# HA

## Design Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity HA is
    Port ( a : in  STD_LOGIC; -- Input A
           b : in  STD_LOGIC; -- Input B
           s : out STD_LOGIC; -- Sum output
           c : out STD_LOGIC); -- Carry output
end HA;

architecture Behavioral of HA is
begin

    s <= a xor b;

    c <= a and b;

end Behavioral;
```

| Port Name | FPGA Pin |
|---|---|
| a | R2 |
| b | T1 |
| s | P1 |
| c | L1 |

# FA

## Design Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FA is
    Port ( a  : in  STD_LOGIC;
           b  : in  STD_LOGIC;
           ci : in  STD_LOGIC;
           s  : out STD_LOGIC;
           co : out STD_LOGIC);
end FA;

architecture Behavioral of FA is
begin
```

```
    s <= a xor b xor ci;

    co <= (a and b) or (ci and (a xor b));

end Behavioral;
```

| Port Name | FPGA Pin |
|---|---|
| a | R2 |
| b | T1 |
| ci | U1 |
| s | P1 |
| co | L1 |

# RCA

## Design Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity FA is
    Port ( a  : in  STD_LOGIC;
           b  : in  STD_LOGIC;
           ci : in  STD_LOGIC;
           s  : out STD_LOGIC;
           co : out STD_LOGIC);
end FA;

architecture Behavioral of FA is
begin
    s <= a xor b xor ci;
    co <= (a and b) or (ci and (a xor b));
end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RCA is
    Port ( a  : in  STD_LOGIC_VECTOR (3 downto 0);
           b  : in  STD_LOGIC_VECTOR (3 downto 0);
           ci : in  STD_LOGIC;
           s  : out STD_LOGIC_VECTOR (3 downto 0);
           co : out STD_LOGIC);
end RCA;

architecture Structural of RCA is
```

```
    component FA is
        Port ( a  : in  STD_LOGIC;
               b  : in  STD_LOGIC;
               ci : in  STD_LOGIC;
               s  : out STD_LOGIC;
               co : out STD_LOGIC);
    end component;

    signal c1, c2, c3 : STD_LOGIC;

begin

    FA0: FA port map(
        a  => a(0),
        b  => b(0),
        ci => ci,
        s  => s(0),
        co => c1
    );

    FA1: FA port map(
        a  => a(1),
        b  => b(1),
        ci => c1,
        s  => s(1),
        co => c2
    );

    FA2: FA port map(
        a  => a(2),
        b  => b(2),
        ci => c2,
        s  => s(2),
        co => c3
    );

    FA3: FA port map(
        a  => a(3),
        b  => b(3),
        ci => c3,
        s  => s(3),
        co => co
    );

end Structural;
```

| | |
|---|---|
| a3 | R2 |
| a2 | T1 |
| a1 | U1 |
| a0 | W2 |

| | |
|---|---|
| b3 | R3 |
| b2 | T2 |
| b1 | T3 |
| b0 | V2 |
| ci | V17 |
| s3 | L1 |
| s2 | P1 |
| s1 | N3 |
| s0 | P3 |
| co | U16 |

# ALU

## Design Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;

entity ALU is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end ALU;

architecture Behavioral of ALU is

begin

with S select
Y <= A + B when "000",
A - B when "001",
A + 1 when "010",
B + 1 when "011",
A and B when "100",
A or B when "101",
A nand B when "110",
A nor B when "111",
A xor B when others;

end Behavioral;
```

| Port Name | FPGA Pin |
|---|---|
| a3 | R2 |
| a2 | T1 |
| a1 | U1 |
| a0 | W2 |
| b3 | R3 |
| b2 | T2 |
| b1 | T3 |
| b0 | V2 |
| s2 | W16 |
| s1 | V16 |
| s0 | V17 |
| y3 | L1 |
| y2 | P1 |
| y1 | N3 |
| y0 | P3 |

# Mod 16 Counter

## Design Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mod16_counter is
    Port (
        clk   : in  STD_LOGIC;
        dir   : in  STD_LOGIC;
        reset : in  STD_LOGIC;
        count : out STD_LOGIC_VECTOR(3 downto 0)
    );
end mod16_counter;

architecture Behavioral of mod16_counter is
    signal counter : STD_LOGIC_VECTOR(3 downto 0) := "0000";
begin

    process(clk, reset)
    begin
```

```
        if reset = '1' then
            counter <= "0000";
        elsif rising_edge(clk) then
            if dir = '0' then
                -- Count up
                if counter = "1111" then
                    counter <= "0000";
                else
                    counter <= counter + 1;
                end if;
            else
                -- Count down
                if counter = "0000" then
                    counter <= "1111";
                else
                    counter <= counter - 1;
                end if;
            end if;
        end if;
    end process;

    count <= counter;

end Behavioral;
```

## ADD TO CONSTRAINT FILE

```
set_property PACKAGE_PIN E3 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk_IBUF]
```

| Port Name | FPGA Pin |
|---|---|
| clk | V17 |
| dir | R2 |
| reset | T1 |
| c3 | L1 |
| c2 | P1 |
| c1 | N3 |
| c0 | P3 |

# USR

## Design Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity usr1 is
  Port (
      clk : in  STD_LOGIC;
      clk_div : inout std_logic;
      rst : in  STD_LOGIC;
      load: in  STD_LOGIC;
      mode : in  STD_LOGIC_VECTOR(1 downto 0);  -- mode selection input
      si : in  STD_LOGIC;  -- serial input
      pi : in  STD_LOGIC_VECTOR(3 downto 0);  -- parallel input
      so : out  STD_LOGIC;  -- serial output
      po : out  STD_LOGIC_VECTOR(3 downto 0)  -- parallel output
  );
end usr1;

architecture Behavioral of usr1 is
   signal shift_reg : STD_LOGIC_VECTOR(3 downto 0);
   signal counter: std_logic_vector(26 downto 0):=( others=>'0');
begin
   process(clk_div, rst,load)
   begin
      if rst = '1' then
         shift_reg <= (others => '0');
      elsif clk_div'event and clk_div= '1' then
         case mode is
            when "00" =>  -- SISO mode
              shift_reg( 3 downto 1) <= shift_reg(2 downto 0);
               shift_reg(0)<=si;
               so <= shift_reg(3);
            when "01" =>  -- SIPO mode
            shift_reg( 3 downto 1) <= shift_reg(2 downto 0);
               shift_reg(0)<=si;
                po <= shift_reg;
           when "10" =>  -- PISO mode
                 if(load='0') then
                         shift_reg <= pi;
                  else
           shift_reg( 3 downto 1) <= shift_reg(2 downto 0);
                  end if;
                  so <= shift_reg(3);
            when "11" =>  -- PIPO mode
               po<= pi;
```

```
            when others =>
                null;
        end case;
    end if;
  end process;

  -- so <= shift_reg(3);
  -- po <= shift_reg;

  process(clk)
   begin
    if clk'event and clk = '1' then
        if rst = '1' then
            counter <= (others => '0');
        else
            counter <= counter + '1';
        end if;
    end if;
  end process;

clk_div<= counter(26);

end Behavioral;
```

| Port Name | FPGA Pin |
|---|---|
| clk | W5 |
| rst | U1 |
| pi3 | W17 |
| pi2 | W16 |
| pi1 | V16 |
| pi0 | V17 |
| si | W15 |
| mode1 | R2 |
| mode0 | T1 |
| load | W2 |
| po3 | V19 |
| po2 | U19 |
| po1 | E19 |
| po0 | U16 |
| so | W18 |
| clk_div | L1 |

# FIFO

## Design Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity fifo_correct is
GENERIC
(
ADDRESS_WIDTH : integer:=2;---8 bit
DATA_WIDTH : integer:=4 ---32 bit
);

port ( clk : in std_logic;
clk_div : inout std_logic;
reset : in std_logic;
enr : in std_logic; --enable read,should be '0' when not in use.
enw : in std_logic; --enable write,should be '0' when not in use.
dataout : out std_logic_vector(DATA_WIDTH-1 downto 0); --output data
datain : in std_logic_vector (DATA_WIDTH-1 downto 0); --input data
empty : out std_logic; --set as '1' when the queue is empty
err : out std_logic;
full : out std_logic --set as '1' when the queue is full
);
end fifo_correct;

architecture Behavioral of fifo_correct is

type memory_type is array (0 to ((2**ADDRESS_WIDTH)-1)) of std_logic_vector(DATA_WIDTH-1 downto
0);


-----distributed-------
signal memory : memory_type ;-- :=(others => (others => '0')); --memory for queue.-----
signal readptr,writeptr : std_logic_vector(ADDRESS_WIDTH-1 downto 0); --read and write pointers.
signal full0 : std_logic;
signal empty0 : std_logic;
signal counter: std_logic_vector(28 downto 0):=( others=>'0');
begin
full <= full0;
empty <= empty0;

fifo0: process(clk_div,reset,datain,enw,enr)
begin
if reset='1' then

readptr <= (others => '0');
writeptr <= (others => '0');
empty0 <='1';
```

```
full0<='0';
err<='0';

elsif  clk_div'event and clk_div = '1' then
if enw='1' and full0='0' then
memory (conv_integer(writeptr)) <= datain ;
writeptr <= writeptr + '1' ;
if (writeptr + '1' = readptr) then
full0<='1';
empty0<= '0';
else
full0<='0';
empty0<= '1';
end if ;
end if ;




if enr='1' and empty0='0' then

dataout <= memory (conv_integer(readptr));
readptr <= readptr + '1' ;
if (readptr + '1'  = writeptr ) then
empty0<='1';
full0<='0';
else
empty0<='0';
full0<='1';
end if ;
end if ;

if (empty0='1' and enr='1') or (full0='1' and enw='1') then
err<='1';
else
err<= '0';
end if ;
end if;

end process;

process(clk)
begin
if clk'event and clk= '1' then
counter<= counter + '1';
end if;
end process;

clk_div<= counter(26);

end Behavioral;
```

| Port Name | FPGA Pin |
|---|---|
| clk | W5 |
| reset | U1 |
| datain3 | W17 |
| datain2 | W16 |
| datain1 | V16 |
| datain0 | V17 |
| enw (enable write) | R2 |
| enr ( enable read) | T1 |
| dataout3 | V19 |
| dataout2 | U19 |
| dataout1 | E19 |
| dataout0 | U16 |
| empty | L1 |
| full | P1 |
| err | N3 |
| clk_div | P3 |

# Keypad

## Design Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity keypad_7seg is
    Port ( clk : in STD_LOGIC;                      -- 100 MHz clock
           row : in STD_LOGIC_VECTOR (3 downto 0); -- Rows from keypad (inputs)
           col : out STD_LOGIC_VECTOR (3 downto 0);  -- Columns to keypad (outputs)
           seg : out STD_LOGIC_VECTOR (6 downto 0); -- Seven-segment cathodes
           an : out STD_LOGIC_VECTOR (3 downto 0)   -- Seven-segment anodes
         );
end keypad_7seg;

architecture Behavioral of keypad_7seg is
    signal col_sel : std_logic_vector(1 downto 0) := "00";
    signal display_digit : std_logic_vector(3 downto 0) := "1111";  -- Latched display value
    signal counter: unsigned(28 downto 0) := (others => '0');
```

```
    signal clk_div : std_logic := '0';
begin
    an <= "1110";  -- Enable rightmost digit (AN[0] = 0)

    -- Clock divider process
    process(clk)
    begin
        if clk'event and clk = '1' then
            counter <= counter + 1;
            clk_div <= counter(19);
        end if;
    end process;

    -- Drive columns based on col_sel
    with col_sel select
        col <= "0111" when "00",
               "1011" when "01",
               "1101" when "10",
               "1110" when "11",
               "1111" when others;

    -- Keypad scanning and display latching process
    process(clk_div)
    begin
      if clk_div'event and clk_div = '1' then
          case col_sel is
              when "00" =>
                  if row(3) = '0' then display_digit <= "0001"; --1
                  elsif row(2) = '0' then display_digit <= "0100"; --4
                  elsif row(1) = '0' then display_digit <= "0111"; --7
                  elsif row(0) = '0' then display_digit <= "0000"; --0
                  end if;
              when "01" =>
                  if row(3) = '0' then display_digit <= "0010"; --2
                  elsif row(2) = '0' then display_digit <= "0101"; --5
                  elsif row(1) = '0' then display_digit <= "1000"; --8
                  elsif row(0) = '0' then display_digit <= "1111"; --F
                  end if;
              when "10" =>
                  if row(3) = '0' then display_digit <= "0011"; --3
                  elsif row(2) = '0' then display_digit <= "0110"; --6
                  elsif row(1) = '0' then display_digit <= "1001"; --9
                  elsif row(0) = '0' then display_digit <= "1110"; --E
                  end if;
              when "11" =>
                  if row(3) = '0' then display_digit <= "1010"; --A
                  elsif row(2) = '0' then display_digit <= "1011"; --B
                  elsif row(1) = '0' then display_digit <= "1100"; --C
                  elsif row(0) = '0' then display_digit <= "1101"; --D
                  end if;
              when others =>
                  null;
          end case;
```

```
            -- Cycle col_sel to scan all columns
            if col_sel = "11" then
                col_sel <= "00";
            else
                col_sel <= std_logic_vector(unsigned(col_sel) + 1);
            end if;
        end if;
    end process;

    -- Seven-segment decoder
    process(display_digit)
    begin
        case display_digit is
            when "0000" => seg <= "1000000";  -- 0
            when "0001" => seg <= "1111001";  -- 1
            when "0010" => seg <= "0100100";  -- 2
            when "0011" => seg <= "0110000";  -- 3
            when "0100" => seg <= "0011001";  -- 4
            when "0101" => seg <= "0010010";  -- 5
            when "0110" => seg <= "0000010";  -- 6
            when "0111" => seg <= "1111000";  -- 7
            when "1000" => seg <= "0000000";  -- 8
            when "1001" => seg <= "0010000";  -- 9
            when "1010" => seg <= "0001000";  -- A
            when "1011" => seg <= "0000011";  -- B
            when "1100" => seg <= "1000110";  -- C
            when "1101" => seg <= "0100001";  -- D
            when "1110" => seg <= "0000110";  -- E
            when "1111" => seg <= "0001110";  -- F
            when others => seg <= "1111111";  -- Blank
        end case;
    end process;

end Behavioral;
```