

Hi John,

As I promised to you at the ROD workshop in Geneva, I composed a file to show you how I implemented a small portion of the behavior of your TIM in VHDL.

I hope this will encourage you to have fun with VHDL yourself. Fun it is!

In my ROD design the most important signals that I need are the Event-ID and Bunch-ID, which are serialized in the TTC4\_n signal, and the Trigger Type, which is serialized in TTC5\_n.

We use a tool called “EASE” to visualize the design. The first figure shows what my simplified VHDL TIM looks like (see figure 1). The figure 2 is a hierarchical level down into the “Par\_To\_Ser” entity. Below this level the design is only composed of pieces of VHDL code, for example like register U5:REG1:

ARCHITECTURE a0 OF Reg1 IS

BEGIN

  Process (Clk, Rst\_n)

  Begin

    If Rst\_n = '0' Then

      Q <= '0';

    Elsif Rising\_Edge(Clk) Then

      Q <= D;

    End If;

  End Process;

END a0 ; -- of Reg1

“EASE” can generate the VHDL code by taking all hierarchical information and all bits and pieces of VHDL code you’ve written. This results in a file as shown below.

I designed the serializer so that it can easily be interfaced to a FIFO. ‘Emp\_EVBCID’ active means that the Event/Bunch-ID FIFO is empty. As soon as this signal goes inactive the serializer reads a word from the FIFO by means of the “RReq\_EVBCID” (Read Request) signal. You can see this happen in the simulation (see figure 3). The serializer starts to clock out all the bits on the TTC4\_n signal. I only simulated the Serialized Event ID and Bunch ID signal TTC4. The TTC5 signal works more or less the same.

Best Regards,

Peter Jansweijer

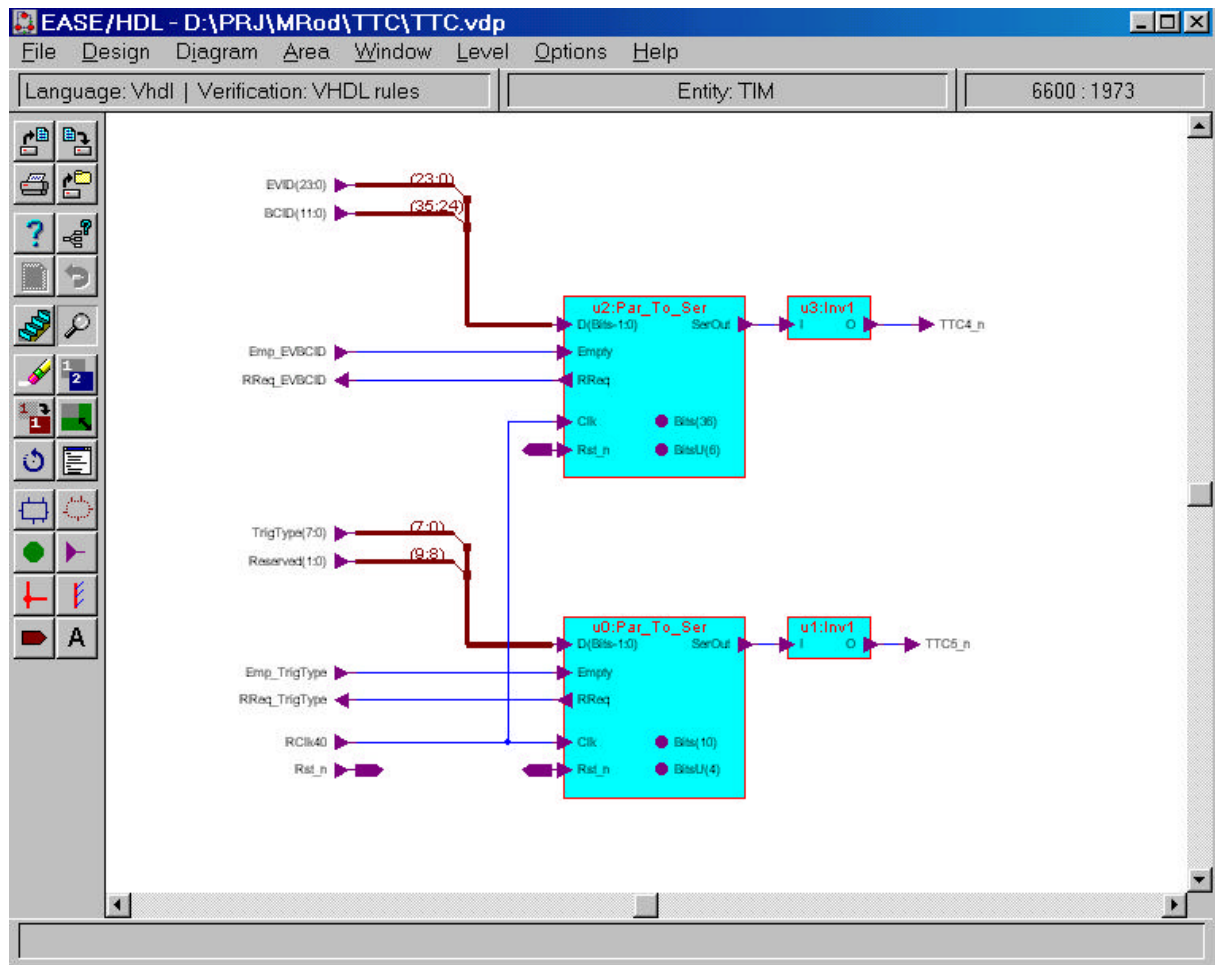


Figure 1: A view of entity TIM

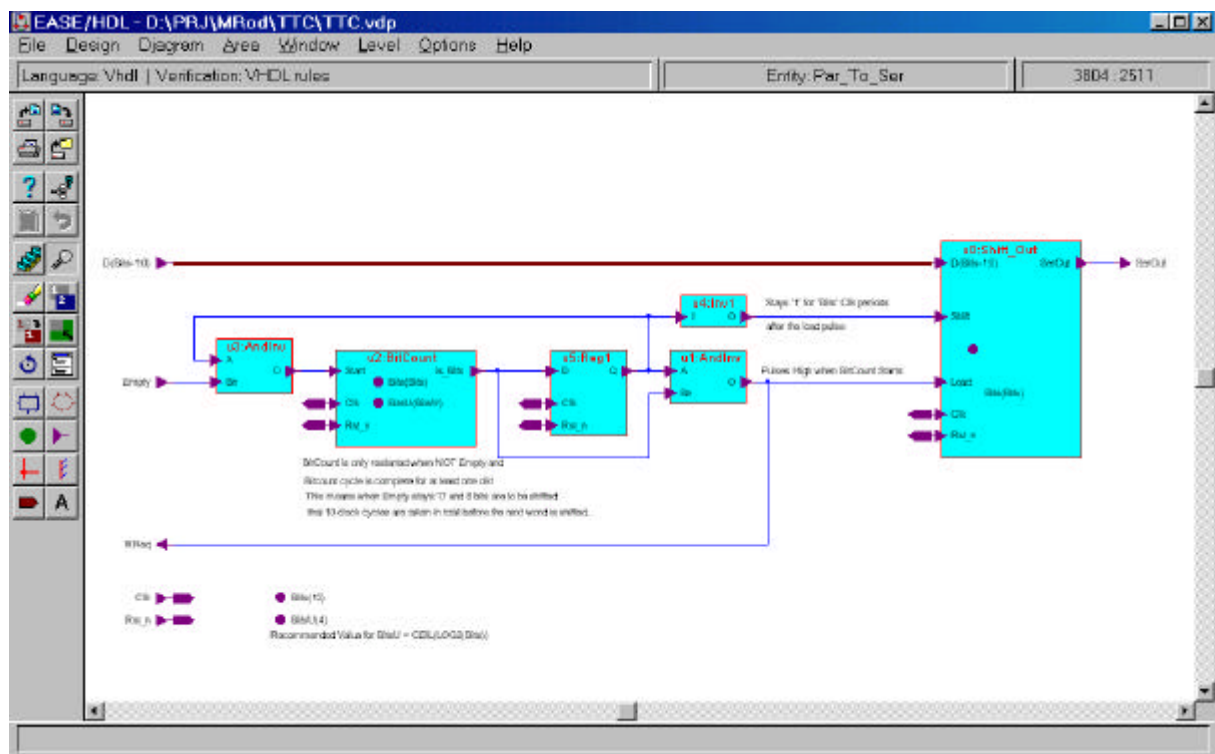


Figure 2: A view of entity Par\_To\_Ser

-----  
-- This Vhdl file is generated by EASE/HDL from TRANSLOGIC BV,  
-- the 'Graphical Systems Design Tool' tool.

--  
-- EASE/HDL Version 3.0 (Revision 4).  
-- Time stamp : Tue Oct 10 09:50:19 2000.  
--

-- Designed by : .  
-- Company : .  
-- Design info : .  
-----

-----  
-- Entity declaration of 'Shift\_Out'.  
-----

library ieee ;  
use ieee.std\_logic\_1164.ALL ;  
use ieee.numeric\_std.all ;

entity Shift\_Out is  
  generic(  
    Bits : Positive := 10 ) ;  
  port(  
    Clk : in std\_logic ;  
    Rst\_n : in std\_logic ;  
    SerOut : out std\_logic ;  
    D : in std\_logic\_Vector(Bits-1 downto 0) ;  
    Load : in std\_logic ;  
    Shift : in std\_logic ) ;  
end Shift\_Out ;

-----  
-- Architecture 'a0' of 'Shift\_Out'  
-----

ARCHITECTURE a0 OF Shift\_Out IS  
begin  
  Process (Clk, Rst\_n)  
    Variable Q\_Int: Std\_Logic\_Vector(Bits-1 downto 0);  
  Begin  
    If Rst\_n = '0' Then  
      SerOut <= '0';  
      Q\_Int := (Others => '0');  
    ElsIf Rising\_Edge(Clk) Then  
      If Load = '1' Then  
        Q\_Int := D;  
        SerOut <= '1';  
      ElsIf Shift = '1' Then  
        SerOut <= Q\_Int(0);

```

        For I In 0 To Bits - 2 Loop
            Q_Int(I) := Q_Int(I+1);
        End Loop;
    Else
        SerOut <= '0';
    End If;
End If;
End Process;
END a0 ; -- of Shift_Out

```

---

```

-- Entity declaration of 'Reg1'.

```

---

```

library ieee ;
use ieee.std_logic_1164.ALL ;
use ieee.numeric_std.all ;

```

```

entity Reg1 is
    port(
        Clk  : in    std_logic ;
        D    : in    std_logic ;
        Rst_n : in    std_logic ;
        Q    : out   std_logic ) ;
end Reg1 ;

```

---

```

-- Architecture 'a0' of 'Reg1'

```

---

```

ARCHITECTURE a0 OF Reg1 IS

```

```

BEGIN
    Process (Clk, Rst_n)
    Begin
        If Rst_n = '0' Then
            Q <= '0';
        Elself Rising_Edge(Clk) Then
            Q <= D;
        End If;
    End Process;
END a0 ; -- of Reg1

```

---

```

-- Entity declaration of 'Inv1'.

```

---

```

library ieee ;
use ieee.std_logic_1164.ALL ;
use ieee.numeric_std.all ;

```

```

entity Inv1 is
  port(
    I : in    std_logic ;
    O : out   std_logic ) ;
end Inv1 ;

```

---

```

-- Architecture 'a0' of 'Inv1'

```

---

```

ARCHITECTURE a0 OF Inv1 IS

```

```

BEGIN
  O <= Not I;
END a0 ; -- of Inv1

```

---

```

-- Entity declaration of 'BitCount'.

```

---

```

library ieee ;
use ieee.std_logic_1164.ALL ;
use ieee.numeric_std.all ;

```

```

entity BitCount is
  generic(
    Bits : Positive := 10 ;
    BitsU : Positive := 4 ) ;
  port(
    Clk : in    std_logic ;
    Rst_n : in   std_logic ;
    Start : in   std_logic ;
    Is_Bits : out std_logic ) ;
end BitCount ;

```

---

```

-- Architecture 'a0' of 'BitCount'

```

---

```

architecture a0 of BitCount is
BEGIN
  Process (Clk, Rst_n)
    Variable Cnt: Unsigned(BitsU - 1 Downto 0);
  Begin
    If Rst_n = '0' Then
      Cnt := To_Unsigned(Bits,BitsU);
    Elself Rising_Edge(Clk) Then
      If To_Integer(Cnt) = Bits Then
        If Start = '1' Then

```

```

        Cnt := (Others => '0');
    End If;
Else
    Cnt := Cnt + 1;
End If;
End If;

If To_Integer(Cnt) = Bits Then
    Is_Bits <= '1';
Else
    Is_Bits <= '0';
End If;
End Process;
end a0 ; -- of BitCount

```

---

```

-- Entity declaration of 'AndInv'.

```

---

```

library ieee ;
use ieee.std_logic_1164.ALL ;
use ieee.numeric_std.all ;

```

```

entity AndInv is
    port(
        A : in    std_logic ;
        Bn : in    std_logic ;
        O : out    std_logic ) ;
end AndInv ;

```

---

```

-- Architecture 'a0' of 'AndInv'

```

---

```

ARCHITECTURE a0 OF AndInv IS

```

```

BEGIN
    O <= a And Not Bn;
END a0 ;

```

---

```

-- Entity declaration of 'Par_To_Ser'.

```

---

```

library ieee ;
use ieee.std_logic_1164.ALL ;
use ieee.numeric_std.all ;

```

```

entity Par_To_Ser is
    generic(

```

```

    Bits : Positive := 10 ;
    BitsU : Positive := 4 ) ;
port(
    Clk : in    std_logic ;
    Rst_n : in    std_logic ;
    SerOut : out  std_logic ;
    D : in    std_logic_Vector(Bits-1 downto 0) ;
    Empty : in    std_logic ;
    RReq : out  std_logic ) ;
end Par_To_Ser ;

```

---

```

-- Architecture 'a0' of 'Par_To_Ser'

```

---

architecture a0 of Par\_To\_Ser is

```

component BitCount
generic(
    Bits : Positive := 10 ;
    BitsU : Positive := 4 ) ;
port(
    Clk : in    std_logic ;
    Rst_n : in    std_logic ;
    Start : in    std_logic ;
    Is_Bits : out  std_logic ) ;
end component ;
component Reg1
port(
    Clk : in    std_logic ;
    D : in    std_logic ;
    Rst_n : in    std_logic ;
    Q : out  std_logic ) ;
end component ;
component Shift_Out
generic(
    Bits : Positive := 10 ) ;
port(
    Clk : in    std_logic ;
    Rst_n : in    std_logic ;
    SerOut : out  std_logic ;
    D : in    std_logic_Vector(Bits-1 downto 0) ;
    Load : in    std_logic ;
    Shift : in    std_logic ) ;
end component ;
component AndInv
port(
    A : in    std_logic ;
    Bn : in    std_logic ;

```

```

    O : out    std_logic ) ;
end component ;
component Inv1
port(
    I : in     std_logic ;
    O : out    std_logic ) ;
end component ;
signal Net_0 : std_logic ;
signal Net_2 : std_logic ;
signal Net_3 : std_logic ;
signal Net_4 : std_logic ;
signal Net_6 : std_logic ;

```

```

begin
-- Recommended Value for BitsU = CEIL(LOG2(Bits))
-- BitCount is only restarted when NOT Empty and
-- Pulses High when BitCount Starts
-- Bitcount cycle is complete for at least one clk!
-- This means when Empty stays '0' and 8 bits are to be shifted
-- that 10 clock cycles are taken in total before the next word is shifted.
-- Stays '1' for 'Bits' Clk periods
-- after the load pulse

```

```

RReq <= Net_0 ;

```

```

u2: BitCount
generic map(
    Bits => Bits,
    BitsU => BitsU )
port map(
    Clk => Clk,
    Rst_n => Rst_n,
    Start => Net_2,
    Is_Bits => Net_3 ) ;

```

```

u5: Reg1
port map(
    Clk => Clk,
    D => Net_3,
    Rst_n => Rst_n,
    Q => Net_4 ) ;

```

```

u0: Shift_Out
generic map(
    Bits => Bits )
port map(
    Clk => Clk,
    Rst_n => Rst_n,
    SerOut => SerOut,
    D => D,

```



```
Load => Net_0,  
Shift => Net_6 ) ;
```

```
u3: AndInv  
port map(  
  A => Net_4,  
  Bn => Empty,  
  O => Net_2 ) ;
```

```
u1: AndInv  
port map(  
  A => Net_4,  
  Bn => Net_3,  
  O => Net_0 ) ;
```

```
u4: Inv1  
port map(  
  I => Net_4,  
  O => Net_6 ) ;  
end a0 ; -- of Par_To_Ser
```

```
-----  
-- Entity declaration of 'TIM'.  
-----
```

```
library ieee ;  
use ieee.std_logic_1164.ALL ;  
use ieee.numeric_std.all ;
```

```
entity TIM is  
port(  
  TTC5_n      : out  std_logic ;  
  TTC4_n      : out  std_logic ;  
  RClk40      : in   std_logic ;  
  Rst_n       : in   std_logic ;  
  Emp_EVBCID  : in   std_logic ;  
  EVID        : in   std_logic_Vector(23 downto 0) ;  
  TrigType    : in   std_logic_Vector(7 downto 0) ;  
  Emp_TrigType : in   std_logic ;  
  Reserved    : in   std_logic_Vector(1 downto 0) ;  
  BCID        : in   std_logic_Vector(11 downto 0) ;  
  RReq_EVBCID : out  std_logic ;  
  RReq_TrigType : out  std_logic ) ;  
end TIM ;
```

```
-----  
-- Architecture 'a0' of 'TIM'  
-----
```

```
architecture a0 of TIM is
```

```

component Par_To_Ser
generic(
  Bits : Positive := 10 ;
  BitsU : Positive := 4 ) ;
port(
  Clk : in std_logic ;
  Rst_n : in std_logic ;
  SerOut : out std_logic ;
  D : in std_logic_Vector(Bits-1 downto 0) ;
  Empty : in std_logic ;
  RReq : out std_logic ) ;
end component ;
component Inv1
port(
  I : in std_logic ;
  O : out std_logic ) ;
end component ;
signal Net_0 : std_logic ;
signal Net_1 : std_logic_Vector(9 downto 0) ;
signal Net_6 : std_logic ;
signal Net_8 : std_logic_Vector(35 downto 0) ;

```

```

begin

```

```

  Net_1(9 downto 8) <= Reserved ;
  Net_1(7 downto 0) <= TrigType ;
  Net_8(35 downto 24) <= BCID ;
  Net_8(23 downto 0) <= EVID ;

```

```

u0: Par_To_Ser
  port map(
    Clk => RClk40,
    Rst_n => Rst_n,
    SerOut => Net_0,
    D => Net_1,
    Empty => Emp_TrigType,
    RReq => RReq_TrigType ) ;

```

```

u1: Inv1
  port map(
    I => Net_0,
    O => TTC5_n ) ;

```

```

u2: Par_To_Ser
  generic map(
    Bits => 36,
    BitsU => 6 )
  port map(

```

```
Clk => RClk40,  
Rst_n => Rst_n,  
SerOut => Net_6,  
D => Net_8,  
Empty => Emp_EVBCID,  
RReq => RReq_EVBCID ) ;
```

```
u3: Inv1  
  port map(  
    I => Net_6,  
    O => TTC4_n ) ;  
end a0 ; -- of TIM
```

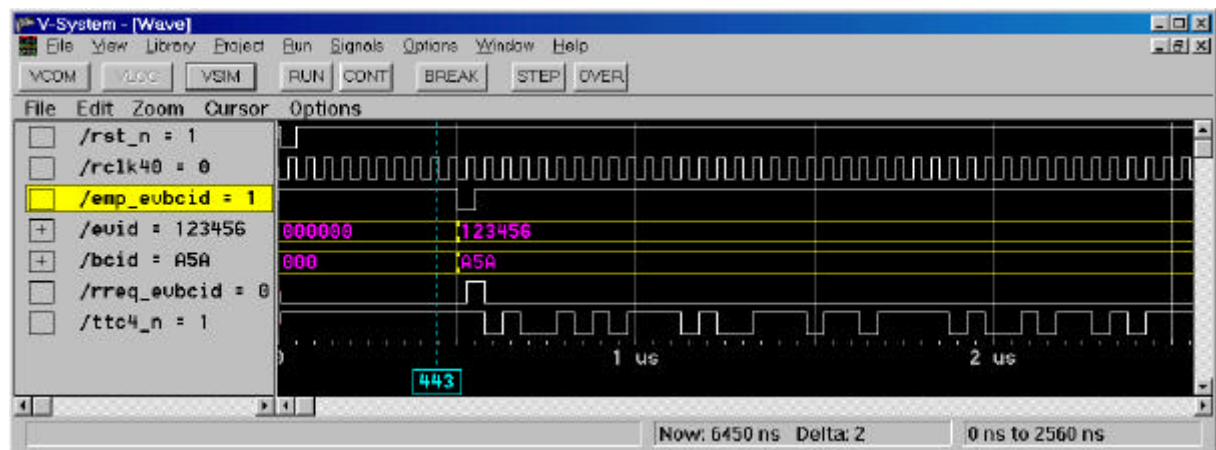


Figure 3: Simulation