Computer Architecture - Project 1B
Datapath Design

Vsevolod Syrtsov
syrtsovv@tcd.ie
source: github.com/futurecertificate

Contents

1 VHDL Sources

1.1 Bit Slice Arithmetic Unit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity bit_slice_au is
Port (
S0: in std_logic;
S1: in std_logic;
A: in std_logic;
B: in std_logic;
Cin: in std_logic;

Gi: out std_logic;
Cout: out std_logic
);

end bit_slice_au;

architecture Behavioral of bit_slice_au is

COMPONENT b_logic_mux
Port (
S0 : in std_logic;
S1 : in std_logic;
B : in std_logic;
Y : out std_logic
);
END COMPONENT;

COMPONENT full_adder
Port (
X : in std_logic;
Y : in std_logic;
Cin : in std_logic;

Gi : out std_logic;
Cout : out std_logic
);
END COMPONENT;

signal Yi : std_logic;

begin

b_logic: b_logic_mux PORT MAP(
S0 => S0,
S1 => S1,
Y => Yi,
B => B
);

fa: full_adder PORT MAP(
X => A,
Y => Yi,
Cin => Cin,
Gi => Gi,
Cout => Cout
);

end Behavioral;
```

1.2 Bit Slice Logic Unit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity bit_slice_lu is
Port (
S0 : in std_logic;
S1 : in std_logic;
A : in std_logic;
B : in std_logic;

G: out std_logic
);

end bit_slice_lu;
```

```
architecture Behavioral of bit_slice_lu is
begin
G <= (A and B) after 1 ns when S0='0' and S1='0'  else
(A or B) after 1 ns when S0='0' and S1='1' else
(A xor B) after 1 ns when S0='1' and S1='0' else
(not A) after 1 ns when S0='1' and S1='1'  else
'0' after 5 ns;
end Behavioral;
```

1.3 Bit Slice ALU

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity bit_slice_alu is
Port (
 Cin: in std_logic;
 A: in std_logic;
 B: in std_logic;
 S0: in std_logic;
 S1: in std_logic;
 S2: in std_logic;

 Cout: out std_logic;
 Gi: out std_logic
);
end bit_slice_alu;

architecture Behavioral of bit_slice_alu is

COMPONENT bit_slice_au
Port (
S0: in std_logic;
S1: in std_logic;
A: in std_logic;
B: in std_logic;
Cin: in std_logic;

Gi: out std_logic;
Cout: out std_logic
```

```
);
END COMPONENT;

COMPONENT bit_slice_lu
Port (
S0 : in std_logic;
S1 : in std_logic;
A : in std_logic;
B : in std_logic;

G: out std_logic
);
END COMPONENT;

COMPONENT au_mux
Port (
G1 : in std_logic;
G2 : in std_logic;
S2 : in std_logic;

G : out std_logic);
END COMPONENT;

signal G1, G2 : std_logic;

begin

au: bit_slice_au PORT MAP(
S0 => S0,
S1 => S1,
A => A,
B => B,
Cin => Cin,
Gi => G1,
Cout => Cout
);

lu: bit_slice_lu PORT MAP(
S0 => S0,
S1 => S1,
A => A,
B => B,
G => G2
);

mux: au_mux PORT MAP(
G1=> G1,
G2=> G2,
S2=> S2,
G=>Gi
);



end Behavioral;
```

1.4 ALU

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

entity ALU is
Port (
A: in std_logic_vector(15 downto 0);
B: in std_logic_vector(15 downto 0);
GS: in std_logic_vector(3 downto 0);

G: out std_logic_vector(15 downto 0);

C: out std_logic;
V: out std_logic
);
end ALU;

architecture Behavioral of ALU is

COMPONENT bit_slice_alu
Port (
 Cin: in std_logic;
 A: in std_logic;
 B: in std_logic;
 S0: in std_logic;
 S1: in std_logic;
 S2: in std_logic;

 Cout: out std_logic;
 Gi: out std_logic
);
END COMPONENT;


signal G0, G1, G2, G3, G4, G5, G6, G7, G8, G9, G10, G11, G12, G13, G14, G15 : st
signal C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16 : s

begin


balu0: bit_slice_alu PORT MAP(
Cin => GS(0),
A => A(0),
B => B(0),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C1,
Gi => G0
);

balu1: bit_slice_alu PORT MAP(
Cin => C1,
A => A(1),
B => B(1),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C2,
Gi => G1
);

balu2: bit_slice_alu PORT MAP(
Cin => C2,
A => A(2),
B => B(2),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C3,
Gi => G2
);

balu3: bit_slice_alu PORT MAP(
Cin => C3,
A => A(3),
B => B(3),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C4,
Gi => G3
);

balu4: bit_slice_alu PORT MAP(
Cin => C4,
A => A(4),
B => B(4),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C5,
Gi => G4
);

balu5: bit_slice_alu PORT MAP(
Cin => C5,
A => A(5),
B => B(5),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C6,
Gi => G5
);
```

```
balu6: bit_slice_alu PORT MAP(
Cin => C6,
A => A(6),
B => B(6),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C7,
Gi => G6
);

balu7: bit_slice_alu PORT MAP(
Cin => C7,
A => A(7),
B => B(7),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C8,
Gi => G7
);

balu8: bit_slice_alu PORT MAP(
Cin => C8,
A => A(8),
B => B(8),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C9,
Gi => G8
);

balu9: bit_slice_alu PORT MAP(
Cin => C9,
A => A(9),
B => B(9),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C10,
Gi => G9
);

balu10: bit_slice_alu PORT MAP(
Cin => C10,
A => A(10),
B => B(10),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C11,
Gi => G10
);

balu11: bit_slice_alu PORT MAP(
Cin => C11,
A => A(11),
B => B(11),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C12,
Gi => G11
);

balu12: bit_slice_alu PORT MAP(
Cin => C12,
A => A(12),
B => B(12),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C13,
Gi => G12
);

balu13: bit_slice_alu PORT MAP(
Cin => C13,
A => A(13),
B => B(13),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C14,
Gi => G13
);

balu14: bit_slice_alu PORT MAP(
Cin => C14,
A => A(14),
B => B(14),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C15,
Gi => G14
);

balu15: bit_slice_alu PORT MAP(
Cin => C15,
A => A(15),
B => B(15),
S0 => GS(1),
S1 => GS(2),
S2 => GS(3),
Cout => C16,
Gi => G15
);

G <= G15 & G14 & G13 & G12 & G11 & G10 & G9 & G8 & G7 & G6 & G5 & G4 & G3 & G2 &
V <= C15 xor C16 after 1 ns;
C <= C16 after 1 ns;


end Behavioral;
```

1.5 Shifter

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

entity shifter is
Port (
B: in std_logic_vector(15 downto 0);
HS: in std_logic_vector(1 downto 0);
IL: in std_logic;
IR: in std_logic;

H: out std_logic_vector(15 downto 0)
);
end shifter;

architecture Behavioral of shifter is

COMPONENT shifter_mux is
Port (
    I0: in std_logic;
    I1: in std_logic;
    I2: in std_logic;
    HS: in std_logic_vector(1 downto 0);
    H: out std_logic
);

END COMPONENT;

signal H0, H1, H2, H3, H4, H5, H6, H7, H8, H9, H10, H11, H12, H13, H14, H15: std

begin

m0: shifter_mux PORT MAP(I0 => B(0), I1 => B(1), I2 => IL, HS => HS, H=>H0 ); --
m1: shifter_mux PORT MAP(I0 => B(1), I1 => B(2), I2 => B(0), HS => HS, H=>H1 );
m2: shifter_mux PORT MAP(I0 => B(2), I1 => B(3), I2 => B(1), HS => HS, H=>H2 );
m3: shifter_mux PORT MAP(I0 => B(3), I1 => B(4), I2 => B(2), HS => HS, H=>H3 );
m4: shifter_mux PORT MAP(I0 => B(4), I1 => B(5), I2 => B(3), HS => HS, H=>H4 );
m5: shifter_mux PORT MAP(I0 => B(5), I1 => B(6), I2 => B(4), HS => HS, H=>H5 );
m6: shifter_mux PORT MAP(I0 => B(6), I1 => B(7), I2 => B(5), HS => HS, H=>H6 );
m7: shifter_mux PORT MAP(I0 => B(7), I1 => B(8), I2 => B(6), HS => HS, H=>H7 );
m8: shifter_mux PORT MAP(I0 => B(8), I1 => B(9), I2 => B(7), HS => HS, H=>H8 );
m9: shifter_mux PORT MAP(I0 => B(9), I1 => B(10), I2 => B(8), HS => HS, H=>H9 );
m10: shifter_mux PORT MAP(I0 => B(10), I1 => B(11), I2 => B(9), HS => HS, H=>H10
m11: shifter_mux PORT MAP(I0 => B(11), I1 => B(12), I2 => B(10), HS => HS, H=>H1
m12: shifter_mux PORT MAP(I0 => B(12), I1 => B(13), I2 => B(11), HS => HS, H=>H1
m13: shifter_mux PORT MAP(I0 => B(13), I1 => B(14), I2 => B(12), HS => HS, H=>H1
m14: shifter_mux PORT MAP(I0 => B(14), I1 => B(15), I2 => B(13), HS => HS, H=>H1
m15: shifter_mux PORT MAP(I0 => B(15), I1 => IR, I2 => B(14), HS => HS, H=>H15 )

H <= H15 & H14 & H13 & H12 & H11 & H10 & H9 & H8 & H7 & H6 & H5 & H4 & H3 & H2 &

end Behavioral;
```

1.6 Function Unit

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

entity function_unit is
Port (
A: in std_logic_vector(15 downto 0);
B: in std_logic_vector(15 downto 0);
FS: in std_logic_vector(4 downto 0);

V: out std_logic;
C: out std_logic;
Z: out std_logic; --return 1 if 0
N: out std_logic; --return MSB
F: out std_logic_vector(15 downto 0)
);
end function_unit;

architecture Behavioral of function_unit is

COMPONENT shifter is
Port (
B: in std_logic_vector(15 downto 0);
HS: in std_logic_vector(1 downto 0);
IL: in std_logic;
IR: in std_logic;

H: out std_logic_vector(15 downto 0)
);
END COMPONENT;

COMPONENT ALU is
Port (
A: in std_logic_vector(15 downto 0);
B: in std_logic_vector(15 downto 0);
GS: in std_logic_vector(3 downto 0);

G: out std_logic_vector(15 downto 0);
C: out std_logic;
V: out std_logic
);

END COMPONENT;

COMPONENT mux_2_16 is
port (
In0 : in std_logic_vector(15 downto 0);
In1 : in std_logic_vector(15 downto 0);
s : in std_logic;
Z : out std_logic_vector(15 downto 0)
);

END COMPONENT;

signal G1, H1: std_logic_vector(15 downto 0);

begin



alu0: ALU PORT MAP(
A => A,
B => B,
GS =>FS(3 downto 0),
G => G1,
C => C,
V => V
);
```

```
shifter0: shifter PORT MAP(
B => B,
HS => FS(3 downto 2),
IL => '0',
IR => '0',
H => H1
);

mux: mux_2_16 PORT MAP(
In0 => H1,
In1 => G1,
s =>FS(4),
Z => F
);

N <= G1(15);
Z <= '1' after 5 ns when G1="0000000000000000"else
'0' after 5 ns;

end Behavioral;
```

1.7 Datapath

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


entity datapath_1B is
Port (
AS0: in std_logic;
AS1: in std_logic;
AS2: in std_logic;

BS0: in std_logic;
BS1: in std_logic;
BS2: in std_logic;

des_A0: in std_logic;
des_A1: in std_logic;
```

```
des_A2: in std_logic;

MB_Select: in std_logic;
MD_Select: in std_logic;

Clk: in std_logic;

LE: in std_logic;
FS: in std_logic_vector(4 downto 0);
Data_in: in std_logic_vector(15 downto 0);

Constant_In: in std_logic_vector(15 downto 0);

A_out : out std_logic_vector(15 downto 0);
B_out : out std_logic_vector(15 downto 0);

V: out std_logic;
C: out std_logic;
Z: out std_logic;
N: out std_logic
);
end datapath_1B;

architecture Behavioral of datapath_1B is

COMPONENT reg_file PORT(
LE : in std_logic;

AS0 : in std_logic;
AS1 : in std_logic;
AS2 : in std_logic;

BS0 : in std_logic;
BS1 : in std_logic;
BS2 : in std_logic;

des_A0 : in std_logic;
des_A1 : in std_logic;
des_A2 : in std_logic;

Clk : in std_logic;
```

```
D_data : in std_logic_vector(15 downto 0);
B_data : out std_logic_vector(15 downto 0);
A_data : out std_logic_vector(15 downto 0)

);
END COMPONENT;

COMPONENT function_unit
Port (
A: in std_logic_vector(15 downto 0);
B: in std_logic_vector(15 downto 0);
FS: in std_logic_vector(4 downto 0);

V: out std_logic;
C: out std_logic;
Z: out std_logic; --return 1 if 0
N: out std_logic; --return MSB
F: out std_logic_vector(15 downto 0)
);

END COMPONENT;

COMPONENT mux_2_16
port (
In0 : in std_logic_vector(15 downto 0);
In1 : in std_logic_vector(15 downto 0);
s : in std_logic;
Z : out std_logic_vector(15 downto 0)
);
END COMPONENT;

signal A_data, B_data, B_data_mux_src : std_logic_vector(15 downto 0); --Data co
signal FZ, D: std_logic_vector(15 downto 0); --Data coming from function unit (


begin

fu: function_unit PORT MAP(
A=> A_data,
B=> B_data_mux_src,
FS => FS,
V => V,
C => C,
Z => Z,
N => N,
F=>FZ
);

muxB: mux_2_16 PORT MAP(
In0 => B_data,
In1 => Constant_In,
s => MB_Select,
Z => B_data_mux_src
);

muxD: mux_2_16 PORT MAP(
In0 => FZ,
In1 => Data_in,
s => MD_Select,
Z => D
);

regf: reg_file PORT MAP(
LE => LE,
AS0 => AS0,
AS1 => AS1,
AS2 => AS2,

BS0 => BS0,
BS1 => BS1,
BS2 => BS2,

des_A0 => des_A0,
des_A1 => des_A1,
des_A2 => des_A2,

Clk => Clk,
D_Data => D,
A_Data => A_Data,
B_Data => B_Data
);
```

```
A_Out <= A_data after 1 ns;
B_Out <= B_data_mux_src after 1 ns;
end Behavioral;
```

2 Component Test Benches

2.1 Bit Slice Arithmetic Unit Testbench

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

entity tb_bs_au is
end tb_bs_au;

architecture Behavioral of tb_bs_au is
COMPONENT bit_slice_au
Port (
A: in std_logic;
Cin: in std_logic;
S0: in std_logic;
S1: in std_logic;
B: in std_logic;

Gi: out std_logic;
Cout: out std_logic
);
END COMPONENT;


--Inputs
signal A: std_logic;
signal Cin: std_logic;
signal S0: std_logic;
signal S1: std_logic;
signal B: std_logic;

--Outputs

signal Gi, Cout: std_logic;


begin
```

```
uut: bit_slice_au PORT MAP(
S0 => S0,
S1 => S1,
A => A,
B => B,
Cin => Cin,
Gi => Gi,
Cout => Cout
);

stim_proc: process
begin

wait for 5ns;
A <= '0';
Cin <= '0';
S0 <= '0';
S1 <= '0';
B <= '0';

wait for 5ns;
A <= '0';
Cin <= '0';
S0 <= '0';
S1 <= '0';
B <= '1';

wait for 5ns;
A <= '0';
Cin <= '0';
S0 <= '0';
S1 <= '1';
B <= '0';

wait for 5ns;
A <= '0';
Cin <= '0';
S0 <= '0';
S1 <= '1';
```

```vhdl
B <= '1';

wait for 5ns;
A <= '0';
Cin <= '0';
S0 <= '1';
S1 <= '0';
B <= '0';

wait for 5ns;
A <= '0';
Cin <= '0';
S0 <= '1';
S1 <= '0';
B <= '1';

wait for 5ns;
A <= '0';
Cin <= '0';
S0 <= '1';
S1 <= '1';
B <= '0';

wait for 5ns;
A <= '0';
Cin <= '0';
S0 <= '1';
S1 <= '1';
B <= '1';

wait for 5ns;
A <= '0';
Cin <= '1';
S0 <= '0';
S1 <= '0';
B <= '0';

wait for 5ns;
A <= '0';
Cin <= '1';
S0 <= '0';
```

```vhdl
S1 <= '0';
B <= '1';

wait for 5ns;
A <= '0';
Cin <= '1';
S0 <= '0';
S1 <= '1';
B <= '0';

wait for 5ns;
A <= '0';
Cin <= '1';
S0 <= '0';
S1 <= '1';
B <= '1';

wait for 5ns;
A <= '0';
Cin <= '1';
S0 <= '1';
S1 <= '0';
B <= '0';

wait for 5ns;
A <= '0';
Cin <= '1';
S0 <= '1';
S1 <= '0';
B <= '1';

wait for 5ns;
A <= '0';
Cin <= '1';
S0 <= '1';
S1 <= '1';
B <= '0';

wait for 5ns;
A <= '0';
Cin <= '1';
```

```
S0 <= '1';
S1 <= '1';
B <= '1';

wait for 5ns;
A <= '1';
Cin <= '0';
S0 <= '0';
S1 <= '0';
B <= '0';

wait for 5ns;
A <= '1';
Cin <= '0';
S0 <= '0';
S1 <= '0';
B <= '1';

wait for 5ns;
A <= '1';
Cin <= '0';
S0 <= '0';
S1 <= '1';
B <= '0';

wait for 5ns;
A <= '1';
Cin <= '0';
S0 <= '0';
S1 <= '1';
B <= '1';

wait for 5ns;
A <= '1';
Cin <= '0';
S0 <= '1';
S1 <= '0';
B <= '0';

wait for 5ns;
A <= '1';
```

```
Cin <= '0';
S0 <= '1';
S1 <= '0';
B <= '1';

wait for 5ns;
A <= '1';
Cin <= '0';
S0 <= '1';
S1 <= '1';
B <= '0';

wait for 5ns;
A <= '1';
Cin <= '0';
S0 <= '1';
S1 <= '1';
B <= '1';

wait for 5ns;
A <= '1';
Cin <= '1';
S0 <= '0';
S1 <= '0';
B <= '0';

wait for 5ns;
A <= '1';
Cin <= '1';
S0 <= '0';
S1 <= '0';
B <= '1';

wait for 5ns;
A <= '1';
Cin <= '1';
S0 <= '0';
S1 <= '1';
B <= '0';

wait for 5ns;
```

```
A <= '1';
Cin <= '1';
S0 <= '0';
S1 <= '1';
B <= '1';

wait for 5ns;
A <= '1';
Cin <= '1';
S0 <= '1';
S1 <= '0';
B <= '0';

wait for 5ns;
A <= '1';
Cin <= '1';
S0 <= '1';
S1 <= '0';
B <= '1';

wait for 5ns;
A <= '1';
Cin <= '1';
S0 <= '1';
S1 <= '1';
B <= '0';

wait for 5ns;
A <= '1';
Cin <= '1';
S0 <= '1';
S1 <= '1';
B <= '1';



end process;

end Behavioral;
```

2.2 Bit Slice Logic Unit Testbench

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

entity tb_bs_lu is
--  Port ( );
end tb_bs_lu;

architecture Behavioral of tb_bs_lu is
COMPONENT bit_slice_lu
Port (
S0 : in std_logic;
S1 : in std_logic;
A : in std_logic;
B : in std_logic;

G: out std_logic
);
END COMPONENT;


--Inputs
signal A: std_logic;
signal B: std_logic;

signal S0: std_logic;
signal S1: std_logic;

--Outputs

signal G: std_logic;


begin



uut: bit_slice_lu PORT MAP(
A => A,
B => B,
S0 => S0,
S1 => S1,
G => G
);

stim_proc: process
begin

wait for 5ns;
A <= '0';
B<= '0';
S0<='0';
S1<='0';

wait for 5ns;
A <= '0';
B<= '0';
S0<='0';
S1<='1';

wait for 5ns;
A <= '0';
B<= '0';
S0<='1';
S1<='0';

wait for 5ns;
A <= '0';
B<= '0';
S0<='1';
S1<='1';

wait for 5ns;
A <= '0';
B<= '1';
S0<='0';
S1<='0';

wait for 5ns;
```

```
A <= '0';
B<= '1';
S0<='0';
S1<='1';

wait for 5ns;
A <= '0';
B<= '1';
S0<='1';
S1<='0';

wait for 5ns;
A <= '0';
B<= '1';
S0<='1';
S1<='1';

wait for 5ns;
A <= '1';
B<= '0';
S0<='0';
S1<='0';

wait for 5ns;
A <= '1';
B<= '0';
S0<='0';
S1<='1';

wait for 5ns;
A <= '1';
B<= '0';
S0<='1';
S1<='0';

wait for 5ns;
A <= '1';
B<= '0';
S0<='1';
S1<='1';
```

```
wait for 5ns;
A <= '1';
B<= '1';
S0<='0';
S1<='0';

wait for 5ns;
A <= '1';
B<= '1';
S0<='0';
S1<='1';

wait for 5ns;
A <= '1';
B<= '1';
S0<='1';
S1<='0';

wait for 5ns;
A <= '1';
B<= '1';
S0<='1';
S1<='1';

end process;

end Behavioral;
```

2.3 Bit Slice ALU Testbench

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

entity tb_bs_alu is
end tb_bs_alu;

architecture Behavioral of tb_bs_alu is
COMPONENT bit_slice_alu
Port (
 Cin: in std_logic;
 A: in std_logic;
 B: in std_logic;
 S0: in std_logic;
 S1: in std_logic;
 S2: in std_logic;

 Cout: out std_logic;
 Gi: out std_logic
);
END COMPONENT;


--Inputs
signal A: std_logic;
signal B: std_logic;
signal Cin: std_logic;
signal S0: std_logic := '0';
signal S1: std_logic := '1';
signal S2: std_logic := '1';

--Outputs

signal Gi, Cout: std_logic;


begin


uut: bit_slice_alu PORT MAP(
S0 => S0,
S1 => S1,
S2 => S2,
A => A,
B => B,
Cin => Cin,
Gi => Gi,
Cout => Cout
);

stim_proc: process
```

```
begin

wait for 5ns;
A <= '0';
B <= '0';
Cin <= '0';

wait for 5ns;
A <= '0';
B <= '0';
Cin <= '1';

wait for 5ns;
A <= '0';
B <= '1';
Cin <= '0';

wait for 5ns;
A <= '0';
B <= '1';
Cin <= '1';

wait for 5ns;
A <= '1';
B <= '0';
Cin <= '0';

wait for 5ns;
A <= '1';
B <= '0';
Cin <= '1';

wait for 5ns;
A <= '1';
B <= '1';
Cin <= '0';

wait for 5ns;
A <= '1';
B <= '1';
Cin <= '1';
```

```
end process;
end Behavioral;
```

2.4 ALU Testbench

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

entity tb_ALU is
--  Port ( );
end tb_ALU;

architecture Behavioral of tb_ALU is
COMPONENT ALU
Port (
A: in std_logic_vector(15 downto 0);
B: in std_logic_vector(15 downto 0);
GS: in std_logic_vector(3 downto 0);

G: out std_logic_vector(15 downto 0);

C: out std_logic;
V: out std_logic
);
    END COMPONENT;

     --Inputs

   signal A : std_logic_vector(15 downto 0);
   signal B : std_logic_vector(15 downto 0);
   signal GS: std_logic_vector(3 downto 0) := (others => '0');

         --Outputs
   signal G : std_logic_vector(15 downto 0);
   signal C : std_logic;
   signal V : std_logic;
```

```
begin

uut: ALU PORT MAP(
A => A,
B => B,
GS => GS,

G => G,

C => C,
V => V
);

stim_proc: process
begin
wait for 50ns;
A <= x"1234";
B <= x"4321";

wait for 50ns;
GS <= x"0";

wait for 50ns;
GS <= x"1";

wait for 50ns;
GS <= x"2";

wait for 50ns;
GS <= x"3";

wait for 50ns;
GS <= x"4";

wait for 50ns;
GS <= x"5";

wait for 50ns;
GS <= x"6";

wait for 50ns;
```

```
GS <= x"7";

wait for 50ns;
GS <= x"8";

wait for 50ns;
GS <= x"A";

wait for 50ns;
GS <= x"C";

wait for 50ns;
GS <= x"E";


end process;
end Behavioral;
```

2.5 Shifter Testbench

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

entity tb_shifter is
--  Port ( );
end tb_shifter;

architecture Behavioral of tb_shifter is

COMPONENT shifter
Port (
B: in std_logic_vector(15 downto 0);
HS: in std_logic_vector(1 downto 0);
IL: in std_logic;
IR: in std_logic;

H: out std_logic_vector(15 downto 0)
);
END COMPONENT;
```

```
   --Inputs

   signal B : std_logic_vector(15 downto 0) := (others => '0');
   signal HS : std_logic_vector(1 downto 0) := (others => '0');
   signal IL: std_logic := '0';
   signal IR: std_logic := '0';

         --Outputs
   signal H : std_logic_vector(15 downto 0);


begin

uut: shifter PORT MAP(
B => B,
HS => HS,
IL => IL,
IR => IR,
H => H
);

stim_proc: process
begin
B <= x"FFFF";
wait for 50ns;
HS <= "00";

wait for 50ns;
HS <= "01";

wait for 50ns;
HS <= "10";


end process;
end Behavioral;
```

2.6 Function Unit Testbench

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

entity tb_function_unit is
--  Port ( );
end tb_function_unit;

architecture Behavioral of tb_function_unit is

COMPONENT function_unit
Port (
A: in std_logic_vector(15 downto 0);
B: in std_logic_vector(15 downto 0);
FS: in std_logic_vector(4 downto 0);

V: out std_logic;
C: out std_logic;
Z: out std_logic; --return 1 if 0
N: out std_logic; --return MSB
F: out std_logic_vector(15 downto 0)
);
END COMPONENT;

   --Inputs

   signal A: std_logic_vector(15 downto 0) := (others => '0');
   signal B: std_logic_vector(15 downto 0) := (others => '0');
   signal FS: std_logic_vector(4 downto 0);

         --Outputs
    signal V : std_logic;
    signal C : std_logic;
    signal Z : std_logic;
    signal N : std_logic;
    signal F : std_logic_vector(15 downto 0);
```

```
begin

uut: function_unit PORT MAP(
A=>A,
B => B,
FS => FS,
V => V,
C => C,
Z => Z,
N => N,
F => F
);

stim_proc: process
begin

wait for 50 ns;
A<= x"B00B";
B<= x"BEEF";

wait for 50 ns;
FS<="00000";

wait for 50 ns;
FS<="00001";

wait for 50 ns;
FS<="00010";

wait for 50 ns;
FS<="00011";

wait for 50 ns;
FS<="00100";

wait for 50 ns;
FS<="00101";

wait for 50 ns;
FS<="00110";
```

```
wait for 50 ns;
FS<="00111";

wait for 50 ns;
FS<="01000";

wait for 50 ns;
FS<="01010";

wait for 50 ns;
FS<="01100";

wait for 50 ns;
FS<="01110";

wait for 50 ns;
FS<="10000";

wait for 50 ns;
FS<="10100";

wait for 50 ns;
FS<="11000";


end process;
end Behavioral;
```

2.7 Datapath Testbench

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

entity tb_datapath is
--  Port ( );
end tb_datapath;

architecture Behavioral of tb_datapath is
```

```
COMPONENT datapath_1B
Port (
AS0: in std_logic;
AS1: in std_logic;
AS2: in std_logic;

BS0: in std_logic;
BS1: in std_logic;
BS2: in std_logic;

des_A0: in std_logic;
des_A1: in std_logic;
des_A2: in std_logic;

MB_Select: in std_logic;
MD_Select: in std_logic;

Clk: in std_logic;

LE: in std_logic;
FS: in std_logic_vector(4 downto 0);
Data_in: in std_logic_vector(15 downto 0);

Constant_In: in std_logic_vector(15 downto 0);

A_out : out std_logic_vector(15 downto 0);
B_out : out std_logic_vector(15 downto 0);

V: out std_logic;
C: out std_logic;
Z: out std_logic;
N: out std_logic
);
END COMPONENT;

   --Inputs

signal AS0: std_logic := '0';
signal AS1: std_logic := '0';
signal AS2: std_logic := '0';
signal BS0: std_logic := '0';
signal BS1: std_logic := '0';
signal BS2: std_logic := '0';

signal des_A0: std_logic := '0';
signal des_A1: std_logic := '0';
signal des_A2: std_logic := '0';

signal MB_Select: std_logic := '0'; --will be under change
signal MD_Select: std_logic := '1'; --will be under change

signal Clk: std_logic;

signal LE: std_logic := '1';
signal FS: std_logic_vector(4 downto 0) := (others => '0'); -- will be under cha
signal Data_in: std_logic_vector(15 downto 0) := x"DADA";

signal Constant_In: std_logic_vector(15 downto 0) := x"1234";

    --Outputs

signal A_out : std_logic_vector(15 downto 0);
signal B_out : std_logic_vector(15 downto 0);

signal V: std_logic;
signal C: std_logic;
signal Z: std_logic;
signal N: std_logic;

begin

uut: datapath_1B PORT MAP(
AS0=>AS0,
AS1=>AS1,
AS2=>AS2,

BS0=>BS0,
BS1=>BS1,
BS2=>BS2,

des_A0 => des_A0,
des_A1 => des_A1,
des_A2 => des_A2,

MB_Select => MB_Select,
MD_Select => MD_Select,

Clk=>Clk,

LE=>LE,
FS=>FS,
Data_in=>Data_in,

Constant_In=>Constant_In,
A_out=>A_out,
B_out=>B_out,

V=>V,
C=>C,
Z=>Z,
N=>N
);

clk_process :process
   begin
                Clk <= '0';
                wait for 5ns;
                Clk <= '1';
                wait for 5ns;
   end process;

stim_proc: process
begin

--test arithmetic unit with MB/MD Select
wait for 50 ns;
MB_Select <= '0';
MD_Select <='0';
FS <= "00001";

wait for 50 ns;
MB_Select <= '0';
```

```
MD_Select <= '0';
FS <= "00010";

wait for 50 ns;
MB_Select <= '0';
MD_Select <='0';
FS <= "00011";


wait for 50 ns;
MB_Select <= '0';
MD_Select <='1';
FS <= "00001";

wait for 50 ns;
MB_Select <= '0';
MD_Select <= '1';
FS <= "00010";

wait for 50 ns;
MB_Select <= '0';
MD_Select <='1';
FS <= "00011";


wait for 50 ns;
MB_Select <= '1';
MD_Select <='0';
FS <= "00001";

wait for 50 ns;
MB_Select <= '1';
MD_Select <= '0';
FS <= "00010";

wait for 50 ns;
MB_Select <= '1';
MD_Select <='0';
FS <= "00011";

--test logic unit
```

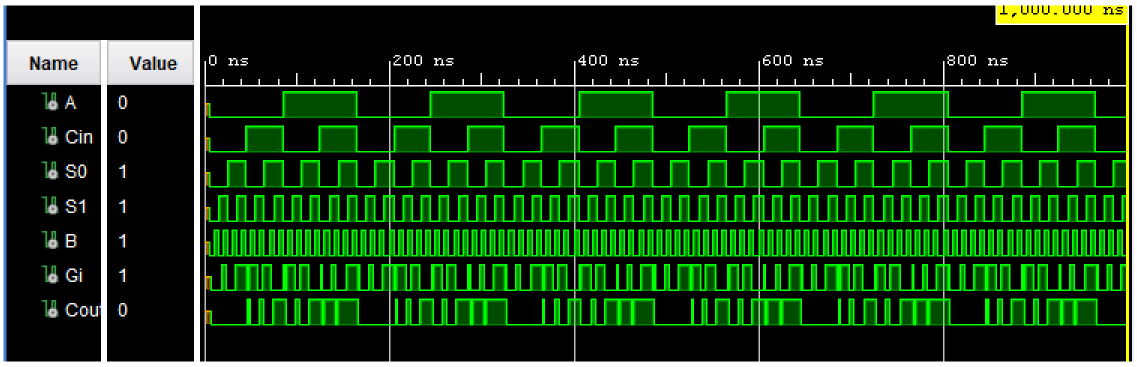
```vhdl
wait for 50 ns;
MB_Select <= '0';
MD_Select <='0';
FS <= "01000";

wait for 50 ns;
MB_Select <= '0';
MD_Select <= '0';
FS <= "01010";

wait for 50 ns;
MB_Select <= '0';
MD_Select <='0';
FS <= "01110";


wait for 50 ns;
MB_Select <= '0';
MD_Select <='1';
FS <= "01000";

wait for 50 ns;
MB_Select <= '0';
MD_Select <= '1';
FS <= "01010";

wait for 50 ns;
MB_Select <= '0';
MD_Select <='1';
FS <= "01110";


wait for 50 ns;
MB_Select <= '1';
MD_Select <='0';
FS <= "01000";

wait for 50 ns;
MB_Select <= '1';
MD_Select <= '0';
```

```
FS <= "01010";

wait for 50 ns;
MB_Select <= '1';
MD_Select <='0';
FS <= "01110";

--test shifter

wait for 50 ns;
MB_Select <= '0';
MD_Select <='0';
FS <= "10000";

wait for 50 ns;
MB_Select <= '0';
MD_Select <= '0';
FS <= "10100";

wait for 50 ns;
MB_Select <= '0';
MD_Select <='0';
FS <= "11000";


wait for 50 ns;
MB_Select <= '0';
MD_Select <='1';
FS <= "10000";

wait for 50 ns;
MB_Select <= '0';
MD_Select <= '1';
FS <= "10100";

wait for 50 ns;
MB_Select <= '0';
MD_Select <='1';
FS <= "11000";
```

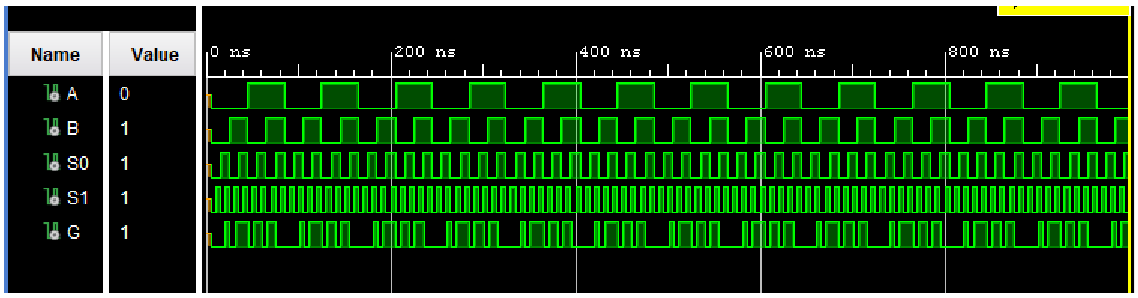
```
wait for 50 ns;
MB_Select <= '1';
MD_Select <='0';
FS <= "10000";

wait for 50 ns;
MB_Select <= '1';
MD_Select <= '0';
FS <= "10100";

wait for 50 ns;
MB_Select <= '1';
MD_Select <='0';
FS <= "11000";



end process;
end Behavioral;
```

3 Testbench Results

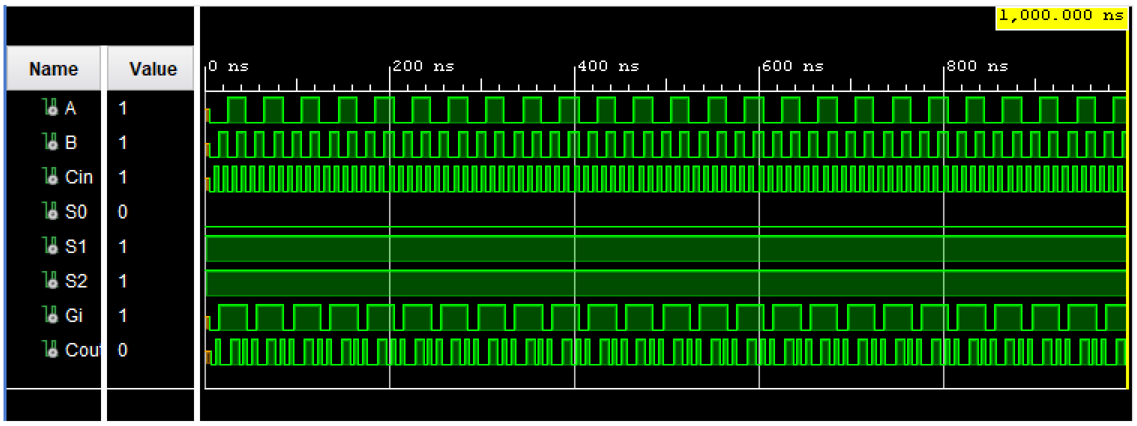
3.1 Bit Slice Arithmetic Unit



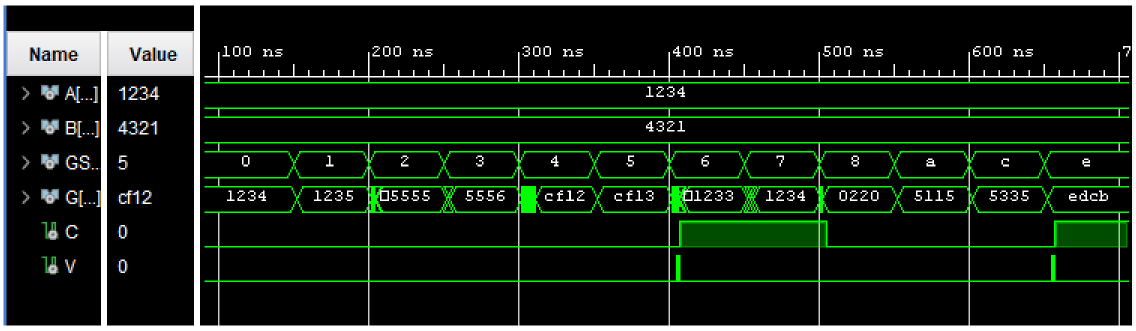
3.2 Bit Slice Logic Unit



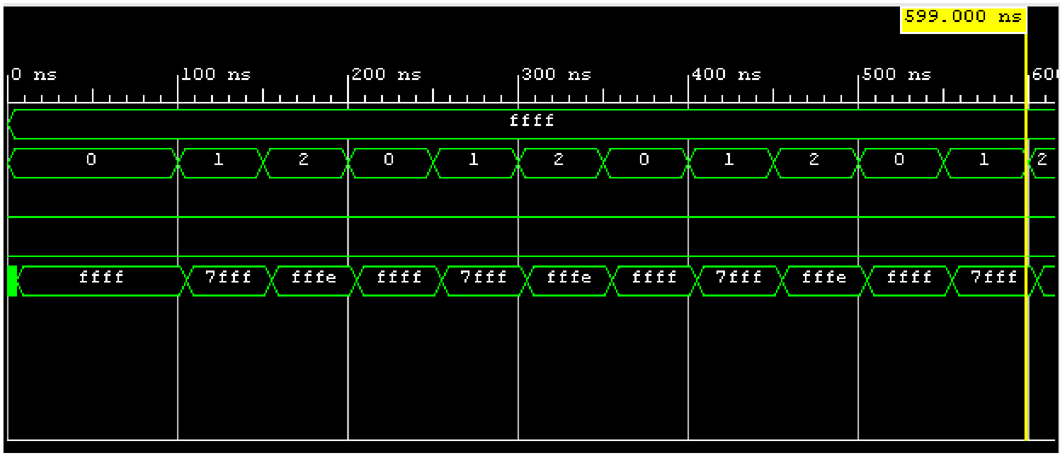
3.3 Bit Slice ALU



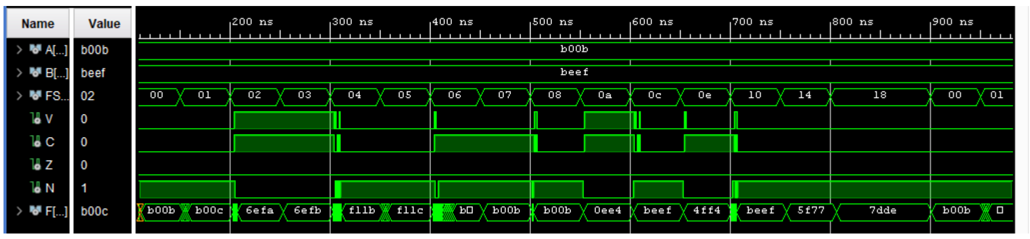
3.4 ALU



3.5 Shifter



3.6 Function Unit


Name
Value
A[...] b00b
B[...] beef
FS... 02
V 0
C 0
Z 0
N 1
F[...] b00c
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
b00b
beef
00 01 02 03 04 05 06 07 08 0a 0c 0e 10 14 18 00 01
b00b b00c 6efa 6efb f11b f11c b b00b b00b 0ee4 beef 4ff4 beef 5f77 7dde b00b

In regards to the ability of the function unit to perform all the micro-operations defined in the table in the assignment it performs them all well. The gate propagation on changing the F-Select does little to alter the result. The logic tables of the two components that compose the function unit are both sound and well defined, so it must follow that the larger function unit accomplishes those tasks given it is connected properly.

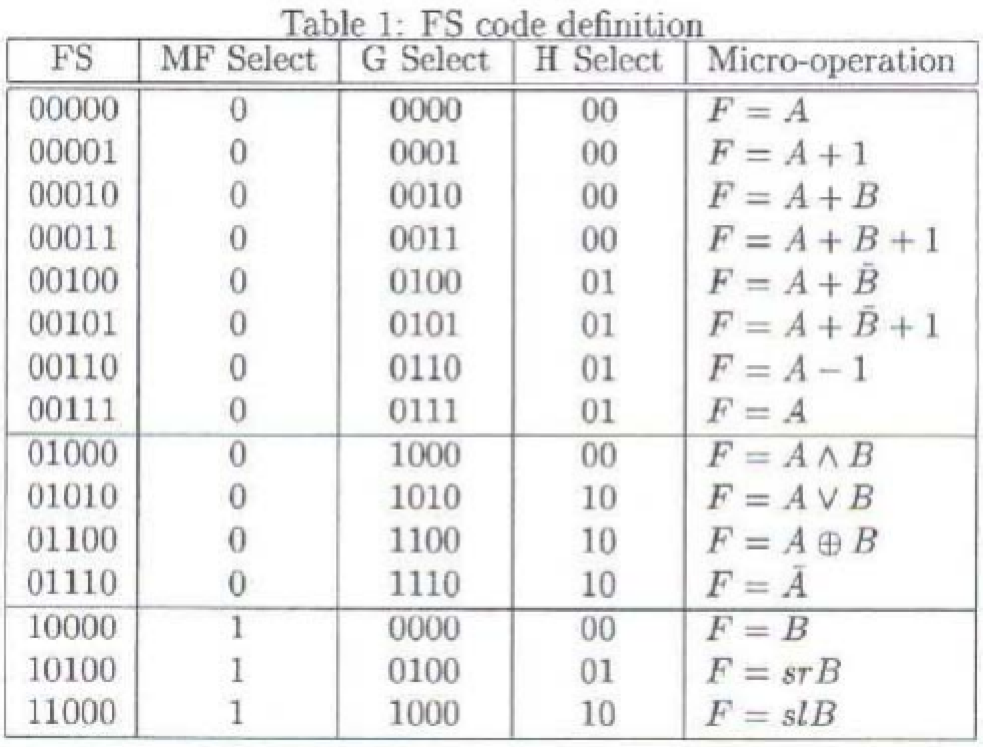
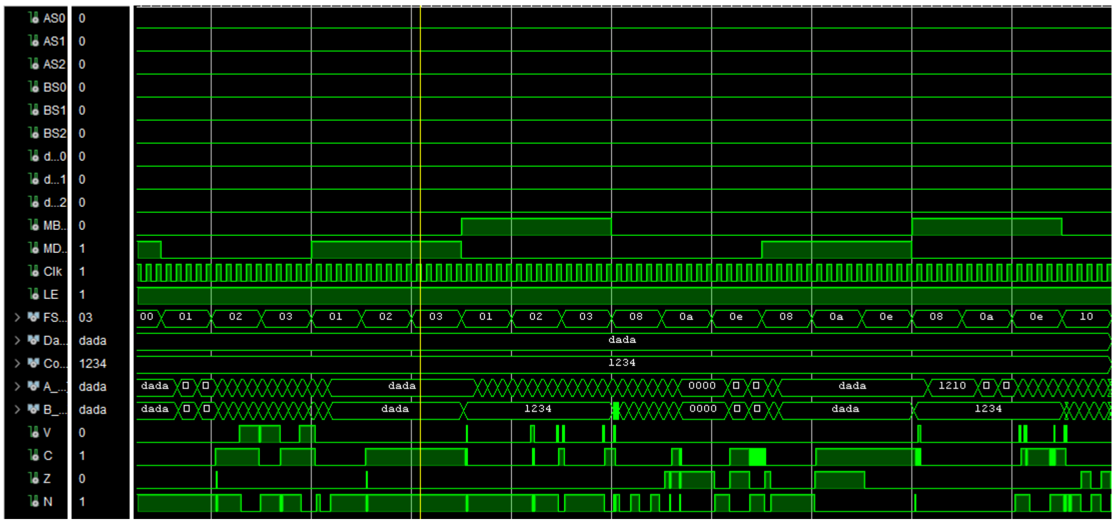


Table 1: FS code definition

FS	MF Select	G Select	H Select	Micro-operation
00000	0	0000	00	$F = A$
00001	0	0001	00	$F = A + 1$
00010	0	0010	00	$F = A + B$
00011	0	0011	00	$F = A + B + 1$
00100	0	0100	01	$F = A + \bar{B}$
00101	0	0101	01	$F = A + \bar{B} + 1$
00110	0	0110	01	$F = A - 1$
00111	0	0111	01	$F = A$
01000	0	1000	00	$F = A \wedge B$
01010	0	1010	10	$F = A \vee B$
01100	0	1100	10	$F = A \oplus B$
01110	0	1110	10	$F = \bar{A}$
10000	1	0000	00	$F = B$
10100	1	0100	01	$F = srB$
11000	1	1000	10	$F = slB$

3.7 Datapath


AS0 0
AS1 0
AS2 0
BS0 0
BS1 0
BS2 0
d...0 0
d...1 0
d...2 0
MB.. 0
MD.. 1
Clk 1
LE 1
FS... 03
Da... dada
Co... 1234
A_... dada
B_... dada
V 0
C 1
Z 0
N 1
00 01 02 03 01 02 03 01 02 03 08 0a 0e 08 0a 0e 08 0a 0e 10
dada
1234
dada 0000 dada 1210
dada 1234 0000 dada 1234