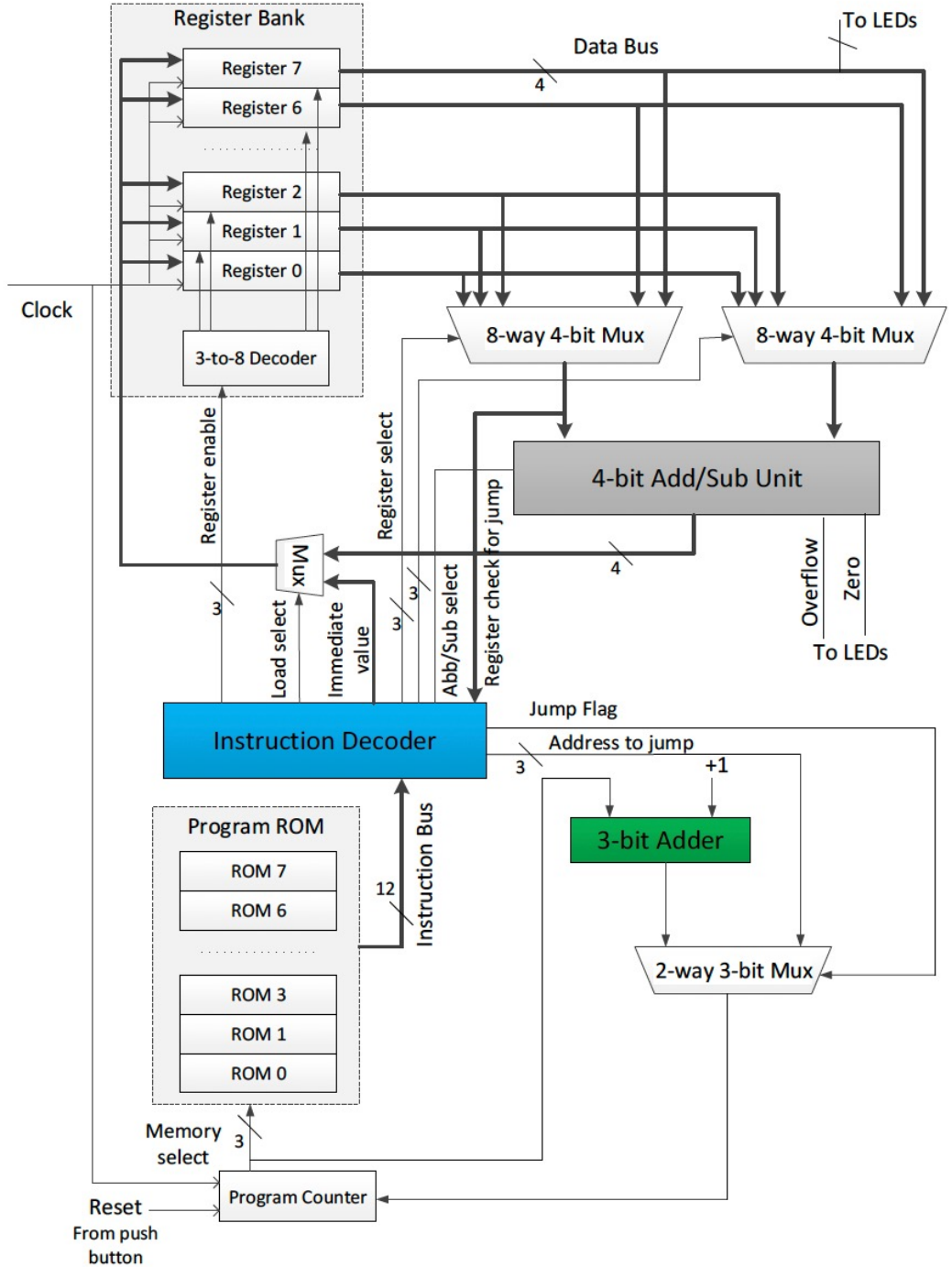
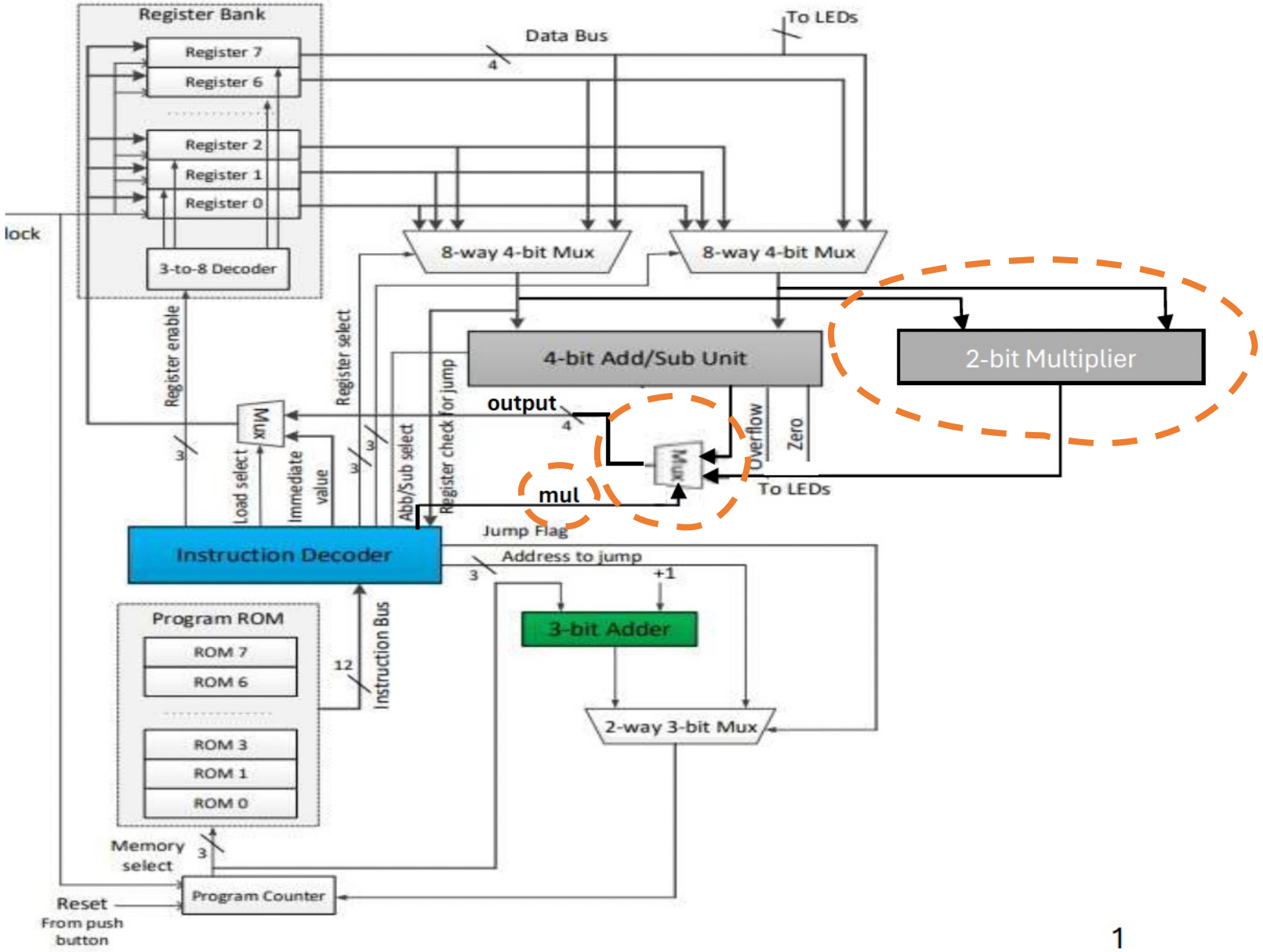
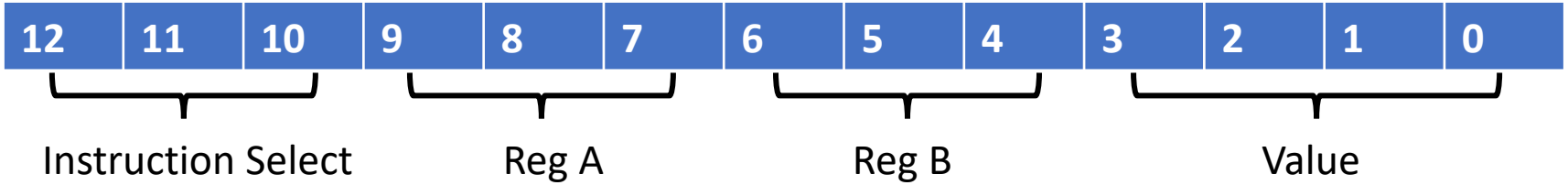

Register Bank
Register 7
Register 6
Register 2
Register 1
Register 0
3-to-8 Decoder
Data Bus
4
To LEDs
Clock
8-way 4-bit Mux
8-way 4-bit Mux
Register enable
Register select
4-bit Add/Sub Unit
Register check for jump
Mux
4
Overflow
Zero
To LEDs
3
Load select
Immediate value
3
3
Abb/Sub select
Jump Flag
Instruction Decoder
Address to jump
3
+1
3-bit Adder
Program ROM
ROM 7
ROM 6
ROM 3
ROM 1
ROM 0
12
Instruction Bus
2-way 3-bit Mux
Memory select
3
Program Counter
Reset
From push button


Register Bank
Register 7
Register 6
Register 2
Register 1
Register 0
3-to-8 Decoder
Data Bus
4
To LEDs
lock
8-way 4-bit Mux
8-way 4-bit Mux
4-bit Add/Sub Unit
2-bit Multiplier
Register enable
Register select
Register check for jump
output
4
Mux
Overflow
Zero
To LEDs
mul
Load select
Immediate value
Abb/Sub select
3
Jump Flag
Instruction Decoder
Address to jump
3
+1
3-bit Adder
Program ROM
ROM 7
ROM 6
ROM 3
ROM 1
ROM 0
12
Instruction Bus
2-way 3-bit Mux
Memory select
3
Reset
From push button
Program Counter



“010” – Move instruction
“000” – Add instruction
“001” – Substraction
“011” – Jump if zero instruction
“100” – Multiply instruction

```
"0100100000011",-- Mov R2, 3
"0101110000010",-- Mov R7, 2
"0011110100000",-- Sub R7,R2
"1011110000000",-- Neg R7
"0100110000010",-- Mov R3, 2
"0001110110000",-- Add R7, R3
"1001110110000",-- Mul R7, R3
"0110000000001"-- JMZ( check R0 )
```

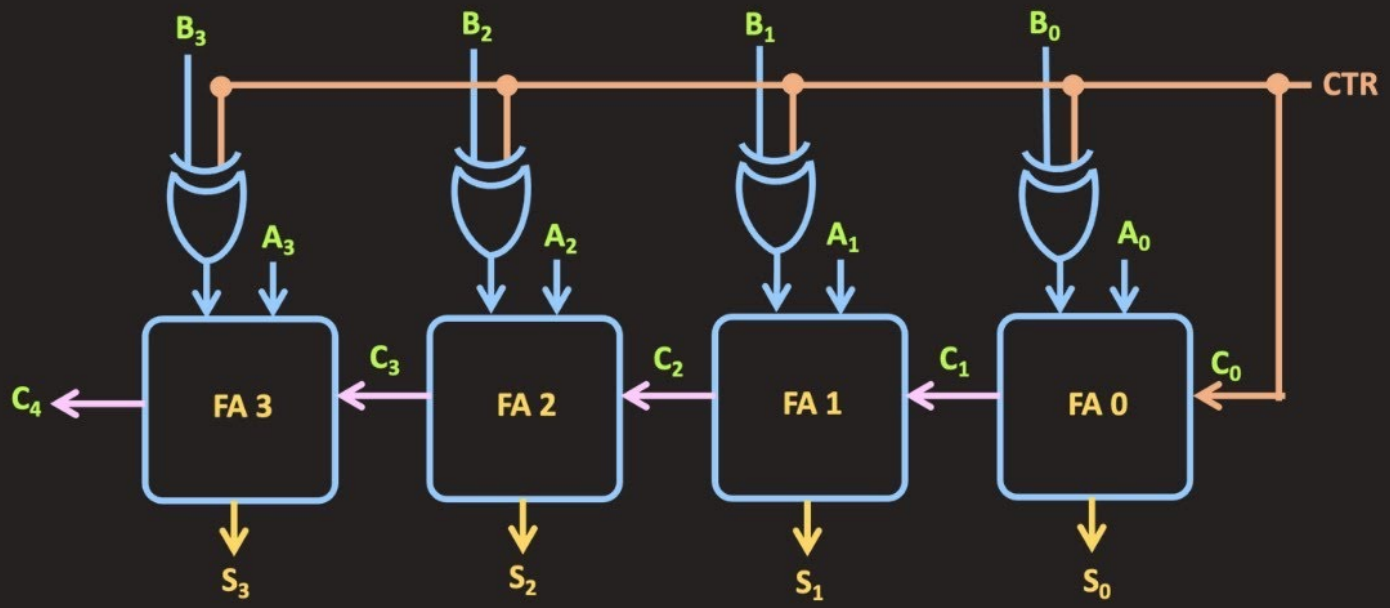
```
"0101110000001", --Mov R7, 1
"0100100000010", --Mov R2, 2
"0001110100000", --Add R7, R2
"0100110000011", --Mov R3, 3
"0001110110000", --Add R7, R3
"0111110000001", --JMZ R7, 1
"0001110000000", --Add R7, R0
"0110000000110"  --JMZ R0, 6
```

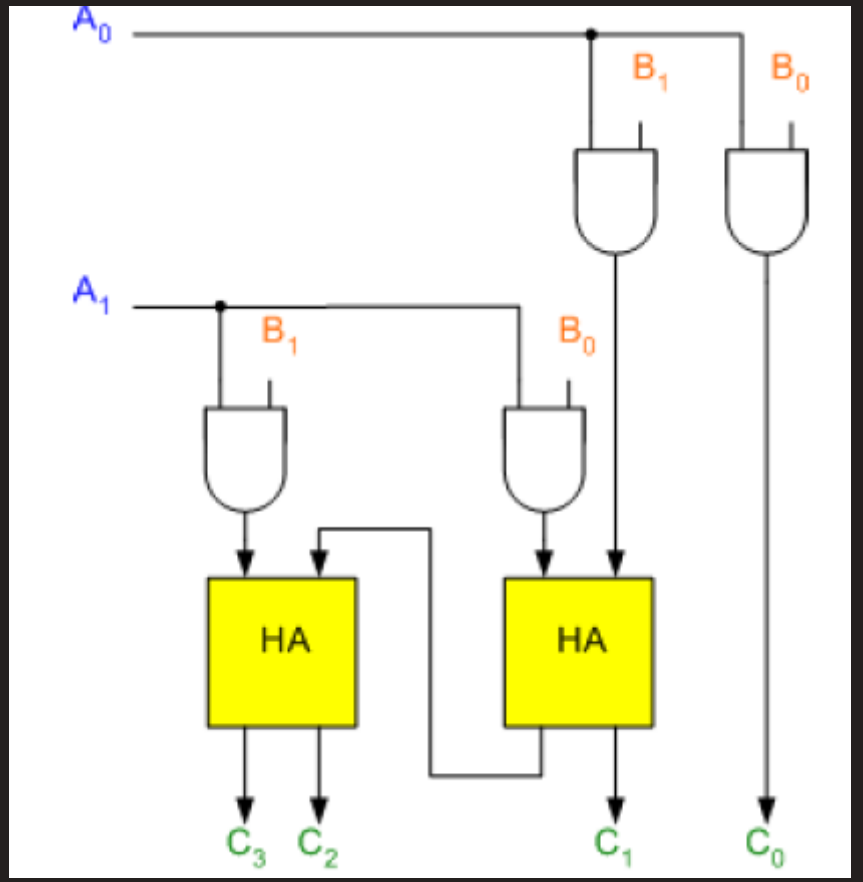
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Instruction_Decoder is
    Port ( I : in STD_LOGIC_VECTOR (12 downto 0);
           Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
           Load_Select : out STD_LOGIC;
           Imm_Value : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_1 : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_2 : out STD_LOGIC_VECTOR (2 downto 0);
           Add_Sub : out STD_LOGIC;
           Jump_Flag : out STD_LOGIC;
           Address : out STD_LOGIC_VECTOR (2 downto 0);
           Mul: out STD_LOGIC
           );
end Instruction_Decoder;

architecture Behavioral of Instruction_Decoder is

component Mux_2_to_1_3bit
    Port ( Sel : in STD_LOGIC;
           D0 : in STD_LOGIC_VECTOR (2 downto 0);
           D1 : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end component;

signal Ins : STD_LOGIC_VECTOR (2 downto 0);
signal RegA : STD_LOGIC_VECTOR (2 downto 0);
signal RegB : STD_LOGIC_VECTOR (2 downto 0);
signal Data : STD_LOGIC_VECTOR (3 downto 0);
signal Sel: STD_LOGIC;
```

```vhdl
begin
Ins <= I(12 downto 10);
RegA <= I(9 downto 7);
RegB <= I(6 downto 4);


with Ins select Reg_select_1<=
        "000" when "101",
        RegA when others;
with Ins select Reg_select_2<=
        RegA when "101",
        RegB when others;


Data <= I(3 downto 0);

Sel <= NOT(Ins(1)) AND Ins(0);
Load_Select <= Ins(1) AND NOT (Ins(0)) ;
Add_Sub <= Ins(0);

Jump_Flag <= Ins(1) AND Ins(0) AND NOT( Reg_Check_Jump(3)
                    OR Reg_Check_Jump(2) OR Reg_Check_Jump(1)OR Reg_Check_Jump(0));
Reg_Enable <= RegA;
Imm_Value <= Data;
Address <= Data(2 downto 0);
Mul <= Ins(2) and not(Ins(0)) and not(Ins(1));

end Behavioral;
```

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

entity Slow_Clk is
    Port ( Clk_in : in STD_LOGIC;
           Clk_out : out STD_LOGIC);
end Slow_Clk;

architecture Behavioral of Slow_Clk is

signal count : integer := 1;
signal clk_status : std_logic := '0';

begin
    process (Clk_in) begin
        if (rising_edge(Clk_in)) then
            count <= count + 1;
            if (count = 100000000)then
                clk_status <= not clk_status;
                Clk_out <= clk_status;
                count <= 1;
            end if;
        end if;
    end process;
end Behavioral;
```

```vhdl
Register_4bit_0 : Register_4bit
 PORT MAP(
    D => "0000",
    En => Y(0),
    Clk => Clk,
    Reset => Reg_S,
    Q => R0
    );
```


B3
B2
B1
B0
CTR
A3
A2
A1
A0
C4
C3
C2
C1
C0
FA 3
FA 2
FA 1
FA 0
S3
S2
S1
S0


A0
B1
B0
A1
B1
B0
HA
HA
C3
C2
C1
C0

LUT Count

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	39	0	20800	0.19
LUT as Logic	39	0	20800	0.19
LUT as Memory	0	0	9600	0.00
Slice Registers	49	0	41600	0.12
Register as Flip Flop	49	0	41600	0.12
Register as Latch	0	0	41600	0.00
F7 Muxes	0	0	16300	0.00
F8 Muxes	0	0	8150	0.00