Loops - I

• Used to perform operations many times
  – See previous Parallel to Serial Example
• Several advantages
  – Coding style (easier to read)
  – Laziness!
  – When used correctly can generate better results at synthesis
• General form:

    for <variable> in <range> loop
    <statements>;
    end loop;
Loops- II

• Range can be in the form
  – *upper downto lower* (e.g. 5 *downto* 0)
  – *lower to upper* (e.g. 0 *to* 5)

• Conventions typically use *upper downto lower*
  – Though no difference for synthesis

• Example

```vhdl
for i in 5 downto 0 loop
  dout (i) <= '0';
end loop;
```
Loops- III

• Attributes can also be used for range
  – Convenient, also independent
• Useful predefined attributes (there are others)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>T'high</td>
<td>Greatest Value (e.g. MSB)</td>
</tr>
<tr>
<td>T'low</td>
<td>Least Value (e.g. LSB)</td>
</tr>
<tr>
<td>T'range</td>
<td>Range of values</td>
</tr>
</tbody>
</table>

• Examples:
  ```vhdl```
  ```
  for i in dout'high downto dout'low loop
  for i in dout'range loop
  ```
Loops- IV

- Some words of caution..
- Consider the following example of a (even) parity generator circuit
- Circuit A coded using for loop
  - for i in 2 downto 0..
- Circuit B manually coded
  - z <= () xor () xor ()
- Note the different maximum propagation delays
- Use loops with caution!
State Machines- I

- VHDL offers many different ways to describe state machines
- Methods shown here are only one way
  - Can vary from synthesis tool to synthesis tool
  - Generally, synthesis tool vendor provides guidelines
- Basically 2 forms of state machine- Moore and Mealy
- All machines have
  - Inputs (X)
  - Outputs (Z)
  - State Variables (S)
State Machines - II

- **Mealy machines**
  - $Z = H(X, S_n)$
  - i.e. outputs depend on present state and system inputs
  - Next state: $S_{n+1} = G(X, S_n)$

- **Moore machines**
  - $Z = H(S_n)$
  - i.e. outputs depend on present state only
  - Next state: $S_{n+1} = G(X, S_n)$

- Following synchronous design methodology, outputs and state only ever change on clock edge
State Machines- III

• 1st example- simple state machine
  – System simply proceeds through states- no outputs

• Program listing in sheet ‘SSMACH.VHD’
State Machines- IV

• So, how does the VHDL proceed?
• Firstly, deal with the state variables

```vhdl
type STATES is (START, OPENED, CLOSED);

signal PRESENT_STATE: STATES;
```

Defines a ‘type’ called ‘states’ which contains all possible states

Defines internal signal for the present state
State Machines - V

- Now the main process

```vhd
process (CLK,RESET)
begin
  if RESET='1' then
    PRESENT_STATE <= START;
  elsif CLK'event and CLK='1' then
    case PRESENT_STATE is
      when START =>
        if INPUT='1' then
          PRESENT_STATE <= OPENED;
        else
          PRESENT_STATE <= START;
        end if;
      when OPENED =>
        PRESENT_STATE <= CLOSED;
      ...  
      when others =>
        PRESENT_STATE <= START;
    end case;
  end if;
end process;
```

Ensures that State Machine is Re-entrant
State Machines- VI

• Simulation output:
State Machines - VII

- More realistic system
- States same - but outputs in each of the states

- Program listing in sheet ‘SSMACH2.VHD’
State Machines - VIIb

- Mealy machine notation for the same result.
- Are you comfortable that this is identical?

- Mealy with outputs independent of inputs is a Moore.
process (CLK,RESET)
begin
  if RESET='1' then
    PRESENT_STATE <= START;
    OUTPUT <= '0';
  elsif CLK'event and CLK='1' then
    OUTPUT <= '0';
    case PRESENT_STATE is
      when START =>
        if INPUT='1' then
          PRESENT_STATE <= OPENED;
          OUTPUT <= '1';
        else
          PRESENT_STATE <= START;
          OUTPUT <= '0';
        end if;
      when OPENED =>
          PRESENT_STATE <= CLOSED;
          OUTPUT <= '0';
    end case;
  end if;
end process;
State Machines- IX

- Simulation output:
• Some general guidelines for optimally modeling state machines in VHDL
• Single process!
• Reset conditions
  – For reset, specify values for each of the outputs and also the state

```vhdl
if RESET='1' then

  PRESENT_STATE <= ... ;

-- set values for all outputs on reset
```

State Machines- X
State Machines- XI

• State and output control
  – Specify default values for all the system outputs and state before any explicit tests
  – If an output is not set for all possible conditions then it is assumed by the synthesis tool that the output doesn’t change and latches will be inferred.

    elsif CLK’event and CLK=’1’ then

      -- set default outputs and state

      case PRESENT_STATE is
State Machines- XI

• How are the states encoded?
  – In the above examples we did not specify the binary values for each of the states (remember they are held by FF’s)
  – Different possibilities (generally can set at synthesis time)
• Sequential coding
  – States are allocated in sequence (00, 01, 10, 11, etc.)
• Gray
  – Only a single bit changes at a time (00, 01, 11, 10, etc.)
• One hot
  – One FF per state for simple decoding (001, 010, 100, etc.)