```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
--use IEEE.STD_LOGIC_ARITH.ALL;
--use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity floppy_controller is
   Port (
      clk50   : in  std_logic; -- 50MHz from CPLD board
      ph0     : in  std_logic; -- PHI0 (8MHz) from mainboard
      ph2     : in  std_logic; -- PHI2 (1MHz) from mainboard
      nCSCPLD : in  std_logic; -- /CS from STM32
      uSCK    : in  std_logic; -- SCK from STM32
      uMOSI   : in  std_logic; -- MOSI from STM32
      nRST    : in  std_logic; -- /Reset from mainboard
      RnWin   : in  std_logic; -- R/W from mainboard
      uRnW    : in  std_logic; -- R/W from STM32
      ucBSY   : out std_logic; -- /Interrupt to STM32
      stmWD_S  : in std_logic;  -- STM32 write Data Reg. or Status Reg.
      uMISO   : out std_logic; -- MISO to STM32
      stmDRQ  : out std_logic; -- Busy gets read by Unicomp to STM32
      Dio     : inout std_logic_vector(7 downto 0); -- Data from mainboard
      A       : in std_logic_vector(15 downto 0);   -- Address from mainboard
      SPI_A   : out std_logic_vector(2 downto 0);   -- Address to STM32
      As      : out std_logic_vector(19 downto 0)    -- here only for debug
   );
end floppy_controller;

architecture Behavioral of floppy_controller is
   -- Version number: Design_Major_Minor
   constant VERSION_NUM       : std_logic_vector(7 downto 0) := x"71";

   signal spi_new_data     : std_logic;
   signal spi_new_status     : std_logic;

   signal spi_data         : std_logic_vector(7 downto 0);
   signal ncs_fdc          : std_logic;
   signal ncs_drv          : std_logic;
   signal ncs              : std_logic;
   signal nBSY_Read        : std_logic;
   signal ucDAT_Read       : std_logic;
   signal spiDAT_Write     : std_logic;
   signal sINT             : std_logic;
   signal bIRQ             : std_logic;
   signal fDRQ             : std_logic;
   signal sStepDir         : std_logic; -- 1 is step in, 0 is step out
   signal uc_data_update   : unsigned(3 downto 0);
   signal sr_data_update   : unsigned(3 downto 0);

   signal D_drv            : std_logic_vector(7 downto 0); -- 8014 Write
   --signal D_irq            : std_logic_vector(7 downto 0); -- 8014 Read
   signal D_fdc_cmd        : std_logic_vector(7 downto 0); -- 8018 Write
   signal D_fdc_sta        : std_logic_vector(7 downto 0); -- 8018 Read
   signal D_fdc_trk        : std_logic_vector(7 downto 0); -- 8019
   signal D_fdc_sec        : std_logic_vector(7 downto 0); -- 801A
   signal D_fdc_dat        : std_logic_vector(7 downto 0); -- 801B

   signal D_address        : std_logic_vector(7 downto 0); --
   signal temp             : std_logic_vector(7 downto 0); --
   signal D_spi_dat           : std_logic_vector(7 downto 0);
   alias bBUSY               is D_fdc_sta(0);
   alias bDRQ                is D_fdc_sta(1);
   alias bTRK0               is D_fdc_sta(2);
   alias bCRCERR             is D_fdc_sta(3);
   alias bRNFERR             is D_fdc_sta(4);
   alias bWFLT_HLD           is D_fdc_sta(5);
   alias bWPRT               is D_fdc_sta(6);
   alias bNRDY               is D_fdc_sta(7);

begin

   spi_slave_out : entity work.spi_slave_out48 -- shift register
   port map (
      sclk     => uSCK and uRnW, -- Read only
      ss_n     => nCSCPLD, -- and not (sr_Data_new or uc_Data_new),
      mosi     => '0', --uMOSI,
      miso     => uMISO,
      data      => D_fdc_cmd&D_fdc_trk&D_fdc_sec&D_fdc_dat&D_drv&D_fdc_sta
   );

   spi_slave_in : entity work.spi_slave_in8 -- shift register
   port map (
      sclk     => uSCK and not uRnW, -- Write only
      ss_n     => nCSCPLD, -- and not (sr_Data_new or uc_Data_new),
      mosi     => uMOSI,
      --miso     => uMISO,
      data      => spi_data
   );

```

```vhdl
-- Chip Select
   ncs_drv <= '0' when A(15 downto 2) = "10000000000101" else '1';-- 8014 - 8017 cs for drive reg.
   ncs_fdc <= '0' when A(15 downto 2) = "10000000000110" else '1';-- 8018 - 801B cs for floppy controller
      ncs <= ncs_drv and ncs_fdc; -- combined cs


-- Bus Isolation
   Dio <= D_fdc_sta  when RnWin = '1' and ncs_fdc = '0' and A(1 downto 0) = "00" and nRST = '1' else (others => 'Z');
   Dio <= D_fdc_trk  when RnWin = '1' and ncs_fdc = '0' and A(1 downto 0) = "01" and nRST = '1' else (others => 'Z');
   Dio <= D_fdc_sec  when RnWin = '1' and ncs_fdc = '0' and A(1 downto 0) = "10" and nRST = '1' else (others => 'Z');
   Dio <= D_spi_dat  when RnWin = '1' and ncs_fdc = '0' and A(1 downto 0) = "11" and nRST = '1' else (others => 'Z');
   Dio <= bIRQ&bDRQ&"000000" when RnWin = '1' and ncs_drv = '0' and nRST = '1' else (others => 'Z');

      -- WRITE

   --ucBSY <= ncs_fdc or RnWin; -- Interrupt only when writing
   ucBSY <= not bBUSY; -- Interrupt only when writing from Unicomp
   stmDRQ <= fDRQ;
   --stmDRQ <= bDRQ;

process (clk50)
   begin
      if rising_edge(clk50) then
         if nRST = '0' then
            --uc_Data_new <= '0';
            --uc_Data_lock <= '0';
            D_fdc_sta <= "00000000";
            D_fdc_cmd <= "00000000";
            D_fdc_trk <= "00000000";
            D_fdc_sec <= "00000001";
            D_fdc_dat <= "00000000";
            D_spi_dat <= "00000000";
            --D_irq <= "00000000";
            spi_new_data <= '0';
            spi_new_status <= '0';
            fDRQ <= '0';
            Dio <= (others => 'Z');
            ucDAT_Read <= '0';
            nBSY_Read <= '0';
         else
            if nCSCPLD = '0' and uRnW = '0' then -- new data will arrive over SPI
               if stmWD_S = '0' then -- we want to write Data
                  spi_new_data <= '1';
                  spi_new_status <= '0';
                  --D_fdc_dat <= spi_data;  -- Data Reg.
                  fDRQ <= '1';  -- block new Data sending
                  bDRQ <= '1';  -- signal New Data Flag
               else                  -- we want to write Status Reg.
                  --D_fdc_sta <= spi_data; -- Status Reg.
                  spi_new_status <= '1';
                  spi_new_data <= '0';
               end if;
            end if;
            if spi_new_data = '1' and nCSCPLD = '1' then
               --bDRQ <= '1';  -- signal New Data Flag
               D_spi_dat <= spi_data;  -- Data Reg.
               spi_new_data <= '0';
            end if;
            if spi_new_status = '1' and nCSCPLD = '1' then
               D_fdc_sta <= spi_data; -- Status Reg.
               spi_new_status <= '0';
            end if;
-------------------------------------------------------------------------------------
--          Handles FDC1771 Reads
-------------------------------------------------------------------------------------
            if RnWin = '1' and ncs_fdc = '0' then
               case A(1 downto 0) is
               when "00" =>
                  --ucDAT_Read <= '0';
                  nBSY_Read <= '1';
               when "01" =>
                  --ucDAT_Read <= '0';
               when "10" =>
                  --ucDAT_Read <= '0';
               when "11" =>
                  ucDAT_Read <= '1';
                  fDRQ <= '0';  -- allow new Data to be sent
                  --bDRQ <= '0';  -- clear New Data Flag
                  --bDRQI <= '0'; -- DRQ Output low (connected to Drive Register)
               end case;
            else
               Dio <= (others => 'Z');
               nBSY_Read <= '0';
            end if;

            if ucDAT_Read = '1' and ncs_fdc = '1' then -- wait until chip is not selected anymore

               bDRQ <= '0';  -- clear New Data Flag
               --if fDRQ = '0' and bDRQ = '0' then
               ucDAT_Read <= '0';
               --end if;
               --bDRQI <= '0'; -- DRQ Output low (connected to Drive Register)
            end if;

            --if uc_Data_new = '1' and nCSCPLD = '0' and uRnW = '1' then --and uc_Data_lock = '0' then -- data is latched into shiftreg.
            -- uc_Data_new <= '0';
            -- uc_Data_lock <= '1'; -- needed if new data comes in before spi is finished
               -- SPI is now 25MHz and needs 6us for 6 Bytes inkl. CS
               -- Response from uc_Data_new high to CS low is 350ns!
               -- Transmission starts BEFORE ncs_fdc is high again.
            --end if;

            --if uc_Data_lock = '1' and nCSCPLD = '1' then
            -- uc_Data_lock <= '0';
            --end if;
-----------------------------------------------------------------------------------
--          Handles FDC1771 and Drive Select Writes
-----------------------------------------------------------------------------------
            if RnWin = '0' and ncs_drv = '0' and ph2 = '1' then -- Write Drive Reg
               D_drv <= Dio;
               --uc_data_update <= "0001";
               --uc_Data_new <= '1';
            elsif RnWin = '0' and ncs_fdc = '0' and ph2 = '1' then -- Write FDC Reg
               --uc_data_update <= "0001";
               --uc_Data_new <= '1';
               case A(1 downto 0) is
                  when "00" =>
                  D_fdc_cmd <= Dio;

                  if Dio(7 downto 4) = "0000" then    -- '0x'          ### Restore ###
                     D_fdc_trk <= (others => '0');  -- set to track 0
                     bIRQ <= '1';                   -- set Interrupt (probably wait some time)
                     bTRK0 <= '1';                  -- set to track 0 flag
                     --bBUSY <= '0';                  -- clear BUSY Flag
                  elsif Dio(7 downto 4) = "0001" then -- '1x'           ### Seek ###
                     D_fdc_trk <= D_fdc_dat;
                     bIRQ <= '1';                   -- set Interrupt (probably wait some time)
                  elsif Dio(7 downto 5) = "001" then  --  '2x, 3x'      ### Step ###
                     if Dio(4) = '1' then           -- update flag is set?
                        if sStepDir = '1' then
                           D_fdc_trk <= std_logic_vector(to_unsigned(to_integer(unsigned(D_fdc_trk)) + 1, 8)); -- update track register
                           bTRK0 <= '0';            -- reset track 0 flag
                        else
                           D_fdc_trk <= std_logic_vector(to_unsigned(to_integer(unsigned(D_fdc_trk)) - 1, 8)); -- update track register
                        end if;
                     end if;
                     bIRQ <= '1';                   -- set Interrupt (probably wait some time)
                  elsif Dio(7 downto 5) = "010" then  --  '4x, 5x'      ### Step in ###
                     if Dio(4) = '1' then           -- update flag is set?
                        D_fdc_trk <= std_logic_vector(to_unsigned(to_integer(unsigned(D_fdc_trk)) + 1, 8)); -- update track register
                        bTRK0 <= '0';               -- reset track 0 flag
                     end if;
                     sStepDir <= '1';
                     bIRQ <= '1';                   -- set Interrupt (probably wait some time)
                  elsif Dio(7 downto 5) = "011" then  --  '6x, 7x'      ### Step out ###
                     if Dio(4) = '1' then           -- update flag is set?
                        D_fdc_trk <= std_logic_vector(to_unsigned(to_integer(unsigned(D_fdc_trk)) - 1, 8)); -- update track register
                     end if;
                     sStepDir <= '0';
                     bIRQ <= '1';                   -- set Interrupt (probably wait some time)
                  elsif Dio(7 downto 5) = "100" then  --  '8x, 9x'      ### Read ###
                     bWFLT_HLD <= '1';              -- set Head Load Flag
                     bBUSY <= '1';                  -- set BUSY Flag
                  elsif Dio(7 downto 5) = "101" then  --   'Ax, Bx'     ### Write ###
                     bWFLT_HLD <= '1';              -- set Head Load Flag
                     bBUSY <= '1';                  -- set BUSY Flag
                  elsif Dio(7 downto 0) = "11000100" then -- 'C4' Read Address
                  elsif Dio(7 downto 1) = "1110010" then  -- 'E4' Read Track
                  elsif Dio(7 downto 0) = "11110100" then -- 'F4' Write Track
                  elsif Dio(7 downto 4) = "1101" then -- 'Dx'           ### Force Interrupt (Reset busy) ###
                     bBUSY <= '0';                  -- reset BUSY Flag
                     bIRQ <= '0';
                  end if;
                  when "01" =>
                  D_fdc_trk <= Dio;
                  when "10" =>
                  D_fdc_sec <= Dio;
                  when "11" =>
                  D_fdc_dat <= Dio;
               end case;
            end if;
         end if;
      end if;
end process;
As(19) <= ncs;         -- Pin 2
--As(19) <= spi_new_data;         -- Pin 2
As(17) <= nBSY_Read; -- Pin 4
As(15) <= ucDAT_Read;    -- Pin 6

end Behavioral;
```