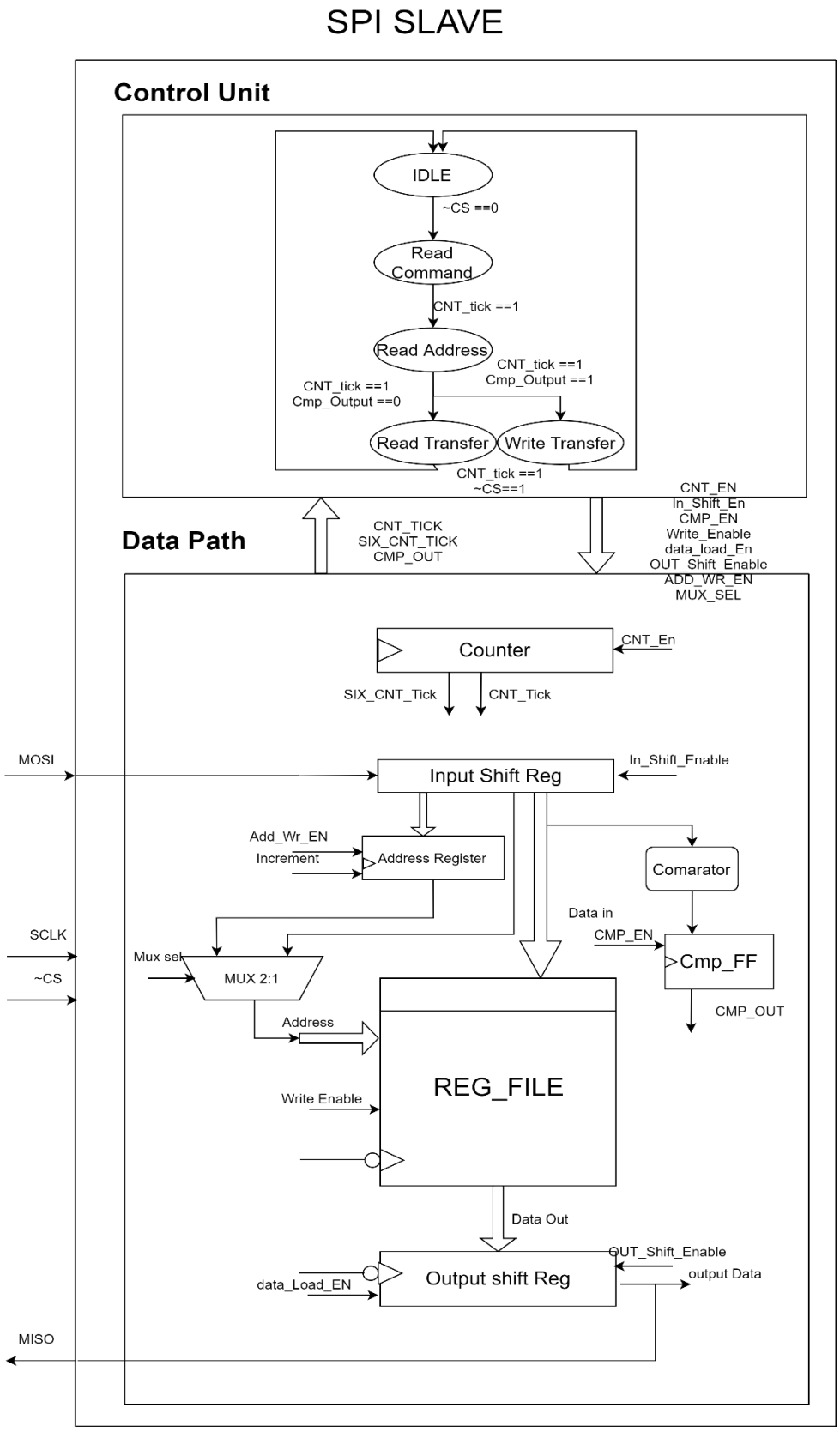
SPI Slave Design Document

Ahmed Emad Ibrahim

Block Diagram:



I/O ports description:

1. **MOSI:** master output slave input (data output from the master)
2. **MISO:** master input slave output (data output from the slave)
3. **SCLK:** Serial Clock (output from master)
4. **~CS:** Chip select (active low)

Functional/implementation description:

States:

1. **IDLE:** CS is high and no transactions request from the master
   -Signals: (CS==1) all enable signals are Zero (default values)
   if (CS ==0)
   {
   -state next = Read Command
   -In_shift_Enable = 1, to sample the first bit in the next rising clk edge
   }

2. **Read Command:** read the 8 bits command and at the end compare the result with Read command and store the result in the flip flop to be later used, enable the counter to count number of the bits and arises counter tick when reaches 8 bits i.e. the counter reaches 7
   -Signals:
   -In_shift_Enable = 1, to sample the input data (MOSI) in the next rising clk edge
   -CNT_ENABLE = 1
   if (CNT_Tick==1) when 8 bits is Received
   {state next = Read Address
   CMP_EN = 1 to allow the FF to store the result
   }

3. **Read Address:** Read the 8 bits Address of Read or Write operation
   If read operation the load enable signal for Output register is activated to load the value to be sent to the receiver and the address is stored in the Address register to be used later if multi read operation

   If write operation the Address Register is used to store the address to which the received data is stored
   -Signals:
   -In_shift_Enable = 1, to sample the input data (MOSI) in the next rising clk edge
   -CNT_ENABLE = 1
   if (CNT_Tick==1 && CMP_Output ==1) **8 bits is received & read operation**
   {
   state next = Read
   Add_Wr_Enable = 1
   Mux_SEL = 0 , i.e. the Received bits is Address
   data_Load_EN = 1 , to allow the output register to capture the data next falling edge
   }
   if (CNT_Tick==1 && CMP_Output ==0) **,8 bits is received &Write operation**
   {
   state next = Write
   Add_Wr_Enable = 1
   }
4. **Read:** Shifts the data until the 8 bits is shifted if CS is high then this is the last transfer if not increment the Address, then start new transmission
   -Signals:
   If (CS ==1) -> **State_Next = IDLE**
   -Out_shift_Enable = 1, to shift the data in the next Falling clk edge
   -CNT_ENABLE = 1
   If (Counter ==6 **) ->** Increment = 1 // Increase Address

if (CNT_Tick==1 && CS =0) when 8 bits is Transmitted and not the last Byte
{state next = Read Address
data_Load_EN = 1
Out_shift_Enable = 0 }

5. **Write:** Read the 8 bits data then store them in the address stored in address register if CS is High then it is last byte if not then increment the Address
   -Signals:
   If (CS ==1) -> **State_Next = IDLE**
   -In_shift_Enable = 1,
   -CNT_ENABLE = 1

   if (CNT_Tick==1) when 8 bits is Transmitted and not the last Byte
   {
   Mux_sel = 1 , to load the address from the register
   Write enable = 1 , to write in the register file
   if (CS==0) -> Increment = 1
   }