

# Designs

- Consist of an entity and an architecture describing it
- Architecture is synthesizable
- May have sub-components

# How to Create a Register

```
process(clk, rst)
begin
    if (rst = '1') then
        a <= "00000000";
    elsif(clk'event and clk = '1')
        a <= new_a;
    end if;
end process;
```

# How to Create a Register w/ an Enable

```
process(clk, rst)
begin
    if (rst = '1') then
        a <= "00000000";
    elsif(clk'event and clk = '1')
        if (enable = '1') then
            a <= new_a;
        end if;
    end if;
end process;
```

# How to Create a Register w/ Assorted Logic

```
process(clk, rst)
begin
    if (rst = '1') then
        a <= "00000000";
    elsif(clk'event and clk = '1')
        if (enable = '1') then
            if (a > b) then
                a <= a+b;
            else
                a <= a - b;
            end if;
        end if;
    end if;
end process;
```

# State Machines – Hand Coding

## State Declarations

```
constant STATE1 : std_logic_vector(3 downto 0) := "0001";  
constant STATE2 : std_logic_vector(3 downto 0) := "0010";  
constant STATE3 : std_logic_vector(3 downto 0) := "0100";  
constant STATE4 : std_logic_vector(3 downto 0) := "1000";  
signal NEXT_STATE: std_logic_vector(3 downto 0);  
signal STATE      : std_logic_vector(3 downto 0);
```

# State Machines – Automated State Declarations

```
TYPE state_type IS (STATE1, STATE2, STATE3, STATE4);  
signal STATE : state_type;  
signal NEXT_STATE : state_type;
```

# State Machines

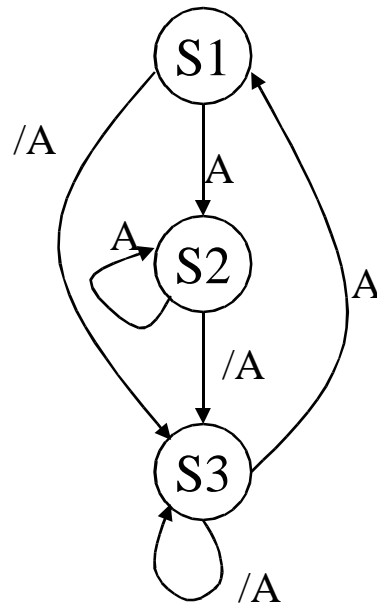
## Processes

```
process(clk, rst)
begin
    if (rst = '1') then
        STATE <= STATE1;
    elsif(clk'event and clk='1') then
        STATE <= NEXT_STATE;
    end if;
end process;
```

# State Machines Processes

```
process(STATE, input1, input2)
begin
  case (STATE) is
    when STATE1 =>
      if(input1 = '1') then
        NEXT_STATE <= STATE2;
      else
        NEXT_STATE <= STATE1;
      end if;
    when STATE2 =>
      .
      .
      .
    when others =>
      NEXT_STATE <= "XXXX";
  end case;
end process;
```

# Example



# State Machines

## Processes

```
process (STATE, A)
begin
  case (STATE) is
    when S1 =>
      if (A = '1') then NEXT_STATE <= S2;
      else NEXT_STATE <= S1;
      end if;
    when S2 =>
      if (A = '0') then NEXT_STATE <= S3;
      else NEXT_STATE <= S2;
      end if;
    when S3 =>
      if (A = '1') then NEXT_STATE <= S1;
      else NEXT_STATE <= S3;
      end if;
    when others => NEXT_STATE <= "XX";
  end case;
end process;
```