Design and Modeling Recommendations

- General recommendations
  - make models generic and thus promoting re-use
  - use subprograms to make the code shorter
- Use abstract data types to make models easier to read
  - enumerated types
  - example

Simulation Accuracy

- Ensure complete sensitivity lists
  - otherwise the simulation may not accurately reflect the physical system

```vhdl
architecture behavioral of test_entity is
begin
  process (A, B)
  variable var1, var2: std_logic;
  begin
    var1 := A and B and C;
    var2 := var1 nand D;
    out_sig<= var1 and var2;
  end architecture behavioral;
```
Simulation Accuracy

- Impact of sensitivity lists and statement triggering semantics
  - how many delta delays does the following code take to stabilize?
    - when A, B, or C change?
    - when D changes?

```vhdl
architecture behavioral of test_entity is
signal sig1, sig2: std_logic;
begn
  sig1 <= A and B and C;
  Y1 <= not sig2;
  sig2 <= sig1 and D;
end architecture behavioral;
```

Simulation Time

- Improving simulation time
  - use processes
  - minimize the number of signals in a sensitivity list leading to less number of signals to monitor
  - minimize the number of processes
    - move groups of logic being clocked from the same source to the same process
  - use integer data types rather than vectored data types e.g., signed and unsigned
  - use variables in a process rather than signals
  - move loop invariant code out of loop and conditional branches
More on Simulation

- Managing entity port types
  - problem: reading output ports
  - solution - use temporary variables

Problem:

```vhdl
for i in 0 to 31 loop;
  ..
  count <= count + 1;
  ..
end loop;

Solution:

.. var_count : var_count + 1; -- in the loop

count <= var_count -- outside the process
```

Signals vs. Variables

- Use of signals vs. variables in a process

```vhdl
architecture behavior of sig_var is
signal sig_s1, sig_s2 : std_logic;
beginn
proc1: process (x, y, z) is -- Process 1
variable var_s1, var_s2: std_logic;
beginn
L1: var_s1 := x and y;
L2: var_s2 := var_s1 xor z;
L3: res1 <= var_s1 nand var_s2;
end process;

proc2: process (x, y, z) -- Process 2
beginn
L1: sig_s1 <= x and y;
L2: sig_s2 <= sig_s1 xor z;
L3: res2 <= sig_s1 nand sig_s2;
end process;
```
Signals vs. Variables

More on Simulation

- Use of deferred constants
  - place constant values in package bodies
  - note compilation dependencies
  - optimizing the recompilation time in large models
Simulating State Machines

- Standard model for capturing deterministic sequences of activities

- Use of asynchronous/synchronous reset
  - no decoding of unused states
  - minimizes next state logic

- State encodings

<table>
<thead>
<tr>
<th>State</th>
<th>Sequential</th>
<th>Gray Code</th>
<th>One Hot</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
<td>0000000001</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>001</td>
<td>00000010</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>011</td>
<td>00000100</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>010</td>
<td>00001000</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>110</td>
<td>00010000</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>111</td>
<td>00100000</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>101</td>
<td>01000000</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>100</td>
<td>10000000</td>
</tr>
</tbody>
</table>