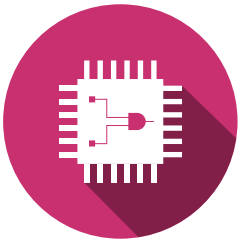


VHSIC Hardware Design Language (üb. VHDL)

Übungen zu eingebetteten Systemen

1 Einführung

2 Strukturelle Beschreibung

3 Verhaltensbeschreibung

3.1 Zuweisungen in einem Prozess

Der folgende Prozess enthält zwei Zuweisungen. Zur Zeit, wo ein Signal der Sensitivitätsliste ändert, wird er neu durchgeführt.

```
1 process(in1, in2, in3)
2 begin
3   or12  <= in1 or in2 after 1 ns;
4   or123 <= or12 or in3 after 1 ns;
5 end process;
```

Listing 1: ODER Prozess

Zeichnen Sie den Zeitverlauf der Signale `or12` und `or123` ins Zeitdiagramm der folgenden Abbildung 1.

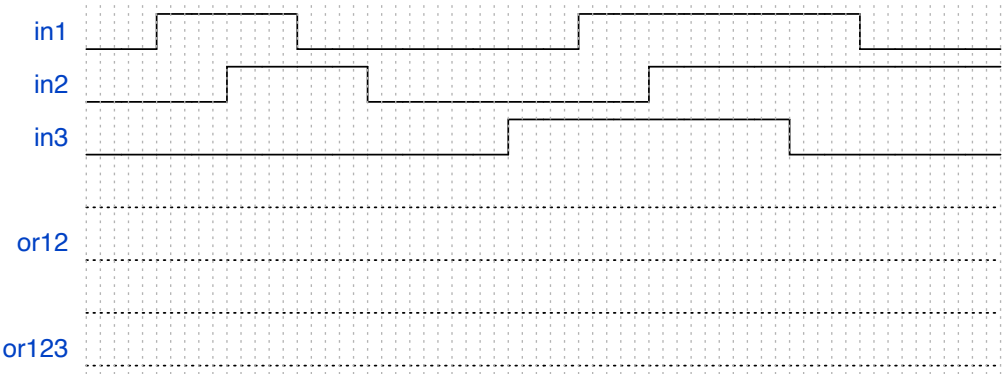
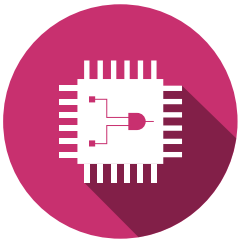


Abbildung 1: Chronogramm

Das vertikale Raster in der Abbildung 1 hat eine Dauer von 1*ns*.

3.2 Multiplexer

Schreiben Sie die Architektur des Multiplexers, dessen Entity hiernach gegeben ist.



```vhdl
library ieee;
  use ieee.std_logic_1164.all;

entity multiplexer is
  port(
    sel    : in  std_ulogic;
    in0    : in  std_ulogic;
    in1    : in  std_ulogic;
    muxOut : out std_ulogic
  );
end multiplexer;
```

Listing 2: Multiplexer `entity`

Das Steuersignal `sel` kann 9 verschiedene Werte nehmen. Weisen Sie dem Ausgang den Wert `'X'` zu, wenn der Wert des Ausgangs nicht bestimmt werden kann. Nehmen Sie in kauf, dass wenn beide Eingänge `in0` und `in1` gleich sind, dann nimmt der Ausgang denselben Wert, auch wenn der Steuersignal einen unbestimmten Wert hat.

4 Übliche Typen und Operationen

4.1 Umwandlung von Unsigned zu Signed

Schreiben Sie die Architektur der Schaltung, welche einen `unsigned` zu einem `signed` umwandelt, dessen `entity` hiernach gegeben ist.

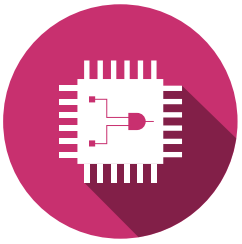
```vhdl
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all;

entity transformToSigned is
  generic(
    dataBitNb : positive := 8
  );
  port(
    signedIn  : in signed(dataBitNb-1 downto 0);
    unsignedOut : out unsigned(dataBitNb-1 downto 0)
  );
end transformToSigned;
```

Listing 3: Transform to signed `entity`

Der Bereich des Eingangssignals liegt zwischen 0 und $2^{dataBitNb} - 1$. Führen Sie die Transformation so durch, dass der Wert 0, der Minimalwert des Eingangssignals, in $-2^{dataBitNb-1}$, den Minimalwert des Ausgangssignals, umgewandelt wird und dass der Maximalwert $2^{dataBitNb} - 1$ des Eingangs den Maximalwert $2^{dataBitNb-1} - 1$ des Ausgangs ergibt. Auf diese Weise wird der Wert $2^{dataBitNb}$ des Eingangs 0 ergeben.

4.2 Verkleinerung des Bereichs eines Signals

Schreiben Sie die Architektur der Schaltung, welche den bereich eines Signals reduziert und dessen Entity hiernach gegeben ist.



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

entity rangeReducer is
  generic(
    signalBitNb     : positive := 16;
    saturationBitNb : positive := 8
  );
  port(
    inputSignal  : in  unsigned(signalBitNb-1 downto 0);
    outputSignal : out unsigned(signalBitNb-1 downto 0)
  );
end rangeReducer;
```

Listing 4: Range reducer `entity`

Die Operation reduziert die maximale Amplitude des Ausgangssignals auf $2^{saturationBitNb} - 1$. Die wird mit Sättigung erstellt: wenn das Eingangssignal grösser ist als die maximale Amplitude, so wird der Ausgang auf diesen Maximalwert gestellt.

Gehen Sie davon aus, dass `saturationBitNb` immer kleiner als `signalBitNb` ist.

4.3 Zähler mit synchroner Nullsetzung

Schreiben Sie die Architektur des Zählers mit einem synchonem Nullsetzungssignal, `signalBitNb`, dessen `signalBitNb` hiernach gegeben ist.

```vhdl
library ieee;
  use ieee.std_logic_1164.all;

entity counter is
  port(
    reset      : in  std_ulogic;
    clock      : in  std_ulogic;
    synchReset : in  std_ulogic;
    countOut   : out unsigned(7 downto 0)
  );
end counter;
```

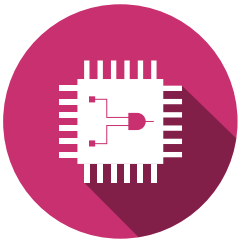
Listing 5: Zähler `entity`

Die VHDL Sprache ist sehr strikt: es ist nur möglich einen Ausgang zuzuweisen, aber nicht dessen Wert innerhalb der Architektur neu zu lesen.

4.4 Nicht wiederstartbare Verzögerung

Schreiben Sie die Architektur des Zählers, der einen Impuls von Dauer gleich eine Taktperiode erzeugt, 100 Taktperioden nach dem Schalten auf `'1'` des Eingangssignals. Seine Entity ist hiernach gegeben.

```vhdl
library ieee;
  use ieee.std_logic_1164.all;
```



```
3
4  entity counter is
5    port(
6      reset    : in  std_ulogic;
7      clock    : in  std_ulogic;
8      trigger  : in  std_ulogic;
9      pulseOut : out std_ulogic
10   );
11 end counter;
```

Listing 6: Zähler `entity`

Wenn der Eingang trigger auf `'0'` und wieder auf `'1'` schaltet bevor der Ausgangspuls gekommen ist, so wird die Schaltung diesen zusätzlichen Impuls ignorieren. Diese Schaltung bezeichnet man als non-retriggerable one-shot.

Gehen Sie davon aus, dass der Eingangssignal wieder auf `'0'` gekommen ist, bevor der Ausgangspuls erscheint.

Die VHDL Sprache ist sehr strikt: es ist nur möglich einen Ausgang zuzuweisen, aber nicht dessen Wert innerhalb der Architektur neu zu lesen.

5 Generische Parameter

5.1 Wiederstartbare Verzögerung

Schreiben Sie Entity und Architektur des Zählers, der einen Impuls von Dauer gleich eine Taktperiode erzeugt, `delay` Taktperioden nach dem Schalten auf `'1'` des Eingangssignals. Die Verzögerung `delay` ist frein zwischen 1 und 100 konfigurierbar.

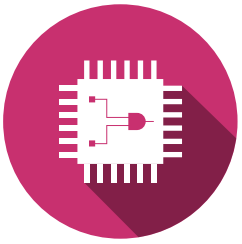
Wenn der Eingang `trigger` auf `'0'` und wieder auf `'1'` schaltet bevor der Ausgangspuls gekommen ist, so wird der interne Zähler neu gestartet. Diese Schaltung bezeichnet man als retriggerable one-shot.

Gehen Sie davon aus, dass der Eingangssignal wieder auf `'0'` gekommen ist, bevor der Ausgangspuls erscheint.

5.2 Addierer

Schreiben Sie die Architektur des Iterativaddierers, dessen Entity hiernach gegeben ist:

```
1  library ieee;
2    use ieee.std_logic_1164.all;
3    use ieee.numeric_std.all;
4
5  entity iterativeAdder is
6    port(
7      Cin  : in  std_ulogic;
8      Ai   : in  std_ulogic;
9      Bi   : in  std_ulogic;
10     Si   : out std_ulogic;
11     Cout : out std_ulogic
12   );
```



```
13 end iterativeAdder;
```

Listing 7: iterativeAdder `entity`

Mit Hilfe dieses Komponenten, schreiben Sie die Architektur des Addierers mit Übertragsfortpflanzung, dessen `entity` hiernach gegeben ist:

```
1  library ieee;
2    use ieee.std_logic_1164.all;
3    use ieee.numeric_std.all;
4
5  entity adder is
6    generic(
7      bitNb : positive := 8
8    );
9    port(
10     A        : in  unsigned (bitNb-1 downto 0);
11     B        : in  unsigned (bitNb-1 downto 0);
12     overflow : out std_ulogic;
13     S        : out unsigned (bitNb-1 downto 0)
14   );
15 end adder;
```

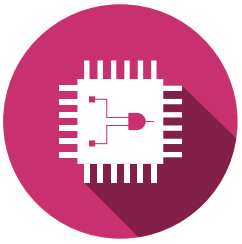
Listing 8: adder `entity`

6 Signale und Variablen

6.1 Zähler

Geben Sie die Sequenz der 3 Zählern der folgenden VHDL Architektur.

```
1  architecture RTL of threeCounters is
2    signal count1_int: unsigned(count1'range);
3    signal count2_int: unsigned(count2'range);
4  begin
5
6    cnt1: process(reset, clock)
7    begin
8      if reset = '1' then
9        count1_int <= (others => '0');
10     elsif rising_edge(clock) then
11       if count1_int = 6 then
12         count1_int <= (others => '0');
13       else
14         count1_int <= count1_int + 1;
15       end if;
16     end if;
17   end process cnt1;
18
19   count1 <= count1_int;
20
21   cnt2: process(reset, clock)
22   begin
23     if reset = '1' then
24       count2_int <= (others => '0');
```



```
25      elsif rising_edge(clock) then
26        count2_int <= count2_int + 1;
27        if count2_int = 6 then
28          count2_int <= (others => '0');
29        end if;
30      end if;
31    end process cnt2;
32
33    count2 <= count2_int;
34
35    cnt3: process(reset, clock)
36      variable count3_int: unsigned(count3'range);
37    begin
38      if reset = '1' then
39        count3_int := (others => '0');
40      elsif rising_edge(clock) then
41        count3_int := count3_int + 1;
42        if count3_int = 6 then
43          count3_int := (others => '0');
44        end if;
45      end if;
46      count3 <= count3_int;
47    end process cnt3;
48
49  end RTL;
```

Listing 9: 3 kleine Zähler

6.2 Parität

Geben Sie die VHDL Architektur des Blocks, welcher die gerade Parität eines Bit-Vektors kalkuliert und dessen `entity` hiernach gegeben ist.

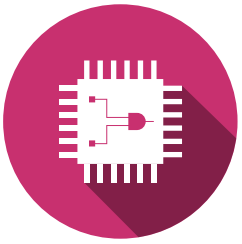
```
1   library ieee;
2     use ieee.std_logic_1164.all;
3     use ieee.numeric_std.all;
4
5   entity parityCalc is
6     generic(
7       dataBitNb : positive := 8
8     );
9     port(
10      vectorIn  : in  std_ulogic_vector(dataBitNb-1 downto 0);
11      vectorOut : out std_ulogic_vector(dataBitNb-1 downto 0)
12    );
13  end parityCalc;
```

Listing 10: parityCalc `entity`

Im Fall einer geraden Parität ist die gesamte Anzahl an Bits auf `'1'`, inklusive Paritätsbit, gerade.

6.3 Mehrheit

Schreiben Sie die Architektur der Schaltung, welche die Mehrheit (`'0'` oder `'1'`) der Bits eines Vektores mit einer ungeraden Zahl en Elementen bestimmt.



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

entity majorityFinder is
  port(
    poll     : in  std_uLogic_vector (1 to 7);
    majority : out std_ulogic
  );
end majorityFinder;
```

Listing 11: majorityFinder `entity`

Im allgemeinen erfolgt die Berechnung der Mehrheit durch das Zählen der Stimmen.

Akronyme

VHDL **V**HSIC **H**ardware **D**esign **L**anguage. 1–7