

## Advanced VHDL

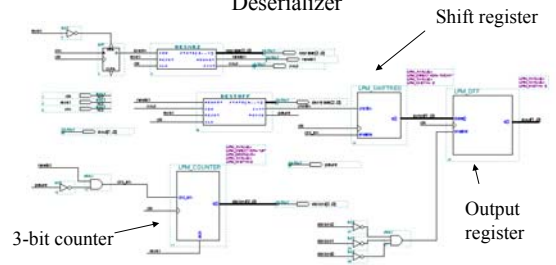
- In this class, have used VHDL primarily finite state machines (random logic + state registers)
- Have used schematic capture + LPMs for datapaths
  - Schematics are nice because a picture can convey the design structure
  - But schematic files are non-portable, file format is specific to Altera Maxplus
- Can use VHDL for entire design including datapath, and connections
  - More portable than schematics, easier to modify
  - Many tools only accept textual representations of designs (Verilog, VHDL, other hardware description languages).

4/22/2002

BR

1

## Deserializer



Goal: Replace Deserializer with a single VHDL file, will include 'desnrz', 'destuff' as components.

4/22/2002

BR

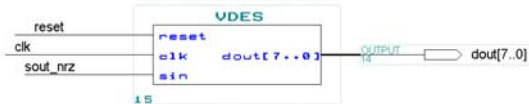
2

## Entity Declaration

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

entity vdes is
  port ( reset,clk,sin : in std_logic;
        dout: out std_logic_vector(7 downto 0)
        );
end vdes;
```

Provides arithmetic functionality needed by counter



4/22/2002

BR

3

## Architecture – Component Declarations

*desnrz*, *destuff* are described by separate VHDL files, will be used as components within 'vdes.vhd'.

```
architecture a of vdes is
-- component declarations
  component destuff
  port ( newbit,sin: in std_logic;
        reset,clk : in std_logic;
        state: out std_logic_vector(2 downto 0);
        sout,pause: out std_logic );
  end component;
  component desnrz
  port ( sin: in std_logic;
        reset,clk : in std_logic;
        state: out std_logic_vector(2 downto 0);
        newbit,sout: out std_logic
        );
  end component;
```

Component declarations are basically just copies of entity definitions.

4/22/2002

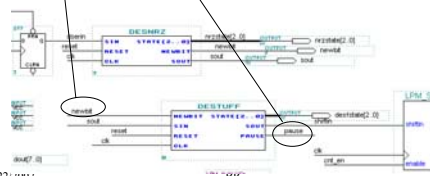
BR

4

## Signals

Need to declare signals that are used to tie components/processes together

```
-- signals
signal dserin : std_logic;
signal dshift : std_logic_vector(7 downto 0);
signal nrzstate: std_logic_vector(2 downto 0);
signal newbit, sout: std_logic;
signal desrstate: std_logic_vector(2 downto 0);
signal shiftin, pause: std_logic;
signal dhitcnt: std_logic_vector(2 downto 0);
signal cnt_en : std_logic;
```



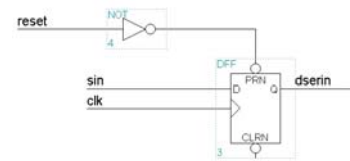
4/22/2002

BR

5

## Input DFF for *sin*

```
-- input DFF
process(reset,clk)
begin
  if (reset = '1') then
    dserin <= '1';
  elsif (clk'event and clk = '1') then
    dserin <= sin;
  end if;
end process;
```



4/22/2002

BR

6

### Shift Register

```

-- shift register
process(clk)
begin
  if (clk'event and clk = '1') then
    if (cnt_en = '1') then
      dshift(6 downto 0) <= dshift(7 downto 1);
      dshift(7) <= shiftin;
    end if;
  end if;
end process;
cnt_en <= newbit and (not pause);

```

Note that this is a right shift, and that *shiftin* comes into MSB.

LPM\_AVALUE=  
LPM\_DIRECTION="RIGHT"  
LPM\_SVALUE=  
LPM\_WIDTH= 8

4/22/2002 BR 7

### 3-bit Counter

```

-- 3 bit counter
process(reset,clk)
begin
  if (reset = '1') then
    dbitcnt <= "000";
  elsif (clk'event and clk = '1') then
    if (cnt_en = '1') then
      dbitcnt <= dbitcnt + 1 ;
    end if;
  end if;
end process ;

```

Increment by 1

LPM\_AVALUE=  
LPM\_DIRECTION="UP"  
LPM\_MODULUS=  
LPM\_SVALUE=  
LPM\_WIDTH=3

4/22/2002 BR 8

### Output Register

```

-- output register
process(reset,clk)
begin
  if (clk'event and clk = '1') then
    if (dbitcnt(2) = '0' and
       dbitcnt(1) = '0' and
       dbitcnt(0) = '0') then
      dout <= dshift;
    end if;
  end if;
end process;

```

LPM\_AVALUE=  
LPM\_SVALUE=  
LPM\_WIDTH= 8

4/22/2002 BR 9

### desnrz Component Instantiation, Connection

```

-- desnrz
ul:desnrz
port map(
  sin => dserin,
  state => nrzstate,
  newbit => newbit,
  sout => sout);

```

Component type  
Component Pin name  
Signal(net) name  
Component name (must be unique in VHDL file)

4/22/2002 BR 10

### destuff Component Instantiation, Connection

```

-- destuff
u2:destuff
port map(
  clk => clk,
  reset => reset,
  newbit => newbit,
  state => desfstate,
  sout => shiftin,
  pause => pause);

```

Mapping of component pins to signals (nets)

4/22/2002 BR 11

### Other Comments

- The *des* schematic used LPM components such as LPM\_DFF, LPM\_SHIFTRREG, LPM\_COUNTER
  - These could have been called as 'components' from VHDL instead of describing their functionality via VHDL processes
  - Their functionality is so simple, was easier to simply use VHDL processes
- For more complex blocks such as SRAMs, Multipliers, etc, that correspond directly to FPGA resources, would have used the LPM versions
  - Instantiate the LPMs directly in the VHDL code same as *desnrz*, *destuff*
  - Component declarations are predeclared, need to include the following in the VHDL file:
 

```

library lpm;
use lpm.lpm_components.all;

```

4/22/2002 BR 12