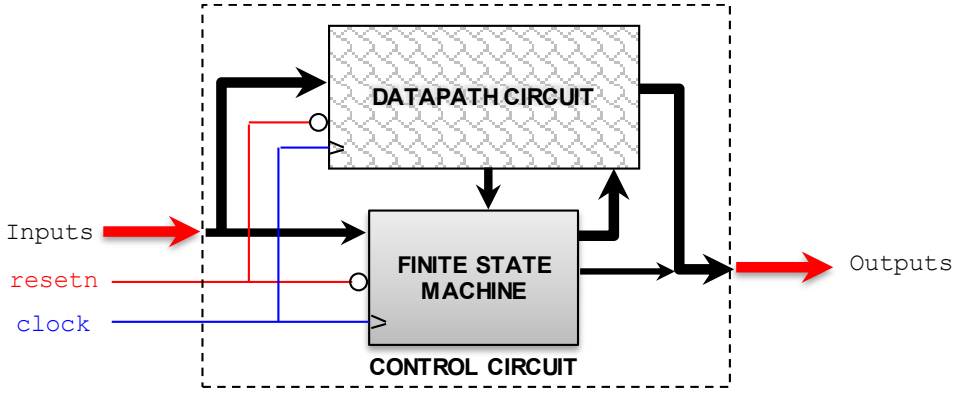
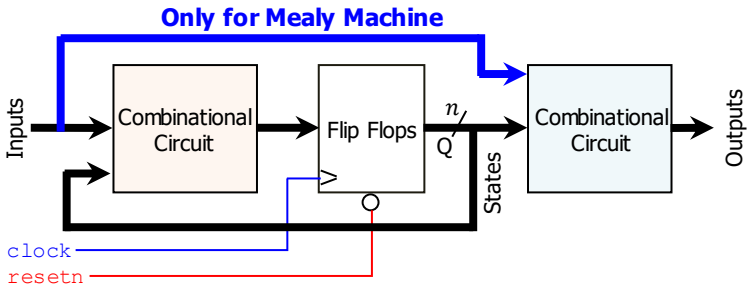
Unit 2 - Digital System Design

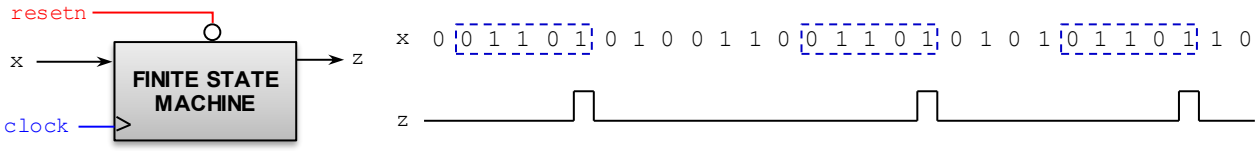
COMPONENTS: FSM (CONTROL) + DATAPATH CIRCUIT



FINITE STATE MACHINES (FSMS)

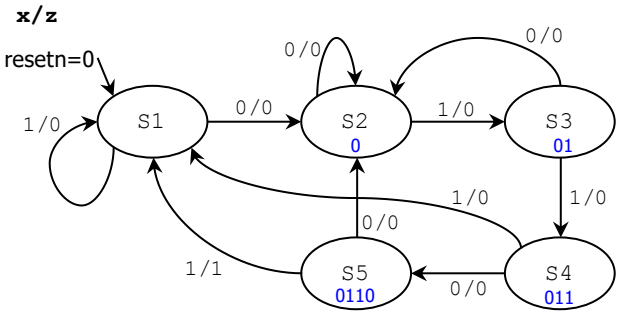
- For a review of this topic in detail, please refer to ECE2700 – Synchronous Sequential Circuits.
- State Machine representation: we will use the Algorithmic State Machine (ASM) Charts.
- **Example:** Sequence detector. It generates z=1 when it detects the sequence 01101. Once the sequence is detected, it looks for a new sequence.

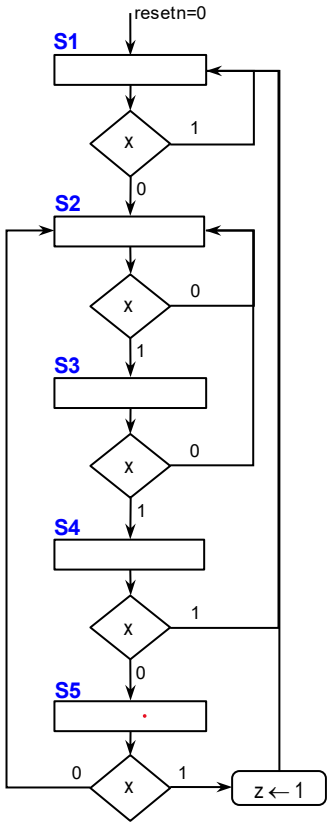




- ✓ Finite State Machine: This is a Mealy FSM.

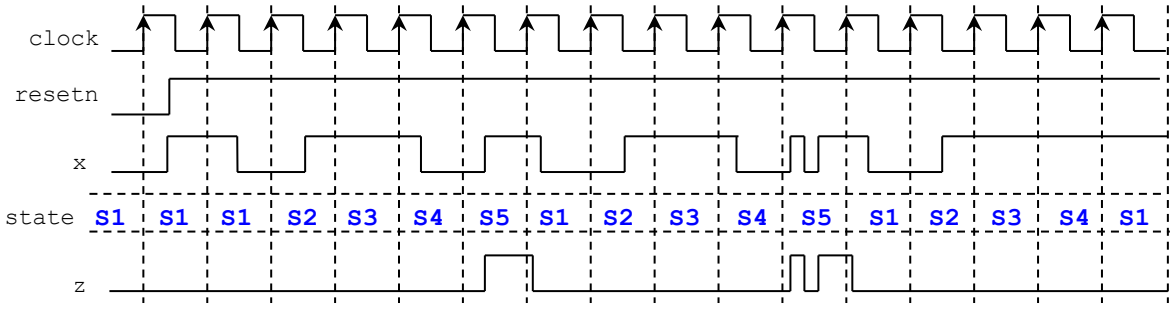
  2 representations:
  - Bubble Form
  - ASM Chart





- ✓ Video: convert bubble form into ASM (also: VHDL coding of ASM).

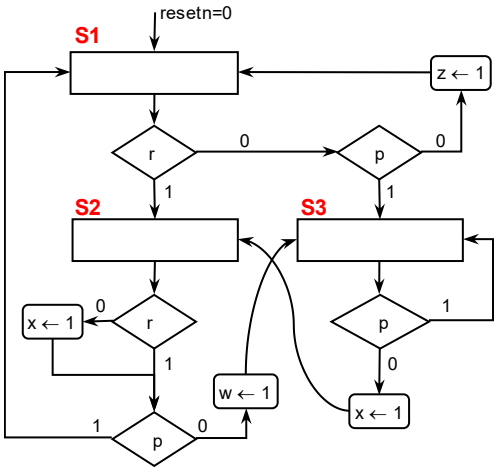
✓ Timing diagram: Sequence detector – 01101.



Note: The output z can change as soon as the input x changes.

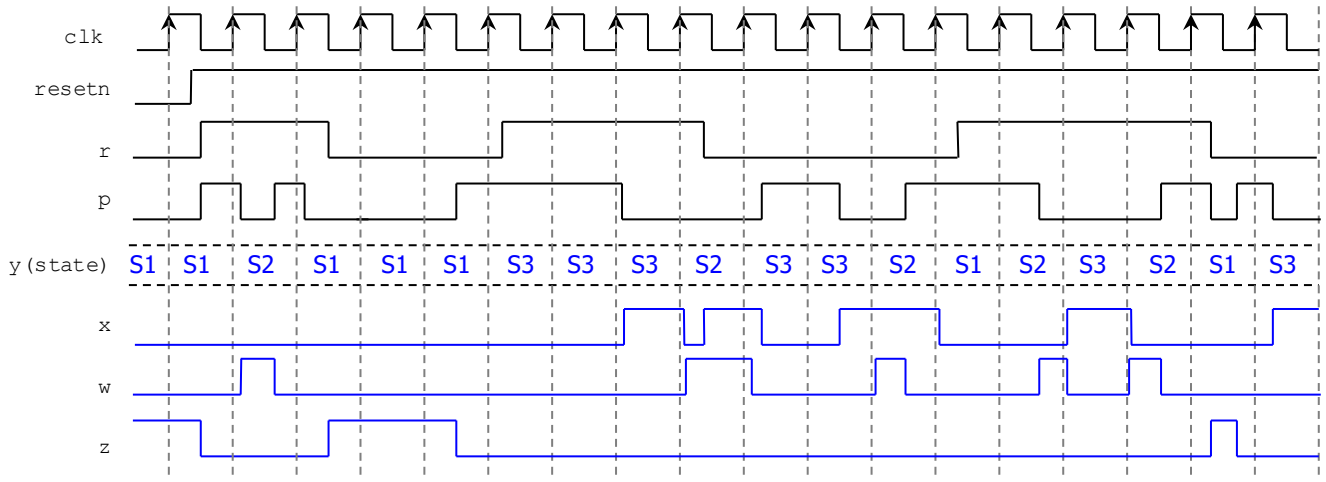
- **Example**: This is a more complex FSM (Mealy-type). Given the VHDL code, we complete the FSM diagram, the timing diagram and provide the state table, excitation table, excitation table, and circuit implementation.

```
library ieee;
use ieee.std_logic_1164.all;

entity myfsm is
   port ( clk, resetn: in std_logic;
          r, p: in std_logic;
          x, w, z: out std_logic);
end myfsm;
```




```
architecture behavioral of myfsm is
   type state is (S1, S2, S3);
   signal y: state;
begin
  Transitions: process (resetn, clk, r, p)
  begin
     if resetn = '0' then y <= S1;
     elsif (clk'event and clk = '1') then
        case y is
          when S1 =>
            if r = '1' then y <= S2;
            else
               if p = '1' then y <= S3; else y <= S1; end if;
            end if;
          when S2 =>
            if p = '1' then y <= S1; else y <= S3; end if;
          when S3 =>
            if p = '1' then y <= S3; else y <= S2; end if;
        end case;
     end if;
  end process;

  Outputs: process (y, r, p)
  begin
      x <= '0'; w <= '0'; z <= '0';
      case y is
         when S1 => if r = '0' then
                       if p = '0' then z <= '1'; end if;
                    end if;
         when S2 => if r = '0' then x <= '1'; end if;
                    if p = '0' then w <= '1'; end if;
         when S3 => if p = '0' then x <= '1'; end if;
      end case;
  end process;
end behavioral;
```

✓ Timing Diagram: Note that the outputs x, w, z can change as soon as the input x changes.



✓ State Table and Excitation Table:

State Assignment:
S1: Q=00, S2: Q=01, S3: Q=10

r p	PRESENT STATE	NEXT STATE	x w z
0 0	S1	S1	0 0 1
0 0	S2	S3	1 1 0
0 0	S3	S2	1 0 0
0 1	S1	S3	0 0 0
0 1	S2	S1	1 0 0
0 1	S3	S3	0 0 0
1 0	S1	S2	0 0 0
1 0	S2	S3	0 1 0
1 0	S3	S2	1 0 0
1 1	S1	S2	0 0 0
1 1	S2	S1	0 0 0
1 1	S3	S3	0 0 0

r p $Q_1Q_0(t)$	$Q_1Q_0(t+1)$	x w z
0 0 0 0	0 0	0 0 1
0 0 0 1	1 0	1 1 0
0 0 1 0	0 1	1 0 0
0 0 1 1	X X	X X X
0 1 0 0	1 0	0 0 0
0 1 0 1	0 0	1 0 0
0 1 1 0	1 0	0 0 0
0 1 1 1	X X	X X X
1 0 0 0	0 1	0 0 0
1 0 0 1	1 0	0 1 0
1 0 1 0	0 1	1 0 0
1 0 1 1	X X	X X X
1 1 0 0	0 1	0 0 0
1 1 0 1	0 0	0 0 0
1 1 1 0	1 0	0 0 0
1 1 1 1	X X	X X X

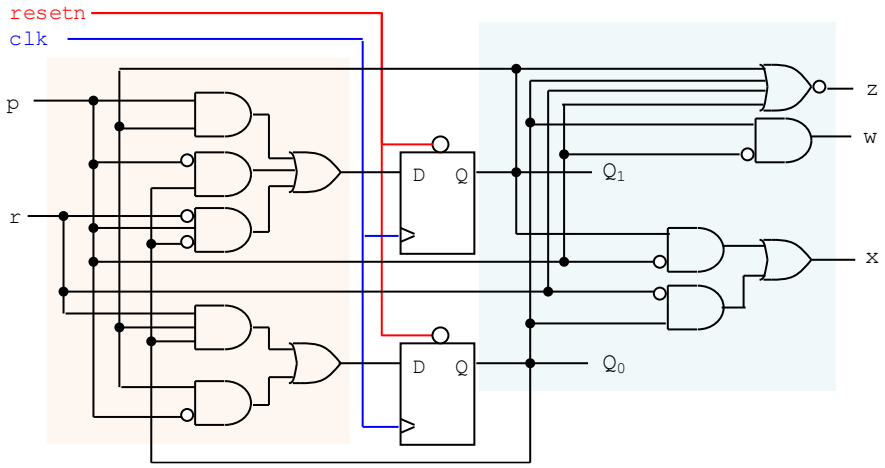
✓ Excitation equations and Boolean output equations:

$Q_1(t+1) \leftarrow \bar{r}p\overline{Q_0(t)} + pQ_1(t) + \bar{p}Q_0(t)$
$Q_0(t+1) \leftarrow r\overline{Q_1(t)}\,\overline{Q_0(t)} + \bar{p}Q_1(t)$
$x = \bar{p}Q_1(t) + \bar{r}Q_0(t) \qquad w = \bar{p}Q_0(t) \qquad z = \bar{r}\bar{p}\overline{Q_1(t)}\,\overline{Q_0(t)}$

✓ Circuit implementation:

Note that we highlight the components of the FSM model.

Also, refer to the VHDL for FPGAs tutorial – Unit 6 on how the VHDL code maps into the circuit components.



DATAPATH CIRCUIT

- The components required here vary widely and depend on your specific application. Typical components found include arithmetic circuits, special encoders/decoders, counters, registers, shift registers, etc.

DIGITAL SYSTEM DESIGN

- Usually, you start with the specifications of your circuit. Then, you define the I/Os. After this, it is up to you to decide on what components to use, how to interconnect them, and how to control it via a FSM.

VHDL CODING

Refer to VHDL for FPGAs Tutorial for a complete tutorial on VHDL design that includes slides and examples (VHDL code). For the description of digital systems in VHDL, the following concepts are of utmost importance:

- **FSM Description**: VHDL coding of ASMs:
  - ✓ VHDL on FPGA Tutorial – Unit 6 slides: How to write VHDL code for FSMs.
  - ✓ Video: VHDL description for ASM chart (sequence detector 01101), synthesis and simulation in Vivado.

- **Interconnection of components**: hierarchical design (port map): VHDL for FPGA Tutorial – Unit 4 slides.

- **Testbench generation**: Refer to VHDL for FPGAs Tutorial – Unit 5 slides for detailed explanation. Also, look at the examples in Unit 6 and Unit 7 of this VHDL for FPGAs Tutorial.

- **Use of generic components**: These components are ubiquitous in digital system design: counters, registers, shift registers. To optimize design time, we recommend using parameterized components (VHDL for FPGAs Tutorial – Unit 5 examples):
  - ✓ n-bit register with enable and synchronous clear: my_rege
  - ✓ Counter modulo-N with enable and synchronous clear: my_genpulse_sclr
  - ✓ n-bit parallel access (right/left) register with enable and synchronous clear: my_pashiftreg_sclr

Usually, we want to use these components with different variations: bit-width, direction (shift registers), modulus (counters). To properly use parametric components, refer to VHDL for FPGAs Tutorial – Unit 7 slides (generic map).

- **Digital System Design**: Refer to the following material:
  - ✓ VHDL for FPGAs Tutorial – Unit 7 slides: Detailed step-by-step example (stopwatch design).
  - ✓ VHDL for FPGAs Tutorial – Unit 7: VHDL code for a variety of digital systems (e.g.: bit-counting, sequential multiplier).
  - ✓ Video: Bit-counting circuit ($n = 8$): step-by-step VHDL coding, synthesis and simulation in Vivado.
  - ✓ Video: 4x4 Sequential Multiplier: step-by-step VHDL coding, synthesis and simulation in Vivado.
  - ✓ Video: 2x2 Sequential Multiplier: VHDL coding, synthesis, simulation, implementation, and testing on ZYBO Board.

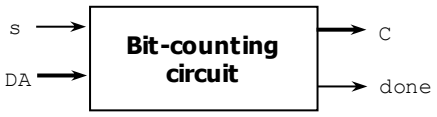
DESIGN EXAMPLES

BIT-COUNTING CIRCUIT (COUNTING 1'S) (Parametric VHDL Code) (Video: VHDL coding in Vivado, n=8)

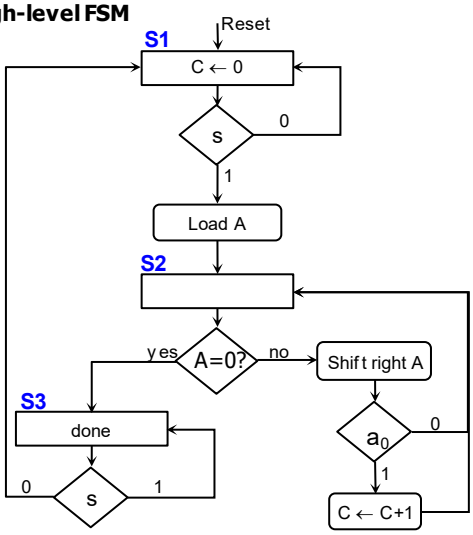
- This circuit counts the number of bits in register A whose value is '1'. Example (n=8): A = 00110111 → C = 0101.
- The sequential algorithm (pseudo-code) is shown. Here, we can follow this procedure to design a digital system:
  - ✓ Sketch the block diagram: We need start and done signals to indicate when the process starts and finishes. We also include input (Data for the Register A) and output data (Count).
  - ✓ Sketch the high-level control mechanism (state machine) for the Bit-counting circuit. Here, you can include combinational blocks as well as common synchronous blocks (registers, shift registers, counters).
  - ✓ With the block diagram and high-level state machine, we can draft the components and their signals in the datapath. From the high-level state machine, we can design the actual FSM that includes actual signals controlling the components.

Sequential Algorithm

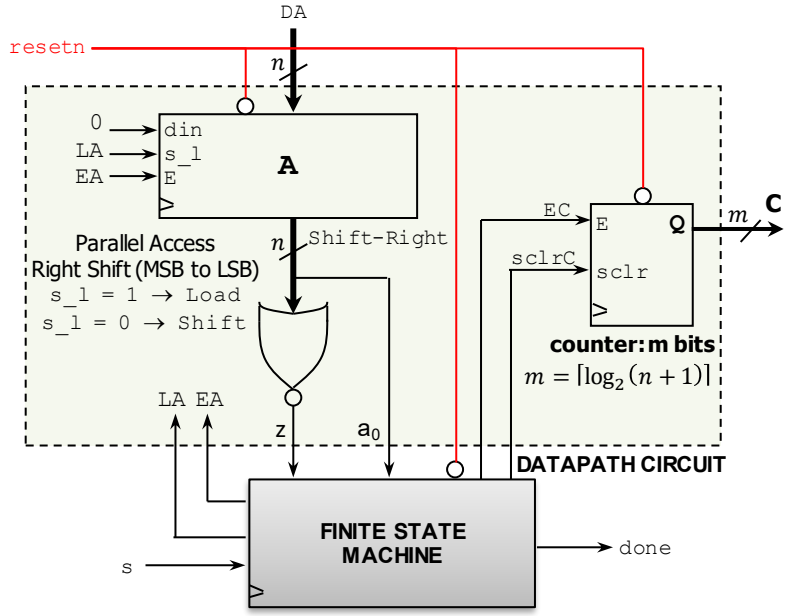
```
C ← 0
while A ≠ 0
   if a0 = 1 then
      C ← C + 1
   end if
   right shift A
end while
```

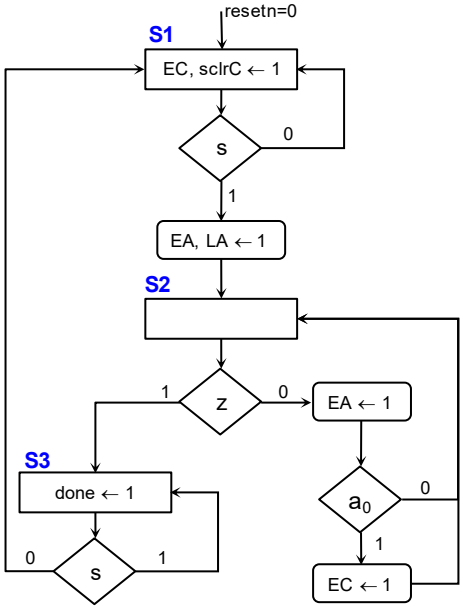


High-level FSM



DIGITAL SYSTEM (FSM + Datapath circuit)





- Components: Behavior on the clock tick.

m-bit counter (modulo-n+1): If E=0, the count stays.	n-bit Parallel access shift register: If E=0, the output is kept.

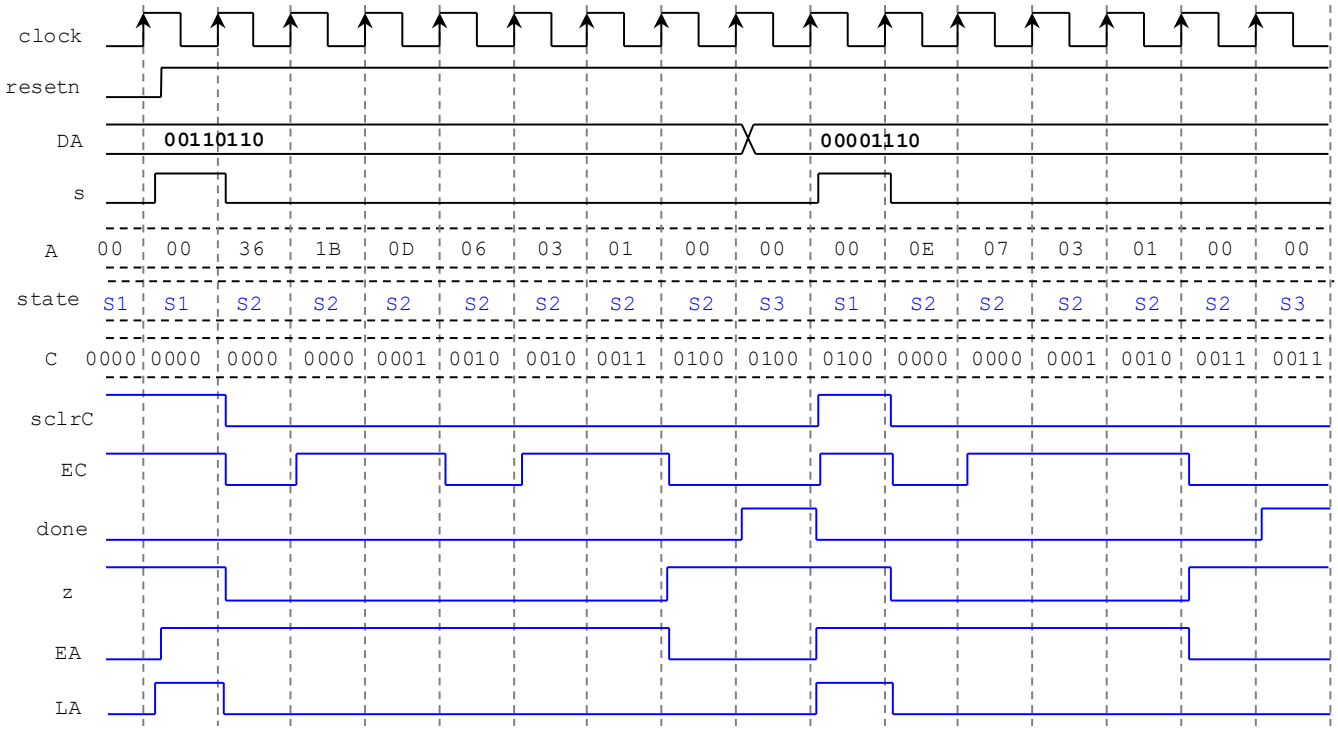
`if E = 1 then`
`  if sclr = 1 then`
`  Q ← 0`
`  else`
`  Q ← Q+1`
`  end if;`
`end if;`
	`if E = 1 then`
	`  if s_l = '1' then`
	`  Q ← D`
	`  else`
	`  Q ← shift in 'din' (to the right)`
	`  end if;`
	`end if;`

- **Timing diagram** (n=8, m=4):

  ✓ Examples shown in figure:
  - If $A = 00110110$, then $C = 0100$.
  - If $A = 00001110$, then $C = 0011$.

  ✓ Video: Bit-counting circuit (n=8): Completing timing diagrams (For DA=10110110 and 00000011)

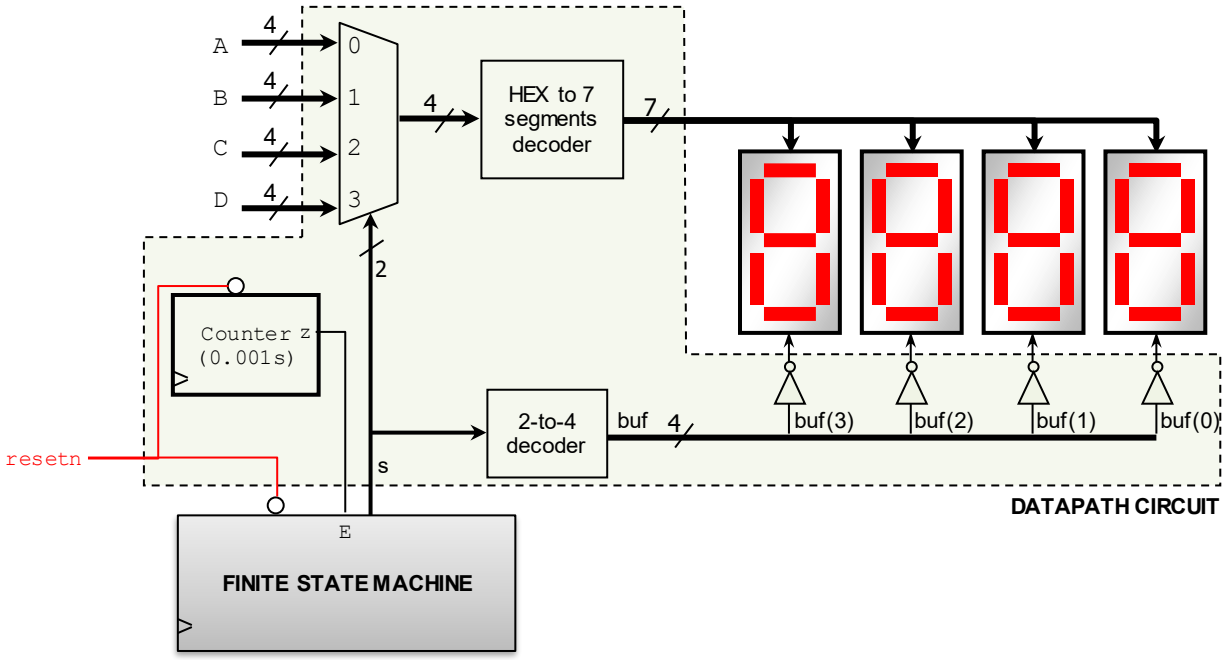
  ✓ To complete timing diagrams for digital systems, the following procedure can be followed:
  For every clock cycle:
  a. Complete the registered signals (i.e.: A, state, C). These signals are kept constant during a clock cycle.
  b. Complete the signals that are not outputs of the FSM, but that are inputs of the FSM (z).
  c. Complete the FSM outputs. Note they might change at any moment during the clock cycle.
  d. For the next clock cycle, go to 'a.'



EXAMPLE: 7-SEGMENT SERIALIZER (VHDL code)

DIGITAL SYSTEM (FSM + Datapath circuit)

- Most FPGA Development boards have a number of 7-segment displays (e.g., 4, 8). However, only one can be used at a time.
- If we want to display four digits (inputs A, B, C, D), we can design a serializer that will only show one digit at a time on the 7-segment displays.
- Since only one 7-segment display can be used at a time, we need to serialize the four BCD outputs. In order for each digit to appear bright and continuously illuminated, each digit is illuminated for 1 ms every 4 ms (i.e. a digit is un-illuminated for 3 ms and illuminated for 1 ms). This is taken care of by feeding the output z of the 'counter to 0.001s' to the enable input of the FSM. This way, state transitions only occur each 0.001 s.
- In the figure, the enable signals for the four 7-segment displays are active low (this is usually the case).

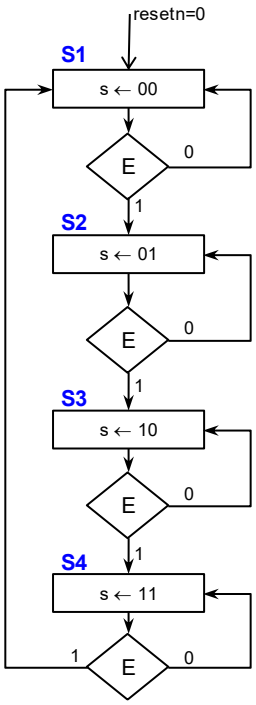


- Component: Behavior on the clock tick.

0.001 s counter (modulo-10^5): Free running counter

```
if Q = 10^5 - 1 then
      Q ← 0
   else
      Q ← Q+1
  end if;
end if;

* z = 1 if Q = 10^5-1
```

- **Algorithmic State Machine (ASM) chart:** This is a Moore-type FSM.



EXAMPLE: SEQUENTIAL MULTIPLIER (Parametric VHDL Code) (Video: VHDL coding in Vivado, n=4)

UNSIGNED MULTIPLICATION: SEQUENTIAL ALGORITHM

```
P ← 0, Load A,B
while B ≠ 0
   if b0 = 1 then
      P ← P + A
   end if
   left shift A
   right shift B
end while
```

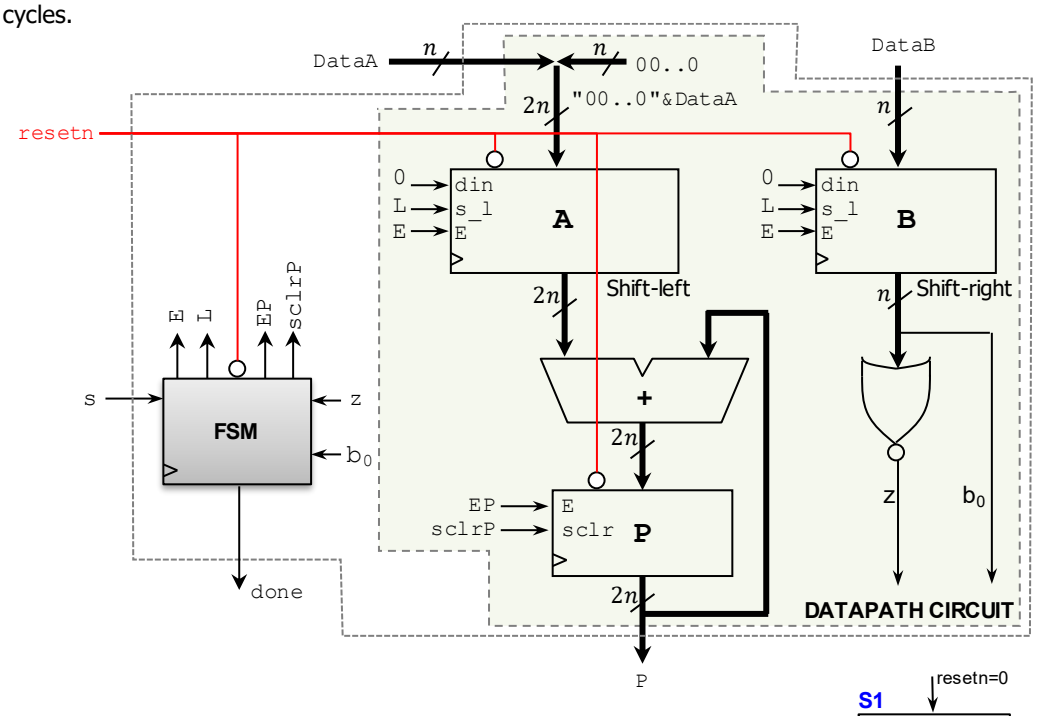
Example:

```
          1 1 1 1 x
          1 1 0 1
          ---------
          1 1 1 1   ⟶ P ← 0 + 1111
        0 0 0 0     ⟶ P ← 1111
      1 1 1 1       ⟶ P ← 1111 + 111100 = 1001011
    1 1 1 1         ⟶ P ← 1001011 + 1111000 = 11000011
  -----------------
  1 1 0 0 0 0 1 1
```

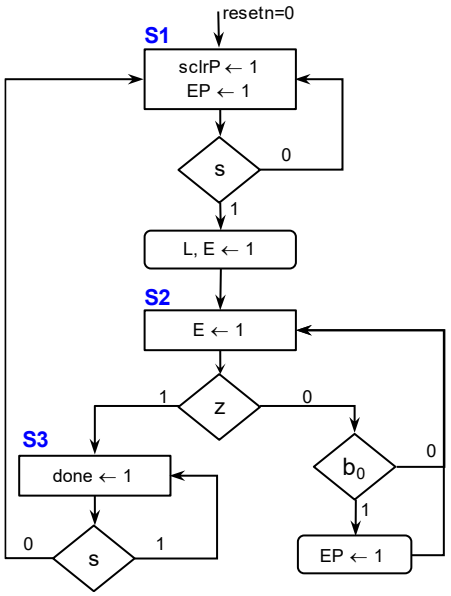
P ← 0, A ← 1111, B ← 1101
b_0=1 ⇒ P ← P + A = 1111. A ← 11110, B ← 110
b_0=0 ⇒ P ← P = 1111. A ← 111100, B ← 11
b_0=1 ⇒ P ← P + A = 1111 + 111100 = 1001011. A ← 1111000, B ← 1
b_0=1 ⇒ P ← P + A = 1001011 + 1111000 = **11000011**. A ← 11110000, B ← 0

DIGITAL SYSTEM (FSM + Datapath circuit)

- Iterative Multiplier Architecture. Register P: $sclr$: synchronous clear. Here, if $E = sclr = 1$, the register contents are initialized to 0. Parallel Access Shift Registers A and B: If $E = 1$: $s\_l = 1 \rightarrow$ Load, $s\_l = 0 \rightarrow$ Shift. The result is computed in at most $n + 1$ cycles.



- **Algorithmic State Machine (ASM) chart:**



- Components: Behavior on the clock tick:

$2n$-bit register: If E=0, the output is kept.	Parallel access shift register: If E=0, the output is kept. A ($2n$ bits, left shift), B (n bits, right shift)

if E = 1 then
  if sclr = 1 then
  Q ← 0
  else
  Q ← D
  end if;
end if;
	if E = 1 then
	  if s_l = '1' then
	  Q ← D
	  else
	  Q ← shift in 'din' (to the left (A) or right (B))
	  end if;
	end if;

- **Timing Diagram:** $n = 4$.