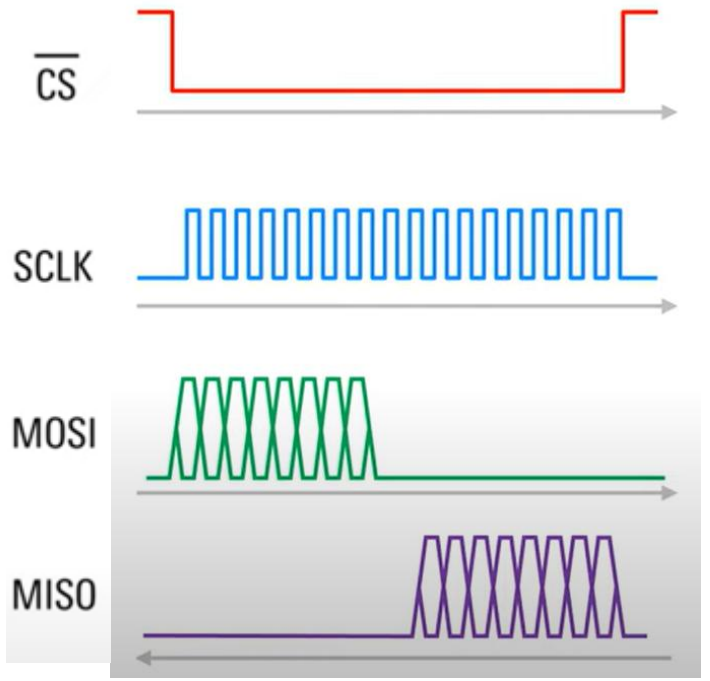


SPI: Multi-slave Configurations



SPI: Summary

- SPI is a common 4 wire serial data interface
  - Used to transfer information between a controller (master) and one or more peripheral devices (slaves)
- Four wires are CSn, SCLK, MOSI, MISO
  - CSn used for addressing, MOSI/MISO for data
- Same clock polarity and clock phase must be used in all nodes
  - Usually summarized as a mode number
- Multiple slaves can be connected either independently (CSn line per slave) or cooperatively (daisy chain)



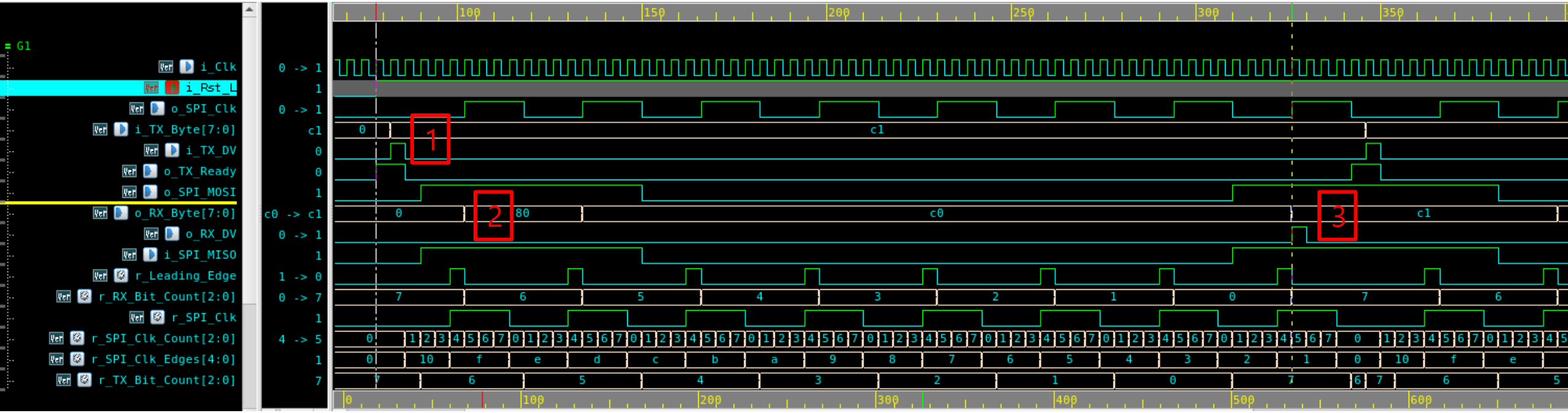
DV Project: SPI Controller

Design Specification:

1. Design consists of:
   - 1 SPI master
   - Support all 4 SPI modes, MSB first out on MOSI, MSB first in on MISO
2. Design Inputs and Outputs:
   - i_Clk: System clock (250MHz)
   - i_Rst-L: reset signal
   - i_TX_Byte[7:0]: Byte to be sent out thru MOSI
   - i_TX_DV: Data Valid pulse with i_TX_Byte
   - o_TX_Ready: Transmit Ready for next byte
   - o_RX_DV: Data Valid pulse
   - o_RX_Byte: Byte received on MISO
   - o_SPI_Clk: SPI Serial clock (31.25MHz)
   - i_SPI_MISO: SPI Master In Slave Out
   - o_SPI_MOSI: SPI Master Out Slave In

Verification requirements: Test Plan, Test code, 95+% Code & Functional Coverage, Design bug found

DV Project: SPI Controller Master Tx Waveform



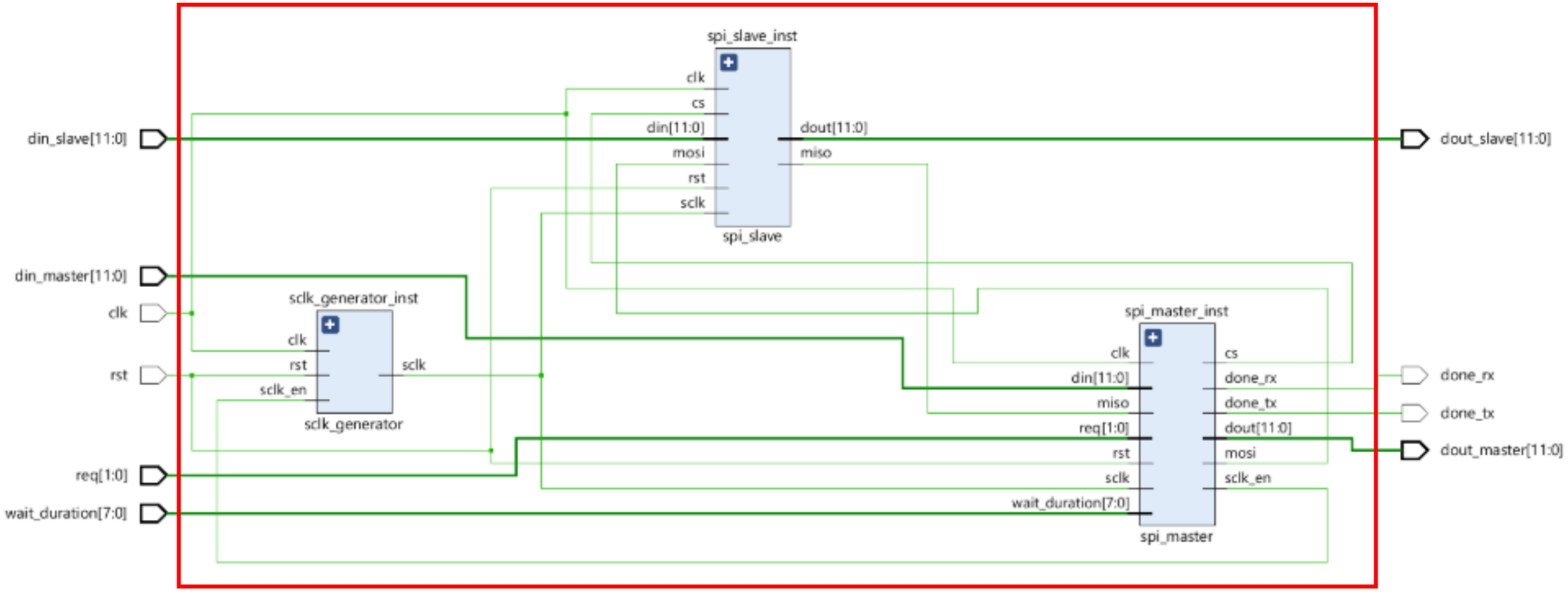
DV Project: SPI Controllers

Design Specification:

1. Design consists of:
   - 1 SPI master
   - 1 SPI Slave
   - 1 SPI Serial Clock generator, SCLK (3.7MHz)
     - SCLK_en to generate SCLK when master is ready to initiate SPI data transfer
   - Support SPI mode 1, MSB first out on MOSI, MSB first in on MISO
2. Design Inputs:
   - clk: System clock (100MHz)
   - rst: reset signal
   - req[1:0]: 00: No Operation, 01: Transmission (din_master->Master –mosi-> Slave->dout_slave), 10: Reception (din_slave->Slave-miso->master->dout_master), 11: full duplex
   - din_slave[7:0]: data to be sent out thru dout_master
   - din_master[7:0]: data to be sent out thru dout_slave
   - wait_duration [7:0]: SPI master uses this value to determine number of clk periods when CS is pulled low to when data transmission starts and for number of clk period to keep CS low after the transmission has finished.
3. Design outputs:
   - Done_rx: flag from SPI master when reception is done
   - Done_tx: flag from SPI master when transmission is done
   - Dout_slave [7:0]: data received from din_master
   - Dout_master[7:0]: data received from din_slave

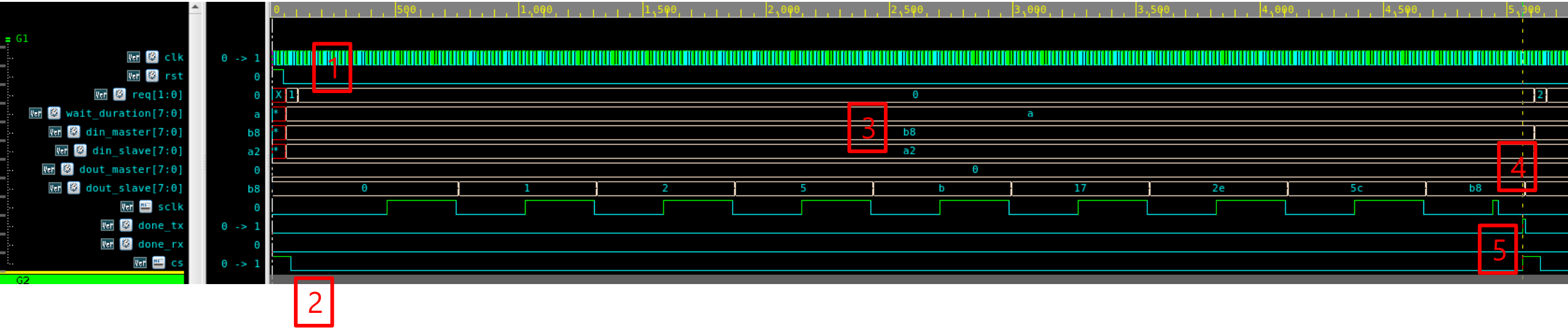
Verification requirements: Test Plan, Test code, 95+% Code & Functional Coverage, Design bug found

DV Project: Design Block diagram



DV Project: SPI Controller Master Tx Waveform

din_master->Master –mosi-> Slave->dout_slave



THANK YOU

For more information, contact Corporate Training Team at
ext 595/517/512 or email corptraining@psdc.org.my

System Verilog Testbench

A testbench helps build an environment to test and verify a design

Key components of a testbench are:

- **Generator:** generate different input stimulus to be driven to DUT (Design Under Test)
- **Driver:** drive the generated stimulus to the DUT/design
- **Monitor:** Monitor the DUT/design input/output ports to capture design activity
- **Scoreboard:** checks output from the DUT/design with expected behavior
- **Interface:** contains DUT/design signals that can be driven or monitored
- **Environment (Env):** contains all the verification components
- **Test:** contains the environment that can be configured with different configuration settings