

VHDL Switch

Erzeugt von Doxygen 1.8.14

Inhaltsverzeichnis

1	Design Unit Index	1
1.1	Design Unit Hierarchy	1
2	Design Unit Index	3
2.1	Design Unit List	3
3	Datei-Verzeichnis	5
3.1	Auflistung der Dateien	5
4	Klassen-Dokumentation	7
4.1	switch Entity Reference	7
4.1.1	Ausführliche Beschreibung	8
4.2	switch Architecture Reference	8
4.2.1	Ausführliche Beschreibung	9
4.2.2	Dokumentation der Datenelemente	9
4.2.2.1	MAX	9
4.3	switch_constants Package Reference	9
4.3.1	Ausführliche Beschreibung	10
4.4	testbench Entity Reference	10
4.4.1	Ausführliche Beschreibung	11
4.5	testbench Architecture Reference	11
4.5.1	Ausführliche Beschreibung	12
5	Datei-Dokumentation	13
5.1	C:/fh/fpga/switch/switch.srcs/sim_1/new/testbench.vhd-Dateireferenz	13
5.1.1	Ausführliche Beschreibung	13
5.2	testbench.vhd	14
5.3	C:/fh/fpga/switch/switch.srcs/sources_1/new/constants.vhd-Dateireferenz	15
5.3.1	Ausführliche Beschreibung	15
5.4	constants.vhd	16
5.5	C:/fh/fpga/switch/switch.srcs/sources_1/new/switch.vhd-Dateireferenz	16
5.5.1	Ausführliche Beschreibung	16
5.6	switch.vhd	17
	Index	19

Kapitel 1

Design Unit Index

1.1 Design Unit Hierarchy

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

testbench	10
switch	7

Kapitel 2

Design Unit Index

2.1 Design Unit List

Here is a list of all design unit members with links to the Entities they belong to:

entity switch	
Switch entity Diese Entity implementiert den Switch. Diverse Parameter werden als Generics erst bei der Instanzierung angegeben	7
architecture switch	
Switch architecture In dieser Architecture wird der Switch implementiert	8
entity testbench	
Testbench entity	10
architecture testbench	
Testbench architecture Mit dieser Testbench wird der Switch getestet. Es werden alle Ausgänge einzeln getestet, sowie ein Broadcast. Ein Paket mit ungültiger Adresse wird ebenfalls getestet, sowie die Funktion des Switch nach einem solchen ungültigen Paket	11

Kapitel 3

Datei-Verzeichnis

3.1 Auflistung der Dateien

Hier folgt die Aufzählung aller dokumentierten Dateien mit einer Kurzbeschreibung:

C:/fh/fpga/switch/switch.srcs/sim_1/new/ testbench.vhd	
Switch Testbench	13
C:/fh/fpga/switch/switch.srcs/sources_1/new/ constants.vhd	
Switch Constants	15
C:/fh/fpga/switch/switch.srcs/sources_1/new/ switch.vhd	
Switch	16

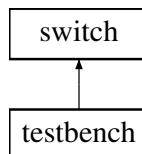
Kapitel 4

Klassen-Dokumentation

4.1 switch Entity Reference

Switch entity Diese Entity implementiert den Switch. Diverse Parameter werden als Generics erst bei der Instanzierung angegeben.

Klassendiagramm für switch:



Entities

- [switch](#) architecture

Switch architecture In dieser Architecture wird der Switch implementiert.

Libraries

- [IEEE](#)

Use standard library.

Use Clauses

- [STD_LOGIC_1164](#)

Use std_logic functions.

- [switch_constants](#)

Use constant definitions.

- [NUMERIC_STD](#)

Use numeric functions.

Generics

- **WIDTH integer:= 8**
Wortbreite (parallel)
- **NUM_OUTPUTS integer:= 4**
Anzahl der Ausgänge.
- **PKT_LEN integer:= 20**
Länge der Datenpakete.
- **PAUSE_LEN integer:= 10**
Länge der Pause zwischen Paketen.

Ports

- **clk in std_logic**
Takt.
- **input in std_logic_vector(WIDTH - 1 downto 0)**
Eingang.
- **outputs out std_logic_array (1 to NUM_OUTPUTS)**
Array von Ausgängen.

4.1.1 Ausführliche Beschreibung

Switch entity Diese Entity implementiert den Switch. Diverse Parameter werden als Generics erst bei der Instanzierung angegeben.

Definiert in Zeile 27 der Datei [switch.vhd](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- C:/fh/fpga/switch/switch.srscs/sources_1/new/[switch.vhd](#)

4.2 switch Architecture Reference

Switch architecture In dieser Architecture wird der Switch implementiert.

Processes

- **read_start(input , clk)**
read_start: speichert das erste Byte des Datenpakets als Zieladresse
- **forward(input , clk)**
warten auf neue Daten - Adresse ist erstes Byte, danach wird listen auf false gesetzt

Constants

- **MAX integer:=PKT_LEN +PAUSE_LEN - 1**
- **ADDR_MAX integer:= 2 **WIDTH - 1**
Broadcast-Adresse.

Signals

- `finished boolean:=false`
Datenpaket komplett übertragen.

Shared Variables

- `address sharedintegerrange 0 to((2**input 'length)- 1)`
Adresse - mögliche Werte: 0 to 255 (für WIDTH=8)

4.2.1 Ausführliche Beschreibung

Switch architecture In dieser Architecture wird der Switch implementiert.

Definiert in Zeile 44 der Datei `switch.vhd`.

4.2.2 Dokumentation der Datenelemente

4.2.2.1 MAX

```
MAX integer:=PKT_LEN +PAUSE_LEN - 1 [Constant]
```

Länge von Datenpaket + Pause - 1: 20 + 10 - 1 = 29 Bytes 1 Takt Abzug wegen Handshake über signal finished

Definiert in Zeile 49 der Datei `switch.vhd`.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- `C:/fh/fpga/switch/switch.srscs/sources_1/new/switch.vhd`

4.3 switch_constants Package Reference

Libraries

- `IEEE`

Use Clauses

- `STD_LOGIC_1164`

Constants

- **NUM_OUTPUTS integer:= 4**
Anzahl der Ausgänge.
- **WIDTH integer:= 8**
Wortbreite.
- **PKT_LEN integer:= 20**
Länge der Datenpakete.
- **PAUSE_LEN integer:= 10**
Länge der Pause.

Types

- **std_logic_arrayarray(positive range <>) of std_logic_vector(WIDTH - 1 downto 0)**
Array von std_logic_vector.

4.3.1 Ausführliche Beschreibung

Definiert in Zeile 13 der Datei [constants.vhd](#).

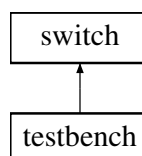
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- [C:/fh/fpga/switch/switch.srsc/sources_1/new/constants.vhd](#)

4.4 testbench Entity Reference

Testbench entity.

Klassendiagramm für testbench:



Entities

- **testbench architecture**
Testbench architecture Mit dieser Testbench wird der Switch getestet. Es werden alle Ausgänge einzeln getestet, sowie ein Broadcast. Ein Paket mit ungültiger Adresse wird ebenfalls getestet, sowie die Funktion des Switch nach einem solchen ungültigen Paket.

Libraries

- **IEEE**
Use standard library.

Use Clauses

- [STD_LOGIC_1164](#)
Use std_logic functions.
- [STD_LOGIC_UNSIGNED](#)
Use unsigned logic functions.
- [numeric_std](#)
Use numeric functions.
- [switch_constants](#)
Use constant definitions.

4.4.1 Ausführliche Beschreibung

Testbench entity.

Definiert in Zeile 22 der Datei [testbench.vhd](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- C:/fh/fpga/switch/switch.srscs/sim_1/new/[testbench.vhd](#)

4.5 testbench Architecture Reference

Testbench architecture Mit dieser Testbench wird der Switch getestet. Es werden alle Ausgänge einzeln getestet, sowie ein Broadcast. Ein Paket mit ungültiger Adresse wird ebenfalls getestet, sowie die Funktion des Switch nach einem solchen ungültigen Paket.

Processes

- [osc\(\)](#)
osc: Taktgenerator
- [Waveforms\(\)](#)
Waveforms: Generiert Testinputs.

Constants

- **T time := 10 ns**
Zeitkonstante.

Signals

- [clk std_logic](#)
Takt.
- [test_in std_logic_vector\(WIDTH - 1 downto 0 \)](#)
Eingang.
- [test_out std_logic_array \(1 to NUM_OUTPUTS \)](#)
Ausgänge.
- [out1 std_logic_vector\(WIDTH - 1 downto 0 \) := test_out \(1 \)](#)
- [out2 std_logic_vector\(WIDTH - 1 downto 0 \) := test_out \(2 \)](#)
- [out3 std_logic_vector\(WIDTH - 1 downto 0 \) := test_out \(3 \)](#)
- [out4 std_logic_vector\(WIDTH - 1 downto 0 \) := test_out \(4 \)](#)

Instantiations

- [sw](#) switch

4.5.1 Ausführliche Beschreibung

Testbench architecture Mit dieser Testbench wird der Switch getestet. Es werden alle Ausgänge einzeln getestet, sowie ein Broadcast. Ein Paket mit ungültiger Adresse wird ebenfalls getestet, sowie die Funktion des Switch nach einem solchen ungültigen Paket.

Definiert in Zeile [32](#) der Datei [testbench.vhd](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- [C:/fh/fpga/switch/switch.srcs/sim_1/new/testbench.vhd](#)

Kapitel 5

Datei-Dokumentation

5.1 C:/fh/fpga/switch/switch.srscs/sim_1/new/testbench.vhd-Dateireferenz

Switch Testbench.

Entities

- [testbench](#) entity

Testbench entity.

- [testbench](#) architecture

Testbench architecture Mit dieser Testbench wird der Switch getestet. Es werden alle Ausgänge einzeln getestet, sowie ein Broadcast. Ein Paket mit ungültiger Adresse wird ebenfalls getestet, sowie die Funktion des Switch nach einem solchen ungültigen Paket.

5.1.1 Ausführliche Beschreibung

Switch Testbench.

Autor

Bernd Wacke
Oliver Hanser

This file defines a testbench for the switch entity.

Definiert in Datei [testbench.vhd](#).

5.2 testbench.vhd

```

00001 -----
00007 -----
00008
00010 library IEEE;
00012 use IEEE.STD_LOGIC_1164.ALL;
00014 use IEEE.STD_LOGIC_UNSIGNED.all;
00016 use ieee.numeric_std.all;
00017
00019 use work.switch_constants.all;
00020
00022 entity testbench is
00023 -- Port ( );
00024 end testbench;
00025
00027
00032 architecture testbench of testbench is
00033     constant T: time := 10 ns;
00034
00035     signal clk: std_logic;
00036     signal test_in: std_logic_vector(WIDTH-1 downto 0);
00037     signal test_out: std_logic_array(1 to NUM_OUTPUTS);
00038
00039     signal out1: std_logic_vector(WIDTH-1 downto 0) := test_out(1);
00040     signal out2: std_logic_vector(WIDTH-1 downto 0) := test_out(2);
00041     signal out3: std_logic_vector(WIDTH-1 downto 0) := test_out(3);
00042     signal out4: std_logic_vector(WIDTH-1 downto 0) := test_out(4);
00043
00044 begin
00045
00046     out1 <= test_out(1);
00047     out2 <= test_out(2);
00048     out3 <= test_out(3);
00049     out4 <= test_out(4);
00050
00051     sw: entity work.switch -- statt component-declaration; siehe
00052         --
00053         http://insights.sigasi.com/tech/four-and-half-ways-write-vhdl-instantiations.html
00054     generic map(
00055         WIDTH => WIDTH,
00056         NUM_OUTPUTS => NUM_OUTPUTS,
00057         PKT_LEN => PKT_LEN,
00058         PAUSE_LEN => PAUSE_LEN
00059     )
00060     port map(
00061         clk => clk,
00062         input => test_in,
00063         outputs => test_out
00064     );
00065
00066     osc: process
00067     begin
00068         clk <= '1';
00069         wait for T/2;
00070         clk <= '0';
00071         wait for T/2;
00072     end process osc;
00073
00075     Waveforms: process
00076     begin
00077         -- Daten versetzt zur clock bereitstellen, so dass bei steigender Flanke immer sichere Daten
00078         anliegen: -- 0 - 1/2 T: Clock_high; 1/2 T bis T: Clock_low
00079             wait for 3*T/4; -- in der Mitte der Clock_low Phase
00080
00082             test_in <= x"00";
00083             wait for T;
00084
00086             for addr in 1 to 4 loop
00087                 test_in <= std_logic_vector(to_unsigned(addr, test_in'length)); -- adresse,
00088                 1. Byte
00089                 wait for T;
00090                 for i in 0 to 19 loop -- payload, 20 Bytes
00091                     test_in <= std_logic_vector(to_unsigned(255-i, 8));
00092                     wait for T;
00093                 end loop;
00094                 for i in 1 to 10 loop -- 10 Pausen-Bytes
00095                     test_in <= x"00";
00096                     wait for T;
00097                 end loop;
00098             end loop;
00099
00100             test_in <= x"FF"; -- adresse, 1. Byte
00101             wait for T;
00102             for i in 0 to 19 loop -- payload, 20 Bytes

```

```

00103         test_in <= std_logic_vector(to_unsigned(255-i, 8));
00104         wait for T;
00105     end loop;
00106     for i in 1 to 10 loop -- 10 Pausen-Bytes
00107         test_in <= x"00";
00108         wait for T;
00109     end loop;
00110
00112     test_in <= x"05"; -- adresse, 1. Byte
00113     wait for T;
00114     for i in 0 to 19 loop -- payload, 20 Bytes
00115         test_in <= std_logic_vector(to_unsigned(255-i, 8));
00116         wait for T;
00117     end loop;
00118     for i in 1 to 10 loop -- 10 Pausen-Bytes
00119         test_in <= x"00";
00120         wait for T;
00121     end loop;
00122
00124     test_in <= x"02"; -- adresse, 1. Byte
00125     wait for T;
00126     for i in 0 to 19 loop -- payload, 20 Bytes
00127         test_in <= std_logic_vector(to_unsigned(255-i, 8));
00128         wait for T;
00129     end loop;
00130     for i in 1 to 10 loop -- 10 Pausen-Bytes
00131         test_in <= x"00";
00132         wait for T;
00133     end loop;
00134
00135     -- Auslauf...
00136     wait for (2**WIDTH + 10) * T;
00137     -- und ende!
00138     assert false
00139     report "test finished" severity failure;
00140 end process Waveforms;
00141
00142 end testbench;

```

5.3 C:/fh/fpga/switch/switch.srsrcs/sources_1/new/constants.vhd-Dateireferenz

Switch Constants.

Entities

- [switch_constants](#) package

5.3.1 Ausführliche Beschreibung

Switch Constants.

Autor

Bernd Wacke
Oliver Hanser

This file defines constants for the Switch project.

Definiert in Datei [constants.vhd](#).

5.4 constants.vhd

```
00001 -----
00007 -----
00008
00009
00010 library IEEE;
00011 use IEEE.STD_LOGIC_1164.ALL;
00012
00013 package switch_constants is
00014
00015     constant NUM_OUTPUTS: integer := 4;
00016     constant WIDTH: integer := 8;
00017
00018     constant PKT_LEN: integer := 20;
00019     constant PAUSE_LEN: integer := 10;
00020
00022     type std_logic_array is array(positive range <>) of std_logic_vector(
        WIDTH-1 downto 0);
00023
00024 end switch_constants;
```

5.5 C:/fh/fpga/switch/switch.srcs/sources_1/new/switch.vhd-Dateireferenz

Switch.

Entities

- [switch](#) entity

Switch entity Diese Entity implementiert den Switch. Diverse Parameter werden als Generics erst bei der Instanzierung angegeben.

- [switch](#) architecture

Switch architecture In dieser Architecture wird der Switch implementiert.

5.5.1 Ausführliche Beschreibung

Switch.

Autor

Bernd Wacke
Oliver Hanser

This file defines the switch entity.

Definiert in Datei [switch.vhd](#).

5.6 switch.vhd

```

00001 -----
00007 -----
00008
00009
00011 library IEEE;
00012
00014 use IEEE.STD_LOGIC_1164.ALL;
00015
00017 use work.switch_constants.ALL;
00018
00020 use IEEE.NUMERIC_STD.ALL;
00021
00022
00024
00027 entity switch is
00028     generic (
00029         WIDTH: integer := 8;
00030         NUM_OUTPUTS: integer := 4;
00031         PKT_LEN: integer := 20;
00032         PAUSE_LEN: integer := 10
00033     );
00034     Port (
00035         signal clk: in std_logic;
00036         signal input: in std_logic_vector(WIDTH-1 downto 0);
00037         signal outputs: out std_logic_array(1 to NUM_OUTPUTS)
00038     );
00039 end switch;
00040
00042
00044 architecture switch of switch is
00046     shared variable address: integer range 0 to ((2 ** input'length) - 1);
00049     constant MAX: integer := PKT_LEN + PAUSE_LEN - 1;
00051     constant ADDR_MAX: integer := 2**WIDTH - 1;
00053     signal finished: boolean := false;
00054 begin
00055
00057     read_start: process(input, clk)
00059         variable temp_addr: integer range 0 to ((2 ** input'length) - 1);
00061         variable last_input: std_logic_vector(input'range);
00063         variable zeros: std_logic_vector(input'range) := (others => '0');
00065         variable listen: boolean := true;
00066     begin
00067         if rising_edge(clk) then
00068             if input /= zeros and last_input = zeros and listen then
00069                 temp_addr := to_integer(unsigned(input));
00070                 if temp_addr <= NUM_OUTPUTS or temp_addr = ADDR_MAX then
00071                     address := temp_addr;
00072                     listen := false;
00073                 end if;
00074             elsif finished then
00075                 listen := true;
00076                 address := 0;
00077             end if;
00078             last_input := input;
00079         end if;
00080     end process read_start;
00081
00086     forward: process(input, clk)
00088         variable remaining_bytes: integer range 0 to MAX := MAX;
00089     begin
00090         if clk'event and clk = '1' then
00091             for i in 1 to NUM_OUTPUTS loop
00092                 outputs(i) <= (others => '0');
00093             end loop;
00094             if address >= 1 and address <= NUM_OUTPUTS then
00095                 finished <= false;
00096                 outputs(address) <= input;
00097             elsif address = ADDR_MAX then
00098                 finished <= false;
00099                 for i in 1 to NUM_OUTPUTS loop
00100                     outputs(i) <= input;
00101                 end loop;
00102             end if;
00103             if address > 0 then
00104                 remaining_bytes := remaining_bytes - 1;
00105             end if;
00106             if remaining_bytes = 0 then
00107                 finished <= true;
00108                 remaining_bytes := MAX;
00109             end if;
00110         end if;
00111     end process forward;
00112
00113 end switch;

```


Index

C:/fh/fpga/switch/switch.srscs/sim_1/new/testbench.vhd,
[13](#), [14](#)

C:/fh/fpga/switch/switch.srscs/sources_1/new/constants.↵
vhd, [15](#), [16](#)

C:/fh/fpga/switch/switch.srscs/sources_1/new/switch.vhd,
[16](#), [17](#)

MAX

switch::switch, [9](#)

switch, [7](#), [8](#)

switch::switch

MAX, [9](#)

switch_constants, [9](#)

testbench, [10](#), [11](#)