EASTERN MEDITERRANEAN UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT
CMPE224 **DIGITAL LOGIC SYSTEMS**

# VHDL EXPERIMENT VII

**TITLE**: VHDL IMPLEMENTATION OF ALGORITHMIC STATE MACHINES

**OBJECTIVES**: VHDL implementation of digital systems described using ASM charts will be studied. The students are expected to learn the implementation issues of ASM charts.

Below, you will find detailed examples on the implementation of different ASM charts and associated digital systems in VHDL. Examine these VHDL codes and implement them in Quartus II environment to understand the underlying ideas better.

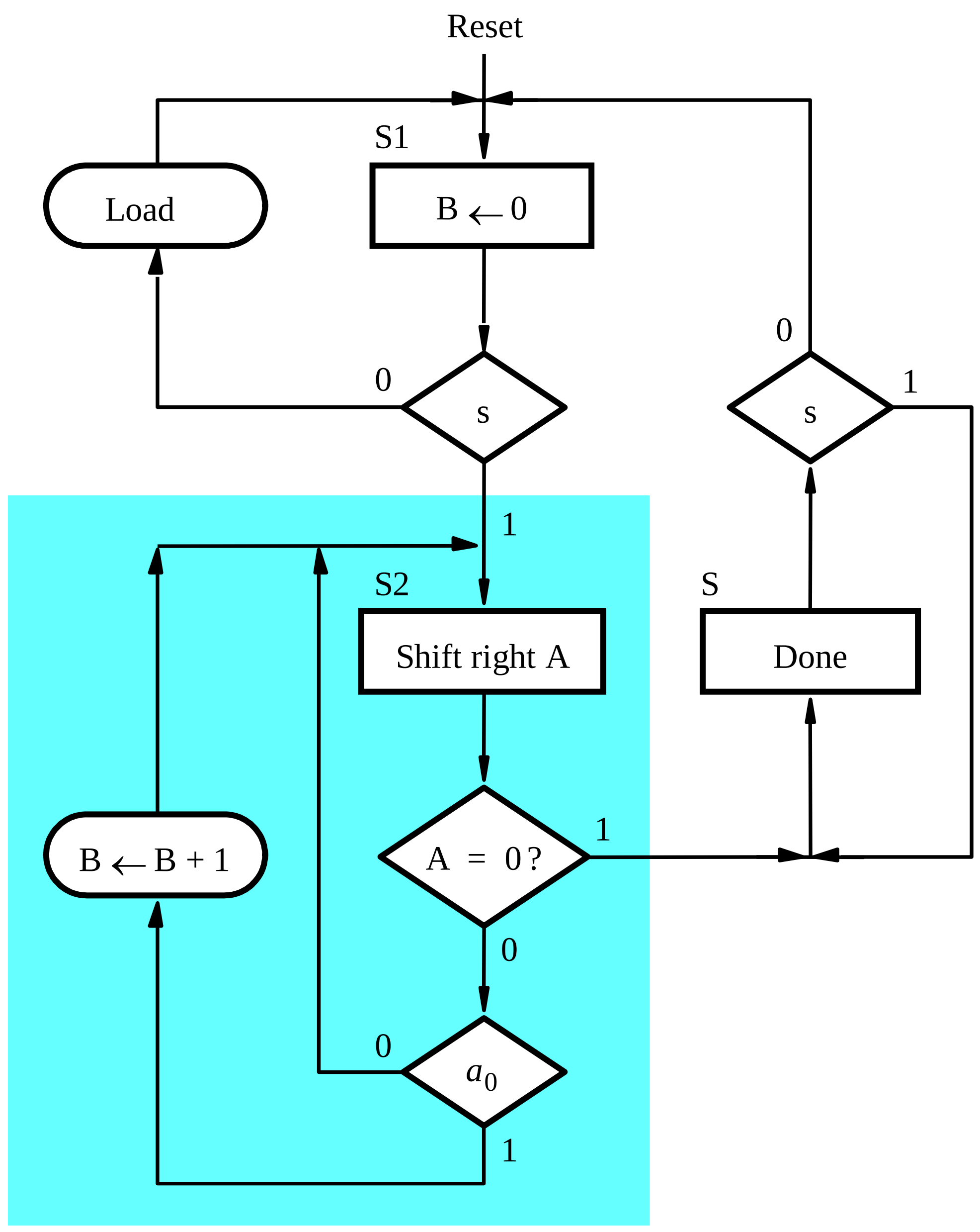
**Example 1:** *Bit-Counting Circuit.*

Pseudo-code of counting the number of 1's within the contents of a register A and storing the count in counter B is as follows:

```
B=0;
while A≠0 do
    if $a_0$=1 then
        B=B+1;
    end if;
    Right-shift A;
end while;
```

ASM chart for this pseudo-code is given below:

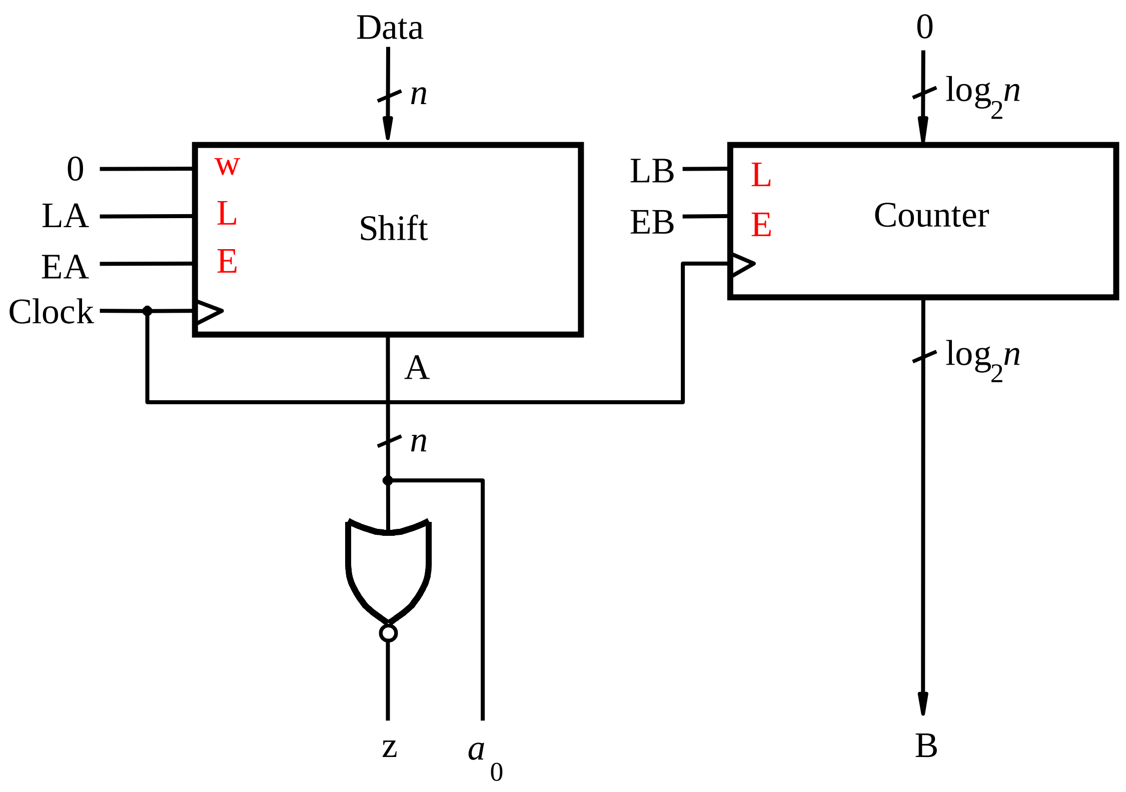
The state box for the starting state S1 specifies that B is initialized to 0. There is an input signal **s** used to indicate when the data to be processed has been loaded into A, so that the machine can start. When **s** becomes one the machine changes to state S2.The decision box below state S2 checks whether A=0. If so, the bit counting operation is complete and the machine should change to state S3. If not, the ASM remains in S2. In state S3, counter B contains the result which is the number of 1's in A. An output signal **Done** is set to 1 to indicate that the algorithm is finished. The FSM stays in S3 until **s** goes back to 0.



ASM chart for the bit-counting circuit.

**Data-path circuit**

The datapath circuit for the bit-counting system described above consists of an n-bit shift register and a $\lceil \log_2 n \rceil$-bit up-counter.



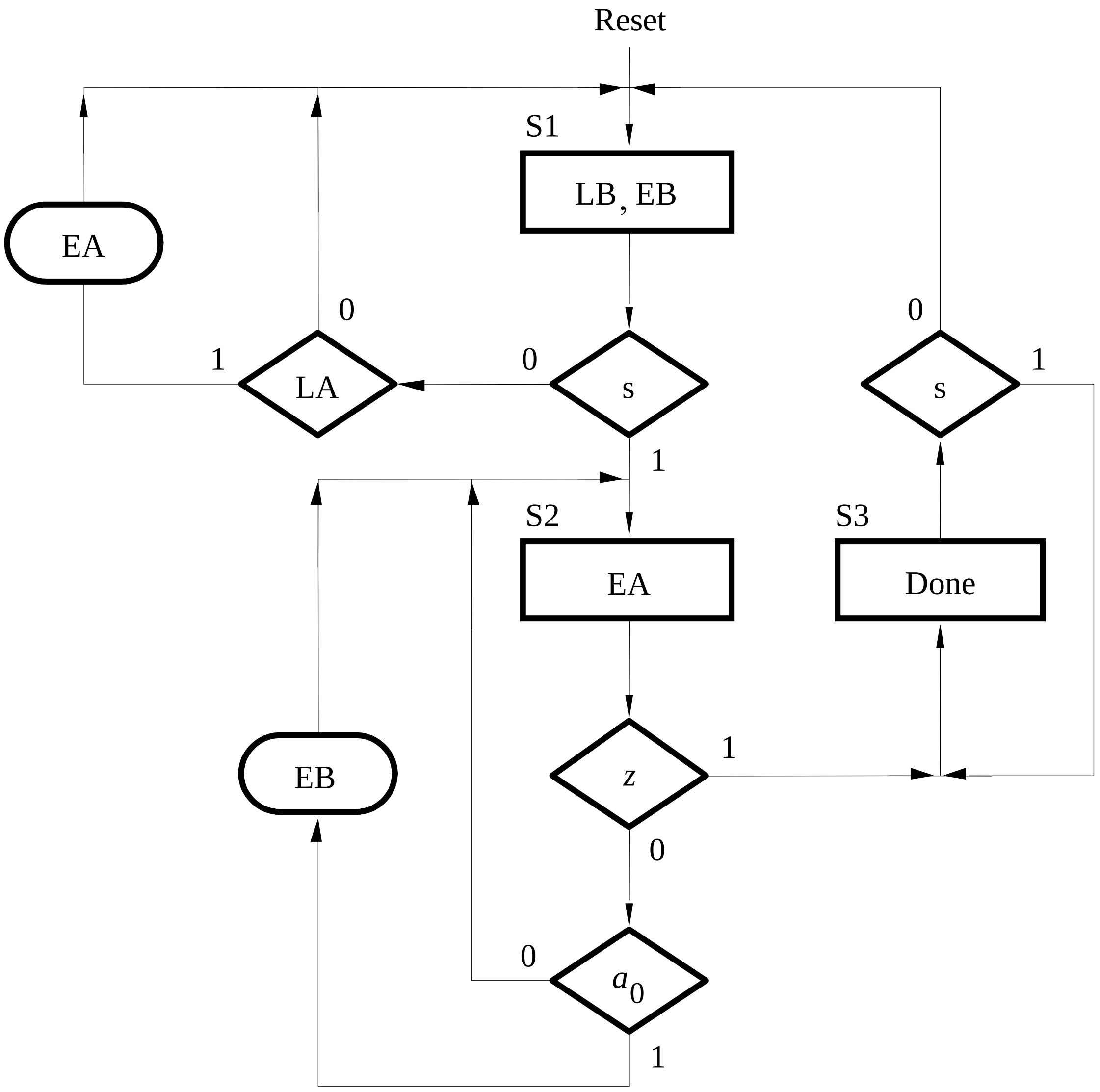
Datapath circuit for ASM of bit-counting system.

In the above datapath circuit LA and EA represents the 'Load A' and 'Enable A' signals. The parallel input to register A is named *Data*, and its parallel aoutput is A. An n-input NOR gate is used to test whether A=0. The output of this gate is 1 when A=0; i.e., z=1 when A=0. The counter has a $\lceil \log_2 n \rceil$ bits, with parallel inputs connected to 0. The counter has a parallel load input LB and an enable input EB.

**Control Circuit**

The ASM needed for the control circuit is given below. This ASM illustrates how the inputs LA, s, $a_0$, and z are processed and how the outputs EA, LB, EB, and *Done* are generated.

The VHDL code implementing the datapath and the control circuit of the bit-counting system is also presented below.



ASM needed for the control circuit of  bit-counting system

**VHDL Code for the BIT COUNTING example**

```
LIBRARY ieee; USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all ;
ENTITY ExpVII IS PORT(
            Clock : IN STD_LOGIC;
            Load_A,s : IN STD_LOGIC;
            InputStream : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
            B : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
            Done : OUT STD_LOGIC);
END ExpVII;

ARCHITECTURE Behavior OF ExpVII IS
SIGNAL STATE : STD_LOGIC_VECTOR (1 DOWNTO 0);
SIGNAL A : STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL BB : STD_LOGIC_VECTOR (3 DOWNTO 0);
SIGNAL z, EA, EB : STD_LOGIC;

BEGIN
State_Transitions:
    PROCESS (Clock)
    BEGIN

        IF (Clock'EVENT AND Clock='1')
        THEN
            if STATE = "00" then
                if s = '0' then
                    STATE <= "00";
                else
                    STATE <= "01";
                end if;
            elsif STATE = "01" then
                if z = '0' then
                    STATE <= "01";
                else
                    STATE <= "11";
                end if;
            elsif STATE = "11" then
                if s = '1' then
                    STATE <= "11";
                else
                    STATE <= "00";
                end if;
            end if;
        END IF;
    END PROCESS;

Control_Outputs:
    PROCESS (STATE,s,A(0))
    BEGIN
        Done<='0';

        IF STATE="00" THEN
            EA<='0';
            EB<='0';

        ELSIF STATE="01" THEN
            EA<='1';
            IF A(0)='1' THEN
                EB<='1';
            ELSE
```

```
                EB<='0';
            END IF;
        ELSIF STATE="11" THEN
            EA <= '0';
            EB <= '0';
            Done<='1';
        END IF;
    END PROCESS;

Datapath:
    PROCESS (Clock)
    BEGIN
        IF (Clock'EVENT AND Clock='1') THEN
            IF STATE="00" THEN
                BB<="0000";
            ELSE
                if EB = '1'
                    then BB<=BB+1;
                end if;
            END IF;
        END IF;
    END PROCESS;

ShiftA:
    PROCESS(Clock,InputStream,Load_A,s,Clock)
    BEGIN
        IF (Clock'EVENT AND Clock='1') THEN
            IF (Load_A='1') THEN
                A<=InputStream;
            ELSE
                IF EA='1' THEN
                    A<='0'&A(7 DOWNTO 1);
                    IF A="00000000" THEN
                        z<='1';
                    ELSE
                        z<='0';
                    END IF;
                END IF;
            END IF;
        END IF;
    END PROCESS;

B<=BB;
END Behavior;
```

## PRELIMINARY WORK:

1. Write down the VHDL code for the bit-counting circuit described above. Perform all experimental steps including VHDL coding, waveform preparation, and simulations; make them ready before you come to the laboratory.

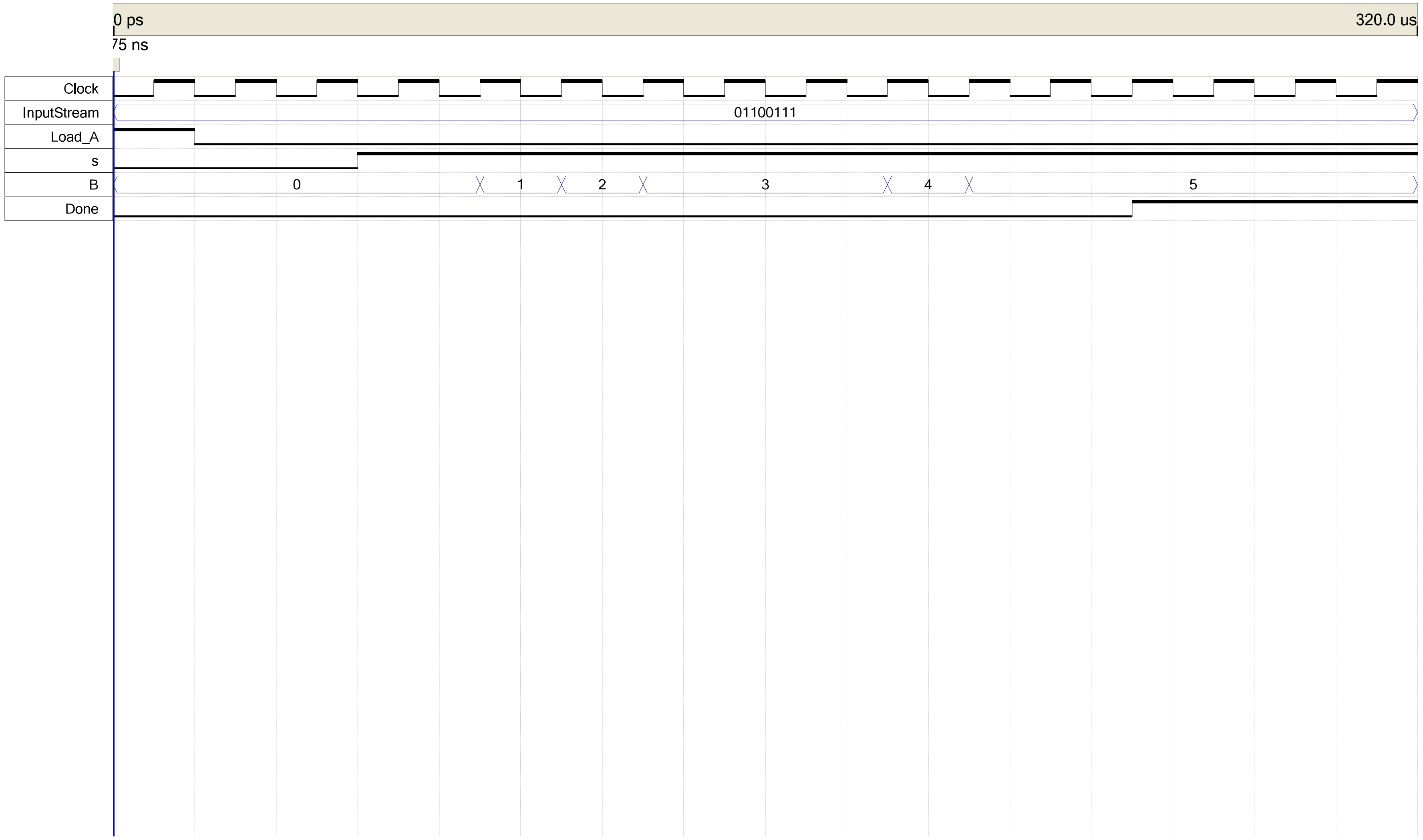
2. Make slight modifications in the above VHDL code to count the number of 0's in A. Perform all experimental steps for this implementation also.

## EXPERIMENTAL WORK:

Demonstrate your preliminary work to the laboratory assistants using Quartus II environment. Be ready for detailed questions on your work.

Good Luck.

Dr. Adnan Acan
Dr. Evgueni Doukhnitch
Dr. Muhammed Salamah.


0 ps
320.0 us
75 ns
Clock
InputStream
Load_A
s
B
Done
01100111
0
1
2
3
4
5