REGISTER MAP –REFER REFERNECE FOR DETAIL ABOUT EACH PIN

Register Address: $___0

	Bit 7	6	5	4	3	2	1	Bit 0
R W	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset:	0	0	0	0	0	1	0	0

Figure 3-1 SPI Control Register 1 (SPICR1)

Read: anytime
Write: anytime

Register Address: $___1

	Bit 7	6	5	4	3	2	1	Bit 0
R W	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
Reset:	0	0	0	0	0	0	0	0

= Reserved

Figure 3-2 SPI Control Register 2 (SPICR2)

Read: anytime
Write: anytime; writes to the reserved bits have no effect

Register Address: $___2

	Bit 7	6	5	4	3	2	1	Bit 0
R W	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
Reset:	0	0	0	0	0	0	0	0

= Reserved

Figure 3-3 SPI Baud Rate Register (SPIBR)

Read: anytime
Write: anytime; writes to the reserved bits have no effect

Register Address: $___3

	Bit 7	6	5	4	3	2	1	Bit 0
R	SPIF	0	SPTEF	MODF	0	0	0	0
W								
Reset:	0	0	1	0	0	0	0	0

= Reserved

Figure 3-4 SPI Status Register (SPISR)

Read: anytime
Write: has no effect

Register Address: $___5

	Bit 7	6	5	4	3	2	1	Bit 0
R W	Bit 7	6	5	4	3	2	2	Bit 0
Reset:	0	0	0	0	0	0	0	0

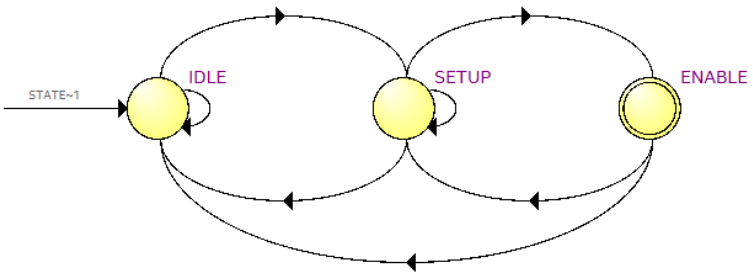
Figure 3-5 SPI Data Register (SPIDR)

Read: anytime; normally read only after SPIF is set
Write: anytime

CPOL CPHA

SPI mode	Clock polarity (CPOL)	Clock phase (CPHA)	Data is shifted out on	Data is sampled on
0	0	0	falling SCLK, and when $\overline{SS}$ activates	rising SCLK
1	0	1	rising SCLK	falling SCLK
2	1	0	rising SCLK, and when $\overline{SS}$ activates	falling SCLK
3	1	1	falling SCLK	rising SCLK

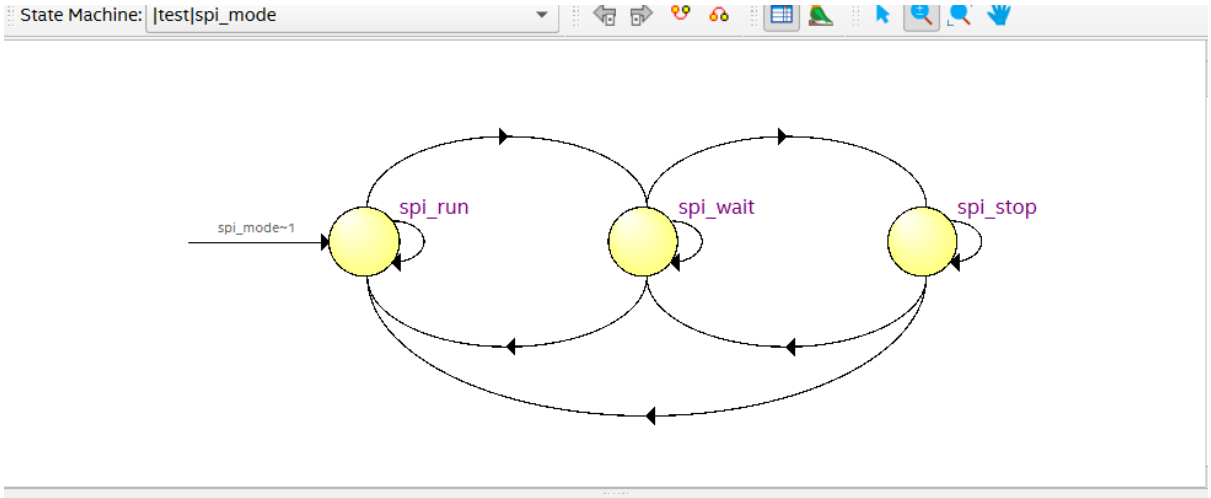
APB FSM



	Source State	Destination State	Condition
1	ENABLE	IDLE	(!PSEL)
2	ENABLE	SETUP	(PSEL)
3	IDLE	IDLE	(!PSEL) + (PSEL).(PENABLE)
4	IDLE	SETUP	(PSEL).(!PENABLE)
5	SETUP	ENABLE	(PSEL).(PENABLE)
6	SETUP	IDLE	(!PSEL)
7	SETUP	SETUP	(PSEL).(!PENABLE)

e Table

SPI FSM

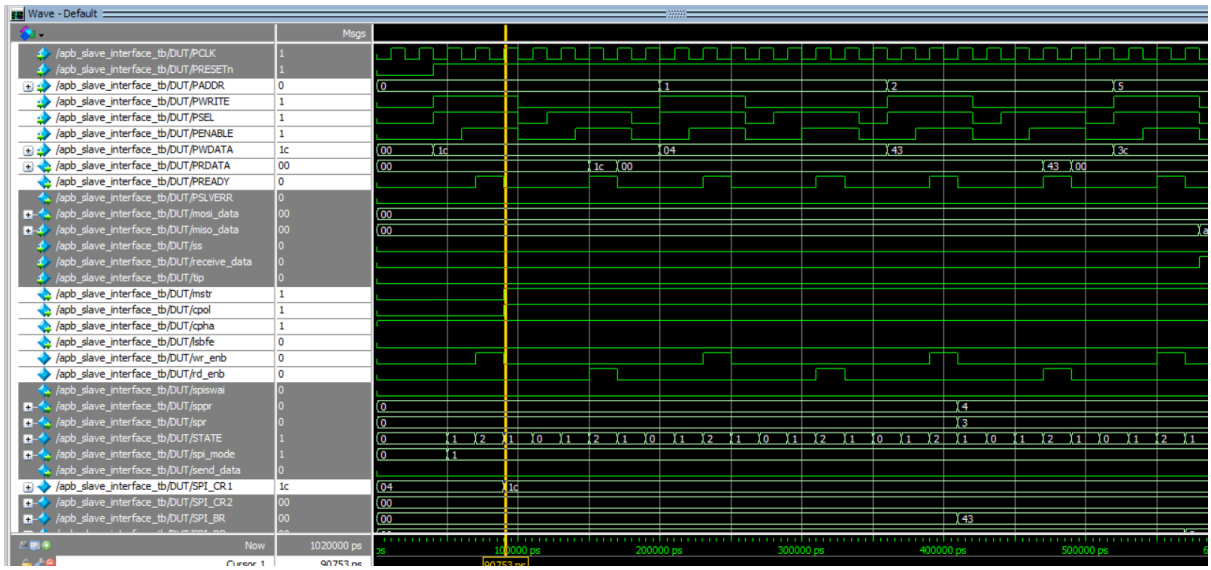


	Source State	Destination State	Condition
1	spi_run	spi_run	(spe)
2	spi_run	spi_wait	(!spe)
3	spi_stop	spi_run	(spe)
4	spi_stop	spi_wait	(!spiswai).(!spe)
5	spi_stop	spi_stop	(spiswai).(!spe)
6	spi_wait	spi_run	(spe)
7	spi_wait	spi_wait	(!spiswai).(!spe)
8	spi_wait	spi_stop	(spiswai).(!spe)

$$BaudRateDivisor = (SPPR + 1) \bullet 2^{(SPR + 1)}$$

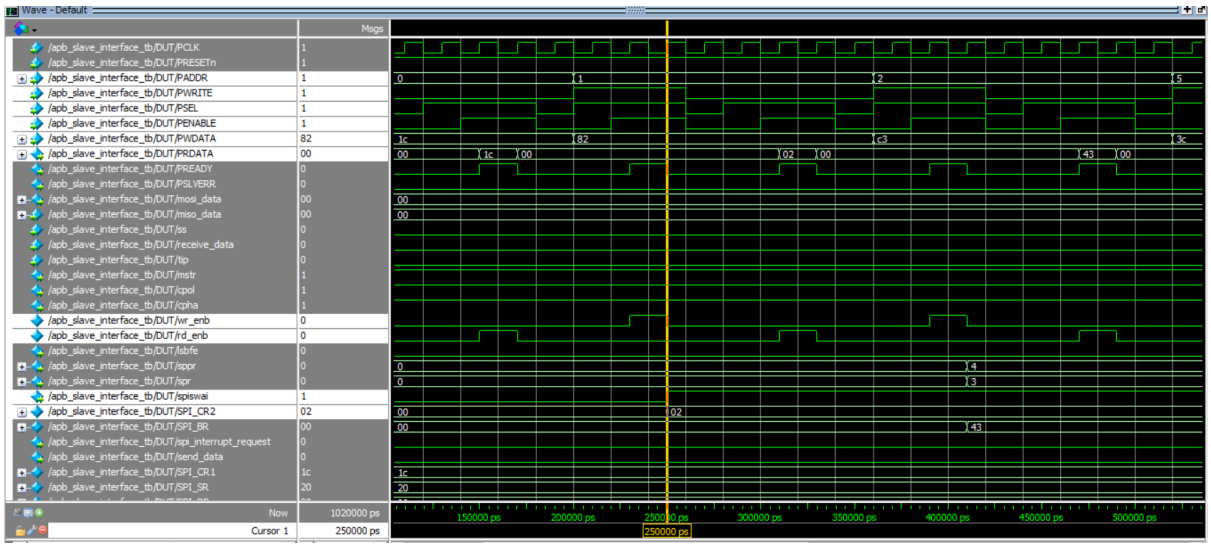
APB SLAVE INTERFACE - RESULTS

Case 1: Read/Write in CR1



CR1 is loaded with 1ch (0001 1100 – which corresponds to mstr=1,cpol=cpha=1,lsbfe=0)

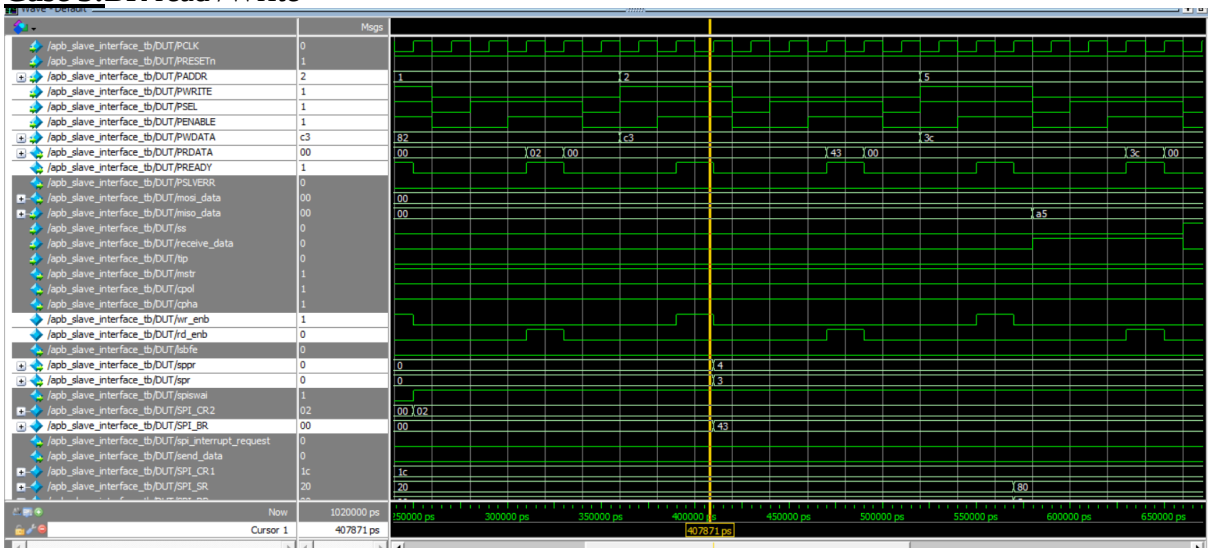
Case 2: Read/Write in CR2



CR2 is loaded with 02 after masking(82h→02h), when read PRDATA contains the same value

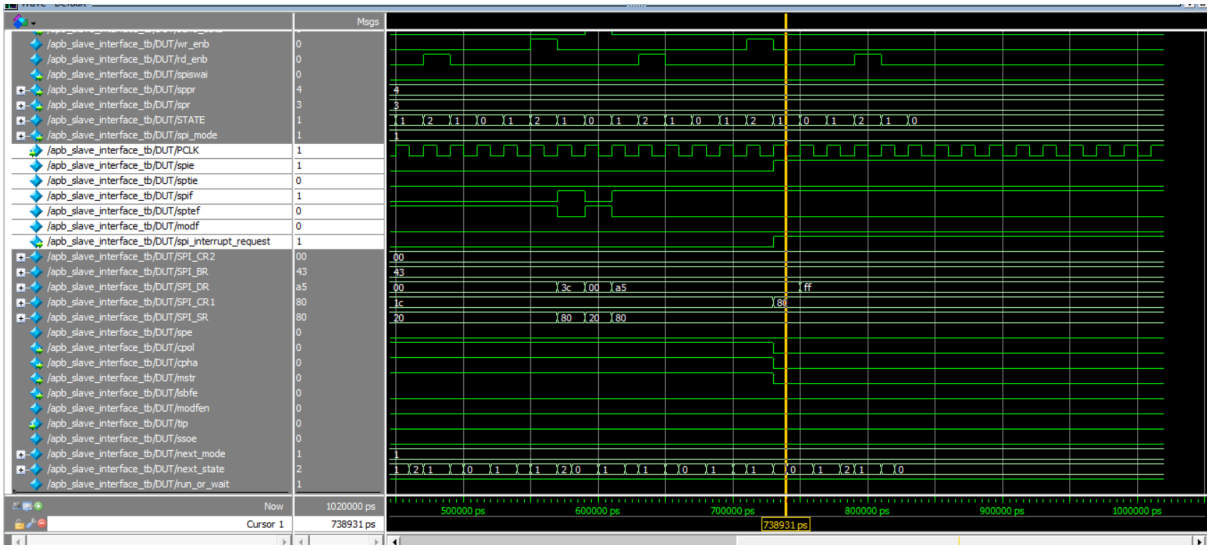
(Note: this spiswai makes spi fsm to go into stop state , so spi wont be in run_or_wait mode, so we cannot perform reception misodata to spi_Dr at this point we need to reset spiswai)

Case 3:BR read /Write

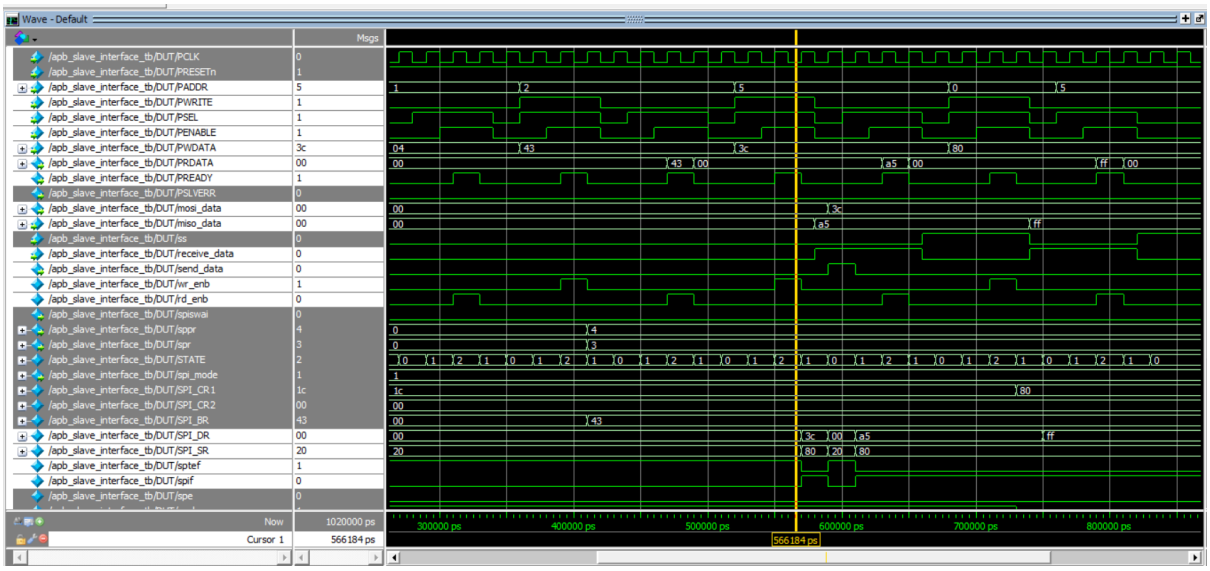


After masking (C3h→43h)SPI_BR becomes 43(hex) when read PRDATA gets 43.

Case 4: DR read/Write

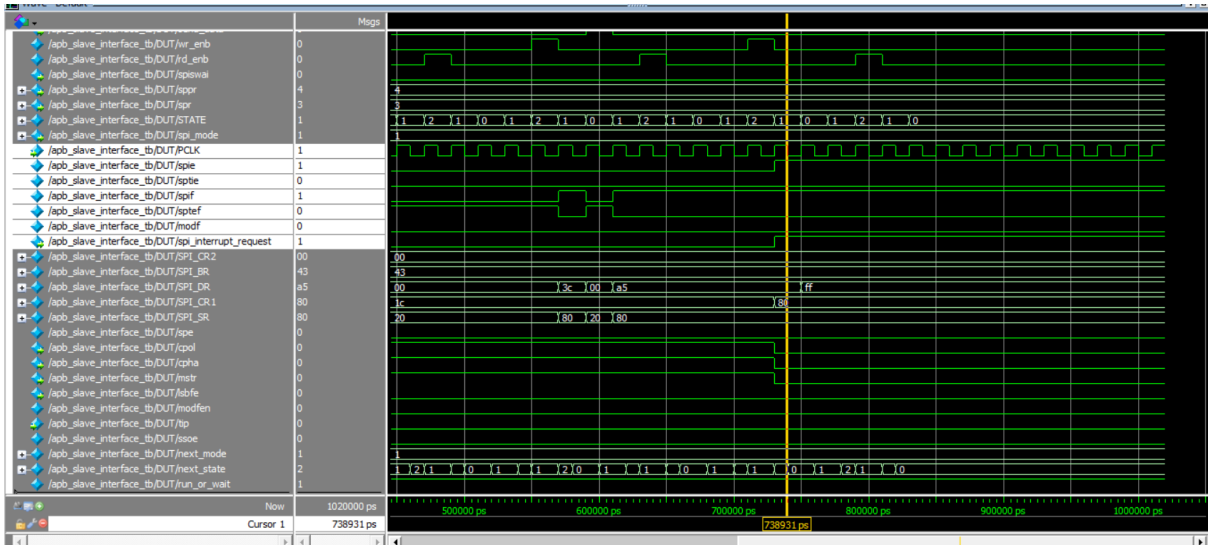


- Note that Status register is updated automatically when dr is loaded with some value (non –zero) so sptef and spif flag changes
- When send_Data is asserted data in SPI_DR=0x3c is loaded into **mosi_data**



- Assume this block has received a **miso_data** = 8’ha5 from the shift register block.
- And 1 clock cycle after receive data=1 SPI_DR is updated with the value received from miso.
- Then now if we read SPI_DR(address = 5) we will get the serially transmitted miso data in PRDATA=8’ha5 .

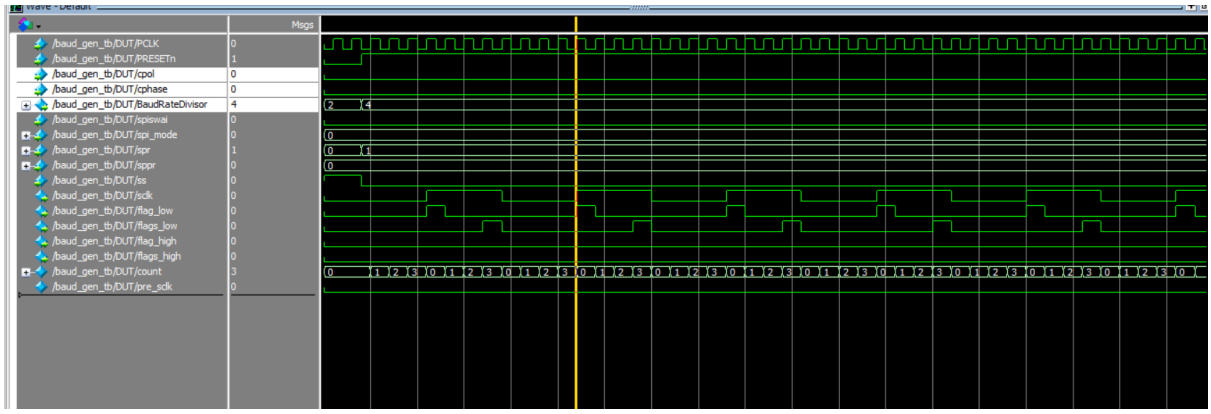
Case 5 : SPI Interrupt request



{spie,sptie}=00 to 10 which corresponds to the following part of code
(i.e., 10: spi_interrupt_request = spif || modf; // Only SPI interrupt enabled)

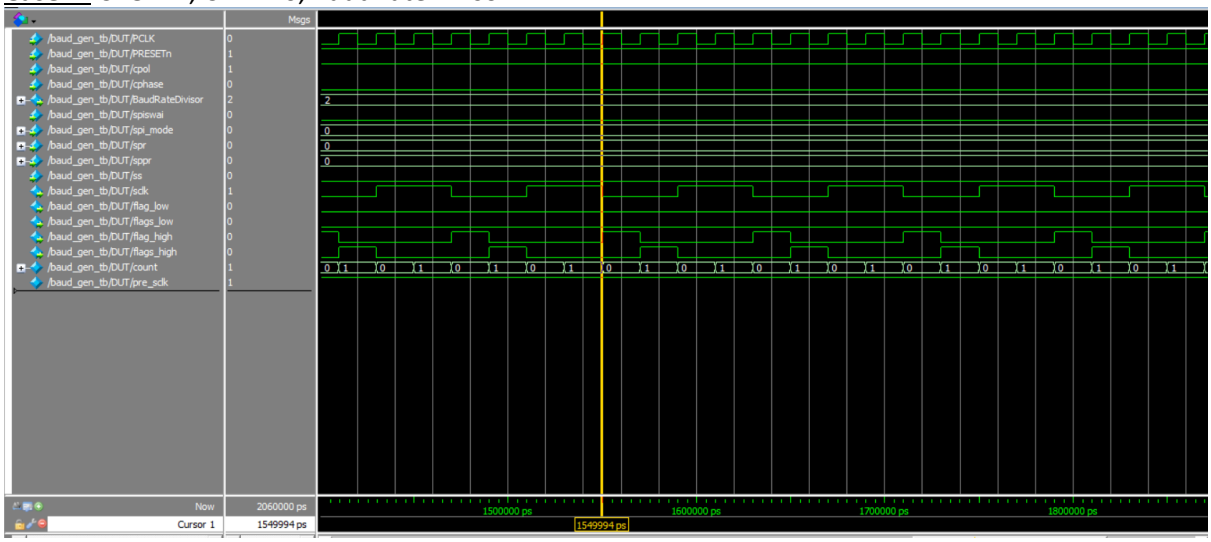
BAUD RATE GENERATOR- RESULTS

Case 1: CPOL=0, CPHA=0, BaudRateDivisor=4



- SCLK is in idle low, and SCLK toggles for every 4 cycles of PCLK.
- CPOL xor CPHASE=0 so flag_low and flags_low will be asserted on rising edge of SCLK and one cycle before falling edge of SCLK respectively.

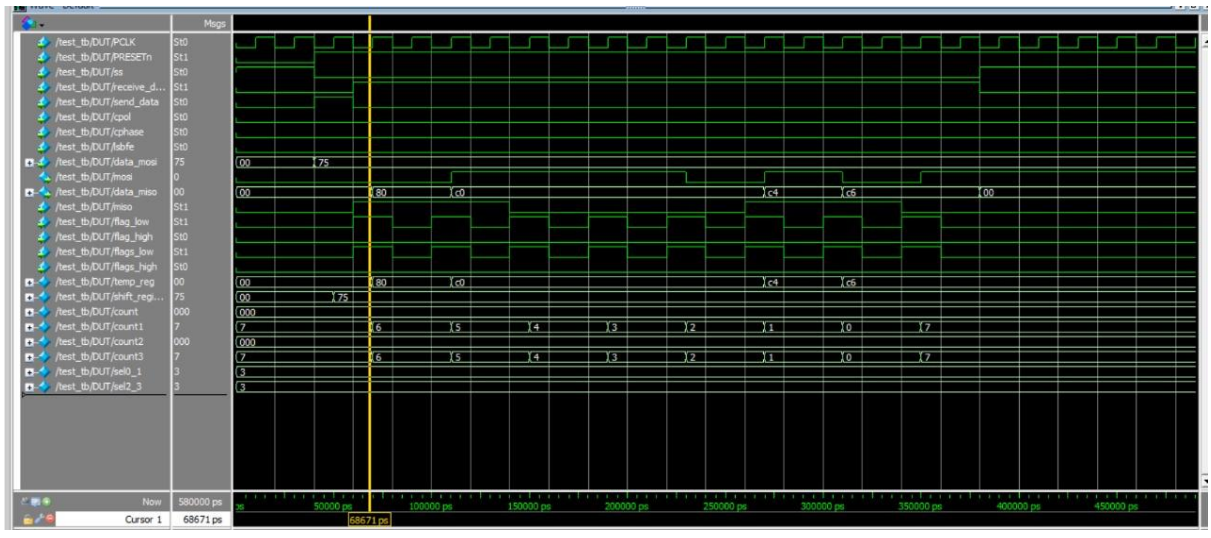
Case 2: CPOL=1, CPHA=0, BaudRateDivisor=2



- SCLK is in idle high, and SCLK toggles for every 2 cycles of PCLK.
- CPOL xor CPHASE=1 so flag_high and flags_high will be asserted on falling edge of SCLK and one cycle before rising edge of SCLK respectively.

SHIFT GENERATOR- RESULTS

Case1: data_mosi=75h, miso serial data=63h,lsbfe=0

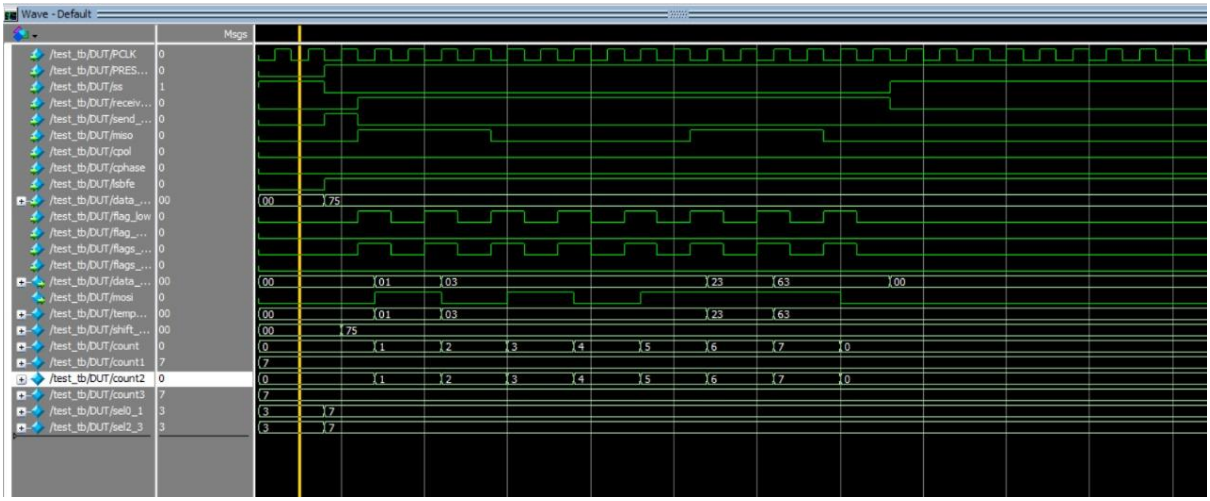


(Note: The flag_low and flags_low do not change at the same time as they are shown here. flags_low and flags_low are inputs to this block and since we don't have their actual hardware signals, we have simulated them for testing and verification purpose. Also receive data is not always 1 as shown-it becomes high only after 8 bit transfer)

- Since lsbfe=0 MSB is shifted out first ,so count1 decrements every time a bit is sent out and count3 decrements every time miso is received in temp_reg.

- You can see the MSB shifting out first in MOSI (0 1 1 1 0 1 0 1 in subsequent clock cycle after flags_low is changed).Also data_miso holds the value in reverse c6h(1100_0110 reverse of 63h 0110_0011)

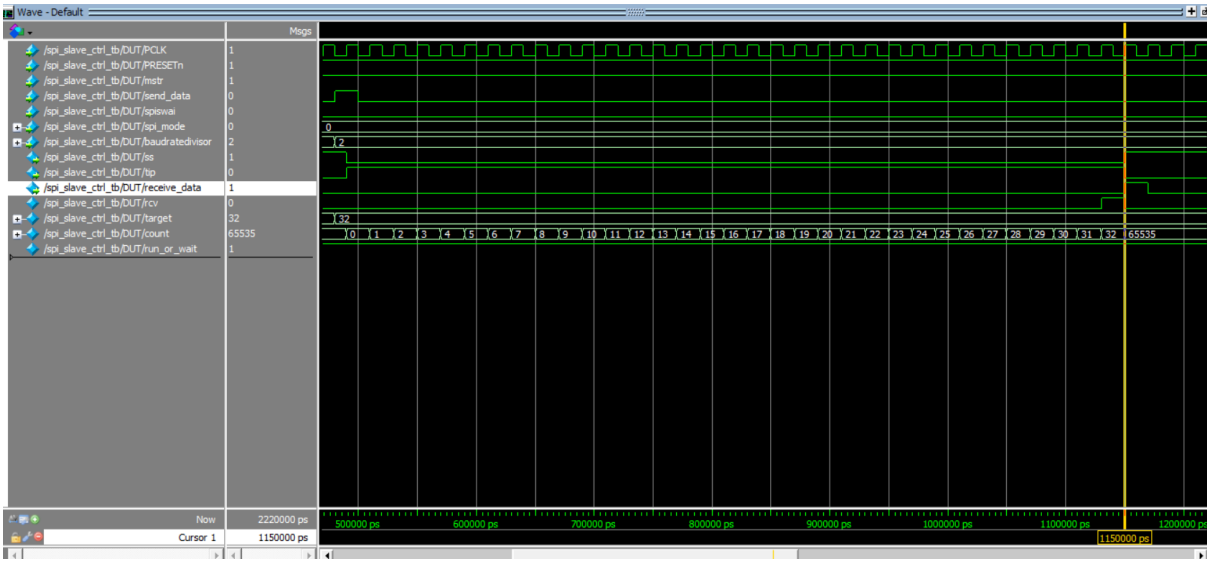
Case2: data_mosi=75h, miso serial data=63h,lsbfe=1



- Since lsbfe=1 LSB is shifted out first ,so count increments every time a bit is sent out and count2 increments every time miso is received in temp_reg.
- You can see the LSB shifting out first in MOSI (1 0 1 0 1 1 1 0) in subsequent clock cycle after flags_low is changed), data_miso holds the value in 63h ,not in reverse like the previous case.

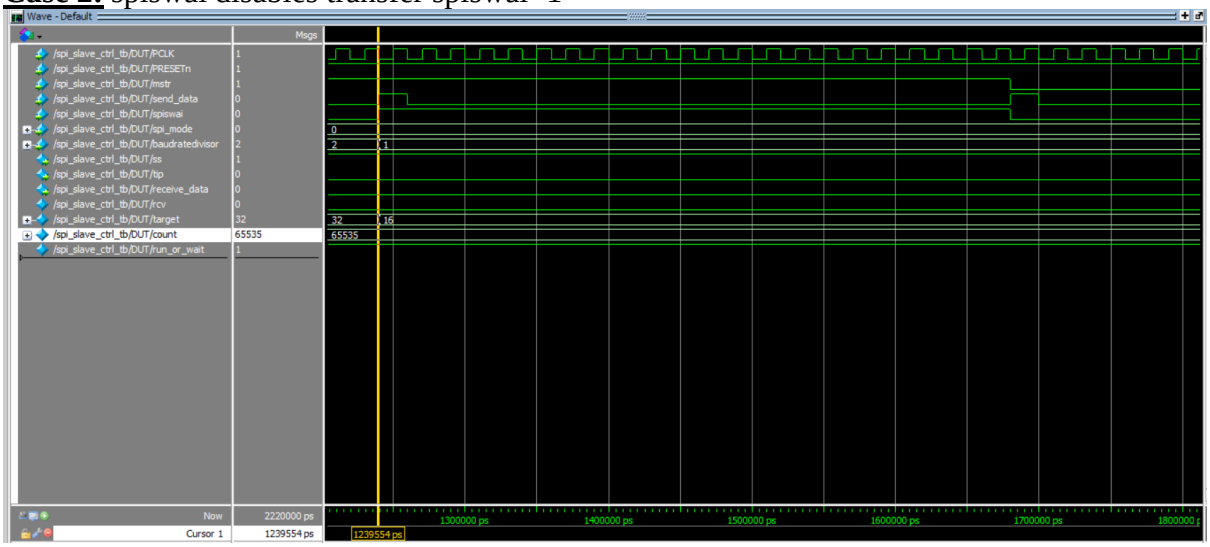
SLAVE CONTROL- RESULTS

Case 1: Divisor = 2, target = 32



After send_data=1, ss is pulled low and transaction starts.Since BaudRatedivisor=2, target will be 16*2=32 ,so after count reaches 32 receive data is asserted to indicate 8 bit transfer has completed .

Case 2: spiswai disables transfer spiswai=1



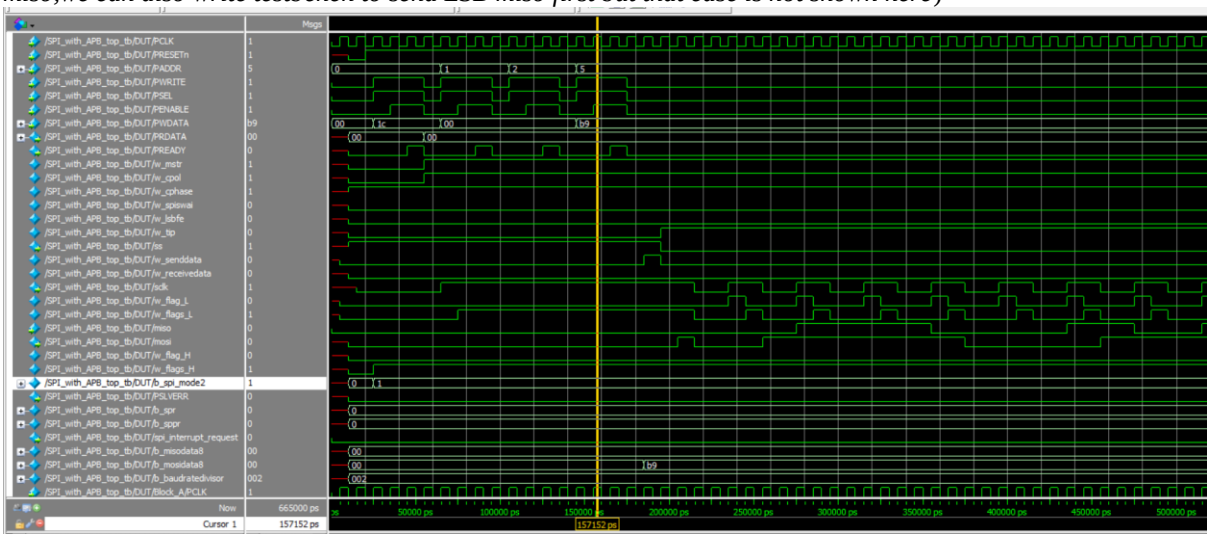
`ss` remains 1, no transaction happens (similar behaviour will happen when mstr=0)

TOP MODULE- Few Cases

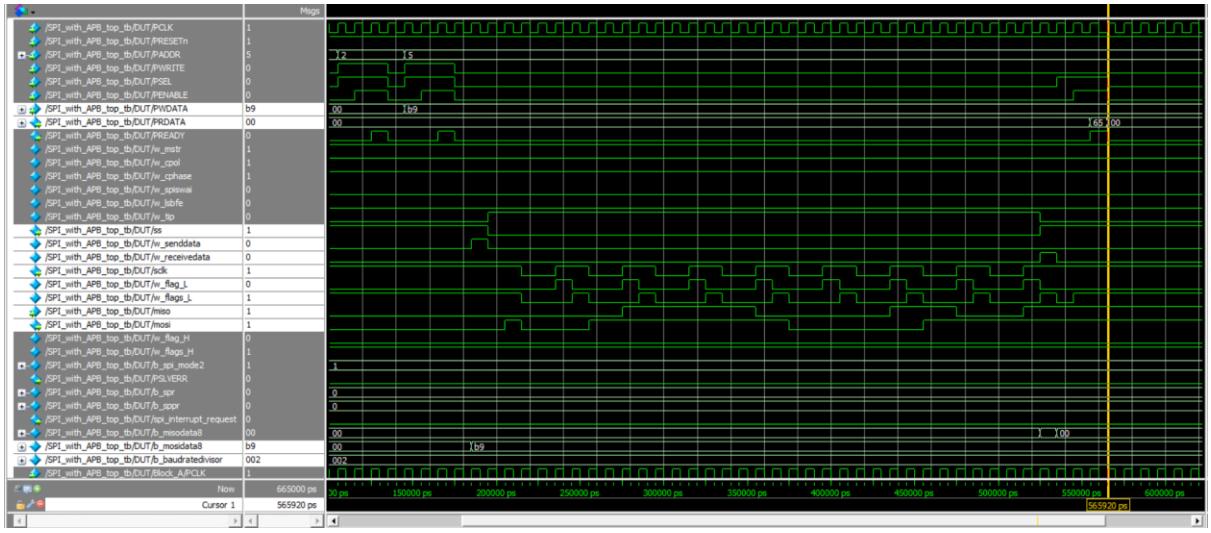
Case1:

CPOL=1,CPHA=1,BaudRateDivisor=2, MISO*=65h(MSB first),MOSI=B9h(lsbfe=0)

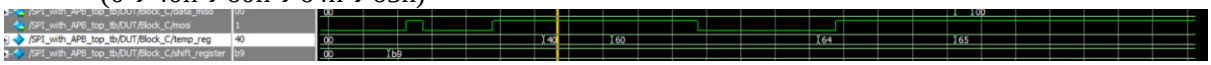
(**Here, MISO=65h means we are simulating SPI slave sending 65h serially MSB first, not the final data in miso,we can also write testbench to send LSB miso first but that case is not shown here*)



- SPI_CR1 Is loaded with 1ch, SPI_BR is loaded with 00h
- SPI_DR is loaded with B9h (8'b1011_1001) to be shifted out via MOSI.

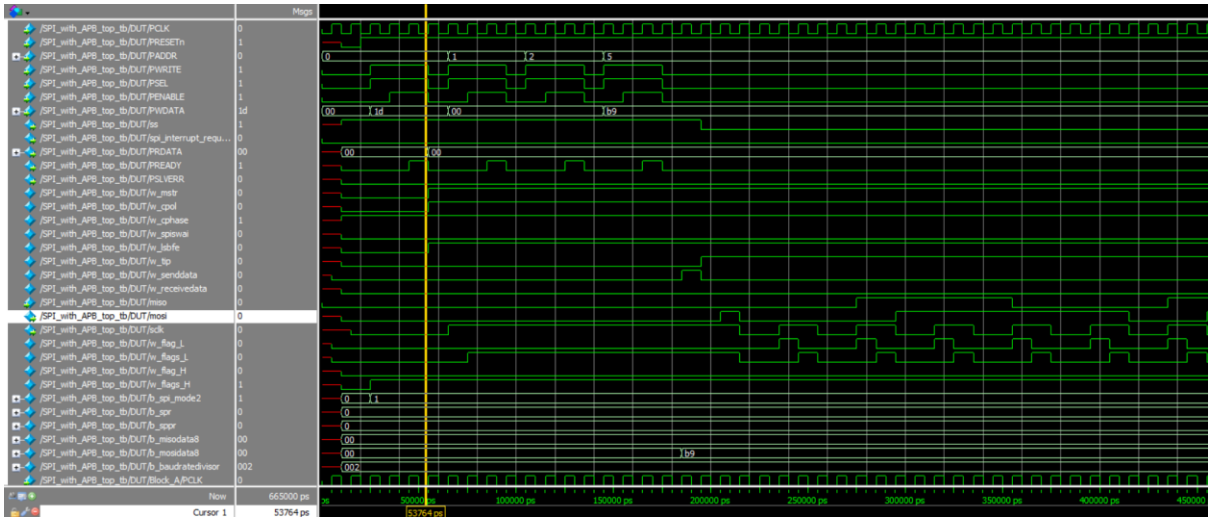


- After SPI_DR is loaded with some value send_data turns high for a clock cycle.
- One clock after send_Data ss goes low which starts sclk generation ,starts shifting out data (mosi)every **falling edge of sclk** and samples data coming from miso at every **rising edge of sclk** .
- After 16 sclk transitions (8 clock cycles),receive data is asserted indicating completion of 1 full 8 bit SPI data transfer.
- After receive_data is asserted and deasserted , APB read is performed on SPI_DR and PRDATA gets the value that miso received serially (65h-8'b0110_0101) .
  (0→40h→60h→64h→65h)

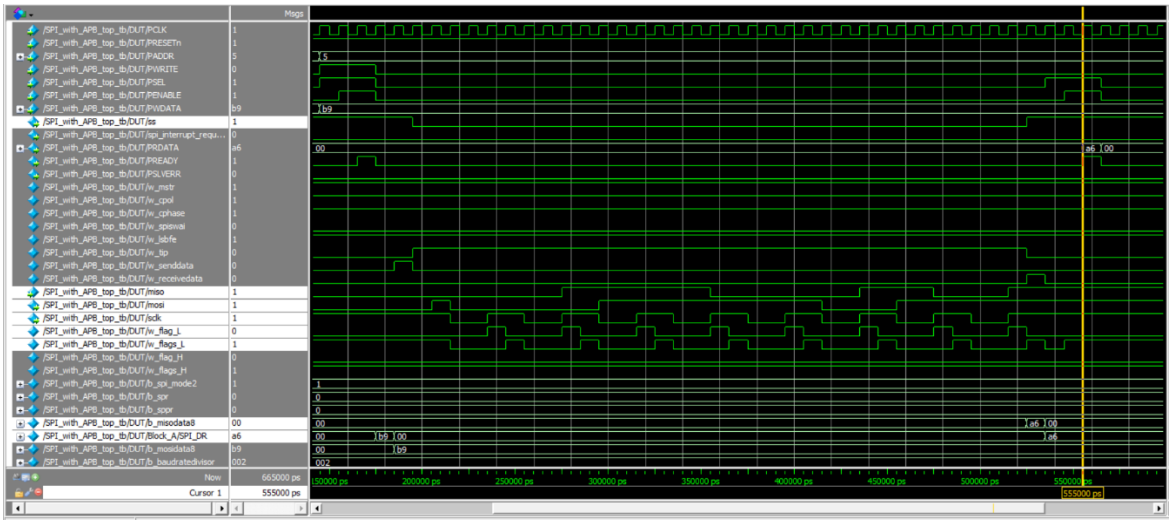


Case2:

CPOL=1,CPHA=1,BaudRateDivisor=2, MISO*=65h(MSB first),MOSI=B9h(lsbfe=1)



- SPI_CR1 Is loaded with 1Dh, SPI_BR is loaded with 00h
- Rest of the steps are same but since lsbfe =1 ,65h(0110_0101) is received as a6h (1010_0110)
  (02h→06h→26h→a6h)


Msgs
/SPI_with_APB_top_tb/DUT/PCLK
/SPI_with_APB_top_tb/DUT/PRESETn
/SPI_with_APB_top_tb/DUT/PADDR
/SPI_with_APB_top_tb/DUT/PWRITE
/SPI_with_APB_top_tb/DUT/PSEL
/SPI_with_APB_top_tb/DUT/PENABLE
/SPI_with_APB_top_tb/DUT/PWDATA
/SPI_with_APB_top_tb/DUT/ss
/SPI_with_APB_top_tb/DUT/spi_interrupt_requ...
/SPI_with_APB_top_tb/DUT/PRDATA
/SPI_with_APB_top_tb/DUT/PREADY
/SPI_with_APB_top_tb/DUT/PSLVERR
/SPI_with_APB_top_tb/DUT/iv_mstr
/SPI_with_APB_top_tb/DUT/iv_cpol
/SPI_with_APB_top_tb/DUT/iv_cphase
/SPI_with_APB_top_tb/DUT/iv_spiswai
/SPI_with_APB_top_tb/DUT/iv_lsbfe
/SPI_with_APB_top_tb/DUT/iv_tip
/SPI_with_APB_top_tb/DUT/iv_senddata
/SPI_with_APB_top_tb/DUT/iv_receivedata
/SPI_with_APB_top_tb/DUT/miso
/SPI_with_APB_top_tb/DUT/mosi
/SPI_with_APB_top_tb/DUT/sclk
/SPI_with_APB_top_tb/DUT/iv_flag_L
/SPI_with_APB_top_tb/DUT/iv_flags_L
/SPI_with_APB_top_tb/DUT/iv_flag_H
/SPI_with_APB_top_tb/DUT/iv_flags_H
/SPI_with_APB_top_tb/DUT/b_spi_mode2
/SPI_with_APB_top_tb/DUT/b_spr
/SPI_with_APB_top_tb/DUT/b_sppr
/SPI_with_APB_top_tb/DUT/b_misodata8
/SPI_with_APB_top_tb/DUT/Block_A/SPI_DR
/SPI_with_APB_top_tb/DUT/b_mosidata8
/SPI_with_APB_top_tb/DUT/b_baudratedivisor
Now
665000 ps
Cursor 1
555000 ps
150000 ps
200000 ps
250000 ps
300000 ps
350000 ps
400000 ps
450000 ps
500000 ps
550000 ps
555000 ps