EE-334
Digital System Design

Discussion for Exercise 3
VHDL and Block Diagrams

Andreas Burg

Task 1: 2-to-1 Multiplexer

- Find the problems in the given code:
  - IF statement has no ELSE and no default assignment:
    - Code will not work as a multiplexer
    - A latch is produced since **OUTxDO** should only follow In1xDI if **SelxSI = '1'**

```
process(all)
begin
  if SelxSI = '1' then
    OutxDO <= In1xDI;
  end if;
end process;
```

✗

- Correct the code to fix the problem: Either default statement or else

```
process(all)
begin
  OutxDO <= In0xDI;

  if SelxSI = '1' then
    OutxDO <= In1xDI;
  end if;
end process;
```

✓

```
process(all)
begin
  if SelxSI = '1' then
    OutxDO <= In1xDI;
  else
    OutxDO <= In0xDI;
  end if;
end process;
```

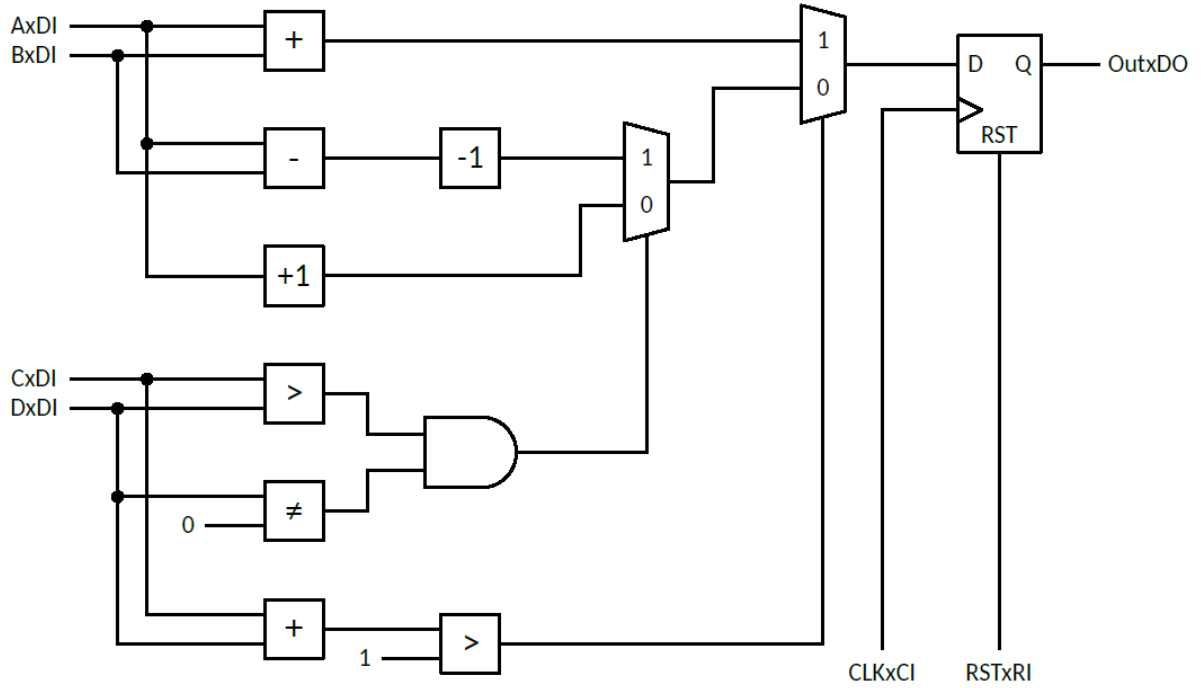
✓

Task 2: Some Arbitrary Combinational Circuit

- Find and correct the errors in the code:
  - Two **syntax errors**
    - Missing semicolon (`;`) in the last line
    - Missing `ELSE` statement
  - **One semantic error**: clocked process assigns **`ResxDN`** instead of **`RexDP`** in one branch of the `IF` statement

```
  signal ResxDN, ResxDP : unsigned(8-1 downto 0);

begin

  process(CLKxCI, RSTxRI)
  begin
    if (RSTxRI = '1') then
      ResxDP <= (others => '0');
    elsif (CLKxCI'event and CLKxCI = '1') then
      ResxDN <= ResxDP;
    end if;
  end process;
  ResxDN <= AxDI + BxDI     when CxDI + DxDI > 1 else
            AxDI - BxDI - 1 when CxDI > DxDI and DxDI /= 0 ELSE
            AxDI + 1;
```

- Draw the schematic:



Task 3a: Another Arbitrary Combinational Circuit

- Find and correct the errors in the code:
  - Two **syntax errors**
    - 2x missing THEN keyword
- **Two semantic errors**:
  - Clocked process is missing the `RSTxRI` signal in the sensitivity list
  - Combinatorial logic process is missing many signals in the sensitivity list: to avoid enumerating all signals we can use **`ALL`** instead

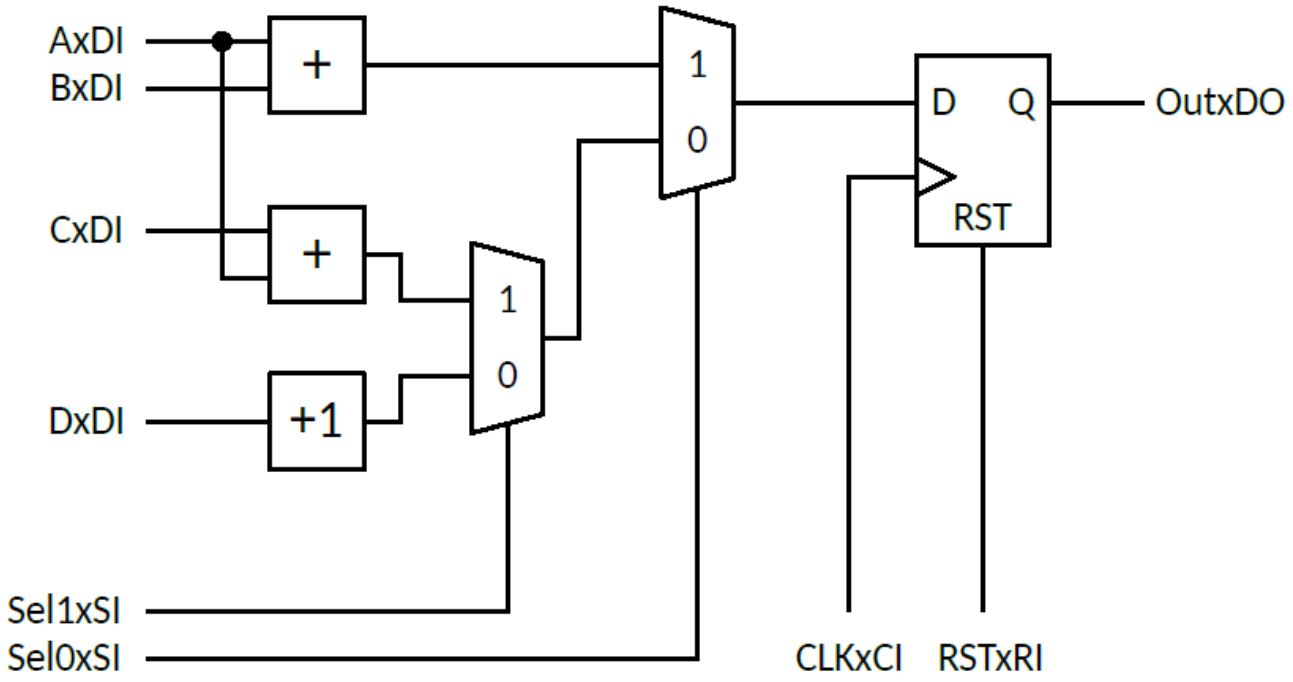
```
  signal ResxDN, ResxDP : unsigned(8-1 downto 0);

begin

  process(CLKxCI)
  begin
    if (RSTxRI = '1') then
      ResxDP <= (others => '0');
    elsif (CLKxCI'event and CLKxCI = '1') then
      ResxDP <= ResxDN;
    end if;
  end process;

  process()
  begin
    if Sel0xSI = '1' THEN
      ResxDN <= AxDI + BxDI;
    elsif Sel1xSI = '1' THEN
      ResxDN <= AxDI + CxDI;
    else
      ResxDN <= DxDI + 1;
    end if;
  end process;
```

(CLKxCI, RSTxRI)

(ALL)

Task 3a: Another Arbitrary Combinational Circuit

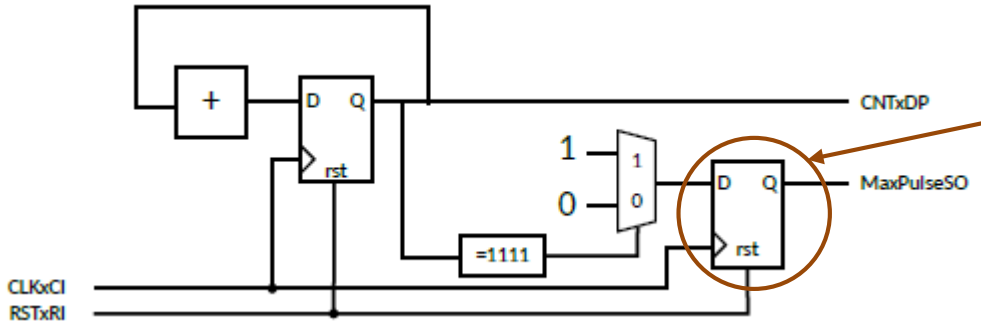
- Reconstruct the schematic of the circuit




```
  signal ResxDN, ResxDP : unsigned(8-1 downto 0);

begin

  process(CLKxCI, RSTxRI)
  begin
    if (RSTxRI = '1') then
      ResxDP <= (others => '0');
    elsif (CLKxCI'event and CLKxCI = '1') then
      ResxDP <= ResxDN;
    end if;
  end process;

  process(all)
  begin
    if Sel0xSI = '1' then
      ResxDN <= AxDI + BxDI;
    elsif Sel1xSI = '1' then
      ResxDN <= AxDI + CxDI;
    else
      ResxDN <= DxDI + 1;
    end if;
  end process;
```

Task 4: Counter indicating when it reaches its MAX

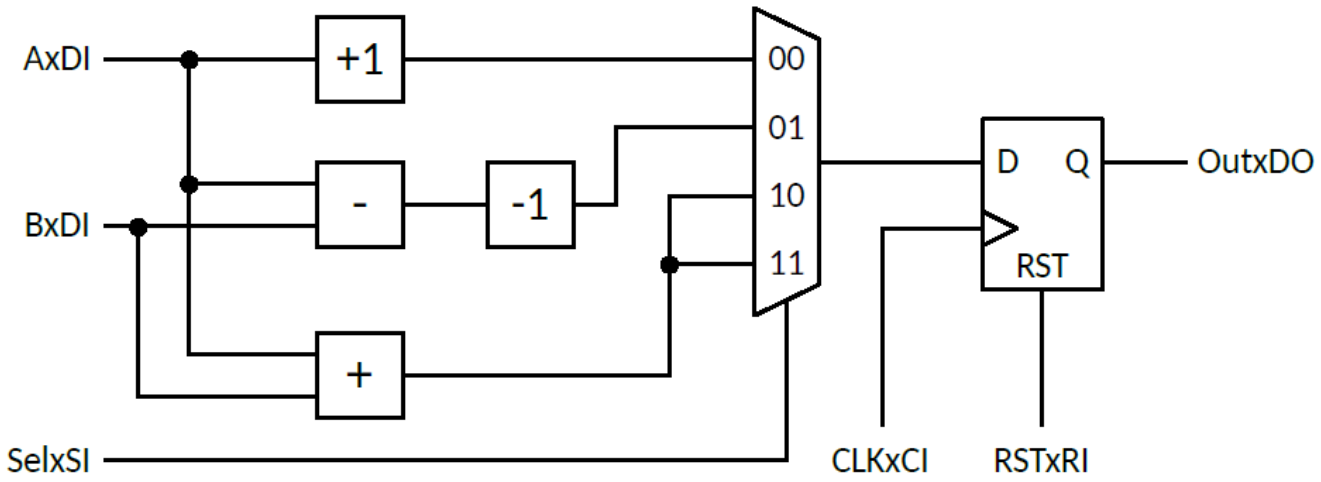
- A free-running counter `CNTxDP` that sets `MaxPulsexSO='1'` in the **same cycle** when it (`CNTxDP`) reaches its maximum value `"1111"`
- Find the issue(s) with the VHDL code:
  - **Serious synchronous design issue**:
    `MaxPulsesxSO` has no reset and should only be clocked when `RSTxRI` is not '1'
    ➔ lead to logic that combines `RSTxRI` with `CLKxCI`
  - **Functional issue**:
    - `MaxPulsexSO` is assigned inside a clocked process (`IF clk'event` statement) ➔ Signal is output of a register




```vhdl
  signal CNTxDP : unsigned(4-1 downto 0);

begin

  process(CLKxCI, RSTxRI)
  begin
    if (RSTxRI = '1') then
      CNTxDP <= (others => '0');
    elsif (CLKxCI'event and CLKxCI = '1') then
      CNTxDP <= CNTxDP + 1;
      if CNTxDP = "1111" then
        MaxPulsexSO <= '1';
      else
        MaxPulsexSO <= '0';
      end if;
    end if;
  end process;
```

Task 5: Block Diagram to VHDL

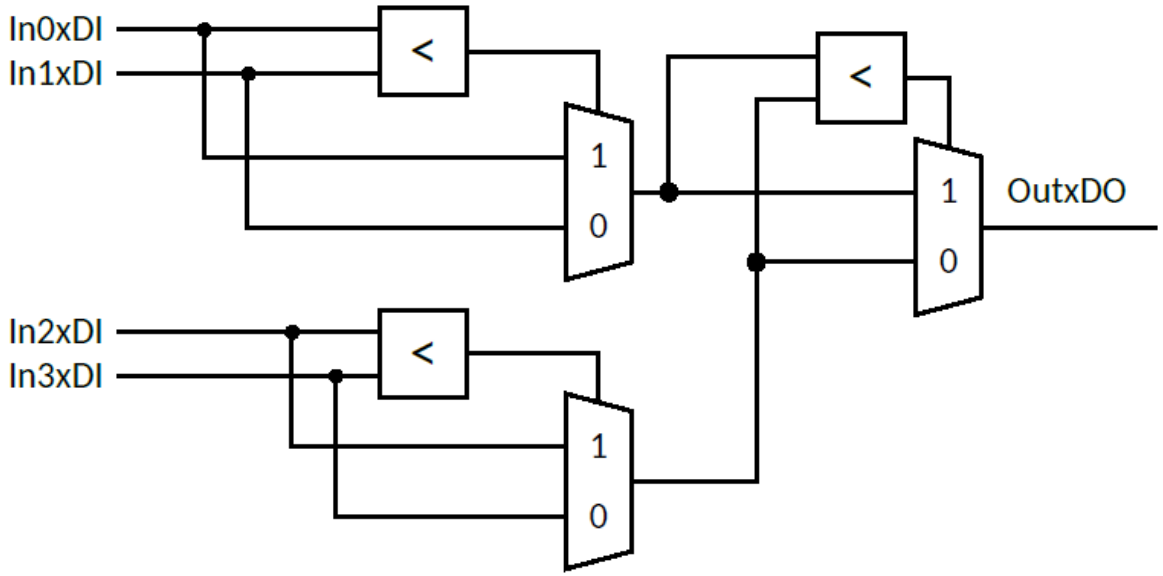
- Arithmetic translates into corresponding logic blocks
- MUX: some form of conditional statement controlled by `SelxSI`




```
  signal ResxDN, ResxDP : unsigned(8-1 downto 0);

begin

  process(CLKxCI, RSTxRI)
  begin
    if (RSTxRI = '1') then
      ResxDP <= (others => '0');
    elsif (CLKxCI'event and CLKxCI = '1') then
      ResxDP <= ResxDN;
    end if;
  end process;

  with SelxSI select
    ResxDN <= AxDI + 1         when "00",
              AxDI - BxDI - 1  when "01",
              AxDI + BxDI      when "10" | "11",
              (others => 'X')  when others;

  OutxDO <= ResxDP;
```

Task 6: Block Diagram to VHDL

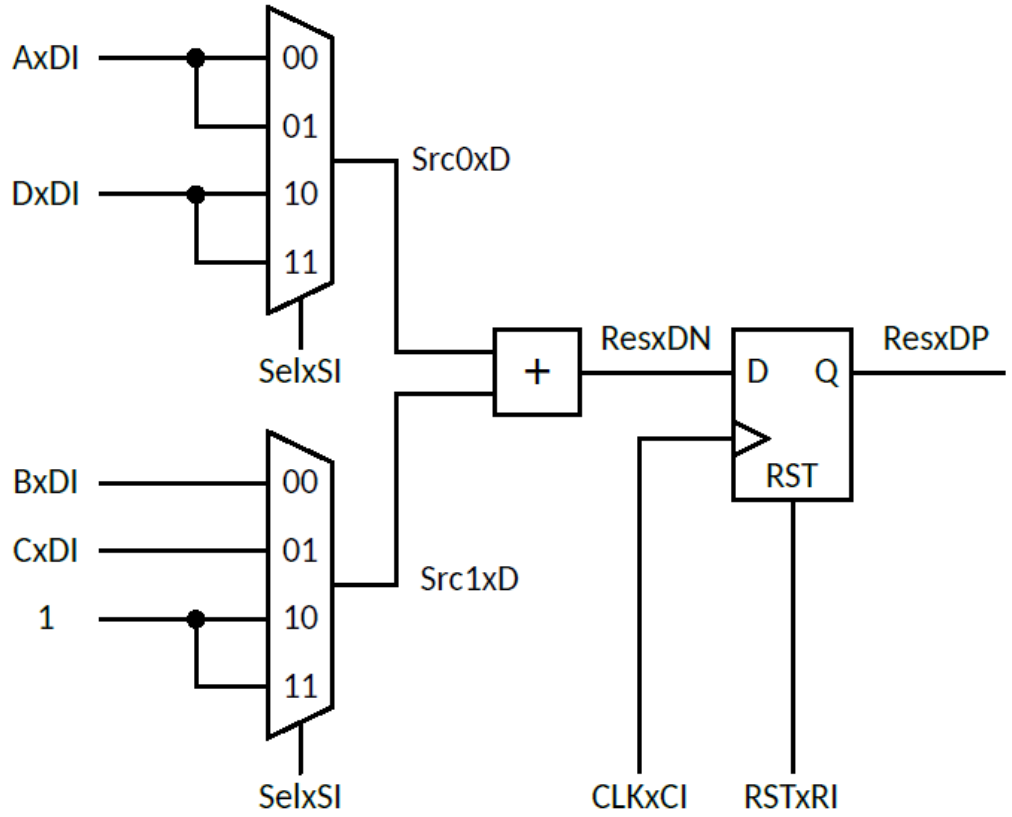
- Multiple MUXes: describe one-by one and connect
- Comparators that control the MUXes can be realized
  - **As combinational logic that generates explicit select signals controlling the MUX (i.e., conditional assignments in VHDL) OR**
  - Directly in the combinational statement




```
  signal Res0xD, Res1xD, Res2xD : unsigned(8-1 downto 0);
  signal Comp0xS, Comp1xS, Comp2xS : std_logic;

begin
  Comp0xS <= '1' when In0xDI < In1xDI else '0'; -- Comparators
  Comp1xS <= '1' when In2xDI < In3xDI else '0';
  Comp2xS <= '1' when Res0xD < Res1xD else '0';

  Res0xD <= In0xDI when Comp0xS = '1' else In1xDI; -- Mux
  Res1xD <= In2xDI when Comp1xS = '1' else In3xDI;
  Res2xD <= Res0xD when Comp2xS = '1' else Res1xD;

  OutxDO <= Res2xD;
```

Task 7: Block Diagram to VHDL

- Adder with multiple possible inputs (shared resource) with a register




```
  signal Src0xD, Src1xD : unsigned(8-1 downto 0);
  signal ResxDN, ResxDP : unsigned(8-1 downto 0);

begin

  process(CLKxCI, RSTxRI)
  begin
    if (RSTxRI = '1') then
      ResxDP <= (others => '0');
    elsif (CLKxCI'event and CLKxCI = '1') then
      ResxDP <= ResxDN;
    end if;
  end process;

  with SelxSI select
    Src0xD <= AxDI            when "00"|"01",
              DxDI            when "10"|"11",
              (others => 'X') when others;

  with SelxSI select
    Src1xD <= BxDI            when "00",
              CxDI            when "01",
              "00000001"      when "10"|"11",
              (others => 'X') when others;

  ResxDN <= Src0xD + Src1xD;

  OutxDO <= ResxDP;
end rtl;
```