FINITE STATE MACHINES (FSM) IN VHDL
State Machine General Diagram

- **Inputs**
- **Next State Logic**
- **Current State Logic**
- **Output Logic**
- **Outputs**

** CLK RST **

<table>
<thead>
<tr>
<th>Style</th>
<th>Clocked Process</th>
<th>State Transition</th>
<th>Output Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style A</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Style B</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Style C</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Style D</td>
<td>○</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

○ Different processes
● Combined processes

C. Sisterna
Spring 03

ECET - CET 486 - 586
Declare an enumerated data type

```-- declare the (state-machine) enumerated type
type FSM_States is (RST, CNT, LOAD, OUT);
```

Declare signals of the enumerated data type

```-- declare signals of FSM_States type
signal current_state, next_state: FSM_States;
```

The only values that current_state and next_state can hold are: RST, CNT, LOAD and OUT
State Machine: Clocked Process

- The clocked process decides when the state machine should change state.

- This process is activated by the state machine’s clock.

- Depending on the present state and the value of the input signals, the state machine can change state at every active clock edge.

- Current state gets the value of the next state on the active edge of the clock.

- Next state value is generated in the state transition process, depending on the values of current state and the inputs.
State Machine: Combinatorial Process

• Assigns the output signals their value depending on the present state

• Next state logic and output logic is best modeled using case statements are better for this process

• All the rules of combinatorial process have to be followed to avoid generating unwanted latches

• For Mealy Machines *if-then-else* statement is used to create the dependency between the current state, the input signal and output signal
State Machine: Reset behavior

- Asynchronous reset: ensure that the state machine is always initialized to a known valid state, before the first active clock transition and normal operation commences.

- No reset or a synchronous reset: there is no way to predict the initial value of the state register flip-flops. It could power up and become permanently stuck in an uncoded state.
State Machine Style Descriptions in VHDL - Example

Moore FSM

- S0: X = 0, Z = 0
- S1: X = 1, Z = 1

Mealy FSM

- X = 1, Z = 1
- X = 0, Z = 0
- X = 1, Z = 0
- X = 0, Z = 1
process (state, X)
begin
    case state is
    when S0 => next_state <=..;
    when S1 => next_State <=..;
    end case;
end process;

process (clk, rst)
begin
    if (rst = '1') then
        state <= S0;
    elsif (rising_edge(clk)) then
        state <= next_state;
    end if;
end process;

process (state)
begin
    case state is
    when S0 => Z <= ...;
    when S1 => Z <= ...;
    end case;
end process;
Style A - Mealy State Machine

Next State Logic

process (state, X)
begin
  case state is
    when S0 => next_state <=...
    when S1 => next_state <=...
  end case;
end process;

Present State Logic

process (clk, rst)
begin
  if(rst = '1') then
    state <= S0;
  elsif (rising_edge(clk)) then
    state <= next_state;
  end if;
end process;

Output Logic

process (state, X)
begin
  case state is
    when S0 => if (X = '0') then
      Z <= ...
    else
      Z <= ...
    end if;
    when S1 => if (X = '1') then
      ....
    else
      ....
    end if;
  end case;
end process;
Style A - Synthesis

Moore

Mealy
process(clk, rst)
begin
  if(rst = '1') then
    state <= s0;
    Z <= '0';
    elsif (rising_edge(clk)) then
      case state is
      when S0 =>
        next_state <= .... ;
        Z <= ... ;
       when S1 =>
        next_state <= ... ;
        Z <= ... ;
      end case;
    end if;
end process;
Style B - Mealy State Machine

State, Next State and Output Logic

process(clk, rst)
begin
  if(rst = '1') then
    state <= S0;
    Z <= '0';
  elsif (rising_edge(clk)) then
    case state is
      when S0 =>
        next_state <= .... ;
        if (X='0') then
          Z <= ... ;
        else
          Z <= ... ;
        end if;
      when S1 =>
        next_state <= ... ;
        if (X = '1') then
          Z <= ... ;
        end if;
    end case;
  end if;
end process;
Style B - Synthesis

Moore

Mealy

ECET - CET 486 - 586
Style C - Moore State Machine

Present State and Next State Logic

process (clk, rst)
begin
  if(rst = '1') then
    State <= S0;
  elsif (rising_edge(clk)) then
    case state is
      when S0 =>
        if(X = '1') then
          next_state <= ... ;
        end if;
      when S1 =>
        if(X = '1') then
          next_state <= ... ;
        end if;
    end case;
  end if;
end process;

Output Logic

process (state)
begin
  case state is
    when S0 => Z <= ... ;
    when S1 => Z <= .... ;
  end case;
end process;
process (clk, rst)
begin
  if (rst = '1') then
  state <= S0;
  elsif (rising_edge(clk)) then
    case state is
    when S0 =>
      if (X = '1') then
        next_state <= ...
      end if;
    when S1 =>
      if (X = '1') then
        next_state <= ...
      end if;
    end case;
  end if;
end process;

process (state, X)
begin
  case state is
  when S0 =>
    if (X = '1') then
      Z <= ...
    else
      Z <= ...
    end if;
  when S1 =>
    if (X = '1') then
      Z <= ...
    else
      Z <= ...
    end if;
  end case;
end process;
Style D - Moore State Machine

Next State and Output Logic

process (state, X)
begin
    next_state <= state;
    case state is
        when S0 =>
            if (X = '1') then
                next_state <= ...;
            end if;
            Z <= ...;
        when S1 =>
            if (X = '1') then
                next_state <= ...;
            end if;
            Z <= ...;
    end case;
end process;

Present State Logic

process (clk, rst)
begin
    if (rst = '1') then
        state <= S0;
    elsif (rising_edge(clk)) then
        state <= next_state;
    end if;
end process;
Next State and Output Logic

process (state, X)
begin
    next_state <= state;
    Z <= ...;
    case state is
        when S0 =>
            if(X = '1') then
                next_state <= ...;
                Z <= ...;
            end if;
        when S1 =>
            if(X = '1') then
                next_state <= ...;
                Z <= ...;
            end if;
    end case;
end process;

Present State Logic

process (clk, rst)
begin
    if(rst = '1') then
        state <= S0;
        state <= next_state;
    elsif (rising_edge(clk)) then
        state <= next_state;
    end if;
end process;
Style D - Mealy State Machine

Moore

Mealy
There are several ways to encode a State Machine:

- Sequential
- Gray
- Binary
- One-hot
- Two-hot

- Minimal number of FFs and wide combinatorial functions → CPLD
- Each state one FF → FPGA
- Narrow combinatorial functions
Override the default FSM Compiler encoding for a FSM

Possible values:

- **Default**: assign an encoding style based on the number of states:
  - **Sequential** for 0 - 4 enumerated types
  - **One-hot** for 5 - 24 enumerated types
  - **Gray** for > 24 enumerated types
- **Sequential**: 000 001 010 011 100 101 110 111
- **One-hot**: 001 010 100
- **Gray**: 000 001 011 010 110 111 101 100
- **Safe**: default encoding + reset logic to a known state
Syntax (source code)

```vhdl
-- declare the (state-machine) enumerated type
type my_state_type is (SEND, RECEIVE, IGNORE, HOLD, IDLE);
-- declare signals as my_state_type type
signal nxt_state, current_state: my_state_type;
-- set the style encoding
attribute syn_encoding of current_state: signal is value;
```

**syn_encoding in SCOPE**
Possible values:

- Binary
- Onehot
- Twohot
- Gray
- Random

```-- Declare the (state-machine) enumerated type
type my_state_type is (SEND, RECEIVE, IGNORE, HOLD, IDLE);
-- Set the TYPE_ENCODING_STYLE of the state type
attribute TYPE_ENCODING_STYLE of my_state_type:type is ONEHOT;```
**type_encoding** - LeonardoSpectrum

**type_encoding** allows **to fully** control the state encoding hard code the state code in the source code

```vhdl
-- Declare the (state-machine) enumerated type
type my_state_type is (SEND, RECEIVE, IGNORE, HOLD, IDLE);
-- Set the type_encoding attribute
attribute type_encoding of my_state_type: type is
("0001","01--","0000","11--","0010");
```

![State Table](chart)

Note: LeonardoSpectrum allows to use `'-'`. It can be used to reduce the size of the circuit.
State Machine Coding: Residual States

- If one-hot state coding is not used, the maximum number of states is equal to $2^{**N}$, where $N$ is the vector length of the state vector.

- In state machines where not all the states are used, there are three options:
  - Let “chance: decide what is to happen if the machine goes to an undefined state
  - Define what is to happen if the state machine goes to an undefined state by ending the case statement with `when others`
  - Define all the possible states in the VHDL code