AGSTU Arbete Genom STUdier Utbildning

VHDL- programmering för inbyggda system
Välkommen!

- Std_logic
- Integer type
- Different standard types
- Type Conversion, ieee.numeric_std.all
- Conversion Functions
- Example Puls generation

AGSTU Arbete Genom STUdier Utbildning

Arithmetic operations

Synthesizable arithmetic operations:

- Addition, +
- Subtraction, -
- Comparisons, >, >=, <, <=
- Multiplication, *
- Division by a power of 2, /2**6 (equivalent to shift)
- Shifts by a constant, SHL, SHR

AGSTU Arbete Genom STUdier Utbildning

Unsigned and signed types

- Unsigned type
  - Value 0 to 2^n-1
- Signed type
  - Value $-2^{(n-1)}$ to $2^{(n-1)}$-1

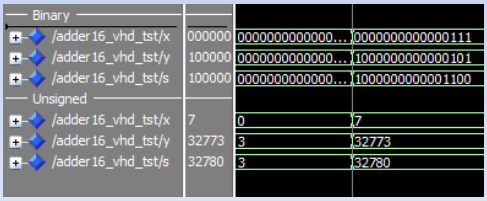
- Usage similar to std_logic_vector:

```
signal A_unsigned    : unsigned(3 downto 0) ;
signal B_signed      : signed  (3 downto 0) ;
signal C_slv         : std_logic_vector (3 downto 0) ;
. . .

A_unsigned <= "1111" ;      ← = 15 decimal

B_signed   <= "1111" ;      ← = -1 decimal

C_slv      <= "1111" ;      ← = 15 decimal only if using std_logic_unsigned
```

AGSTU Arbete Genom STUdier Utbildning

Addition of Unsigned Numbers

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.numeric_std.all ;

ENTITY adder16 IS
      PORT (   X, Y          : IN      UNSIGNED(15 DOWNTO 0) ;
               S             : OUT     UNSIGNED (15 DOWNTO 0) ) ;
END adder16 ;


ARCHITECTURE Behavior OF adder16 IS

BEGIN
      Sum <= X + Y;
END Behavior ;
```

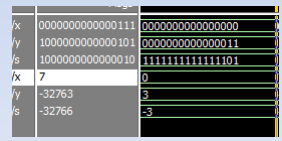


AGSTU Arbete Genom STUdier Utbildning

Addition of Signed Numbers

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.numeric_std.all ;

ENTITY adder16 IS
        PORT (    X, Y                    : IN        SIGNED(15 DOWNTO 0) ;
                  S                       : OUT       SIGNED(15 DOWNTO 0) ) ;
END adder16 ;
ARCHITECTURE Behavior OF adder16 IS
BEGIN
        S <= X + Y;
END Behavior ;
```

S <= **X + Y;**



Definition of std_logic

```
TYPE std_ulogic IS (
                'U',  -- Uninitialized
                'X',  -- Forcing  Unknown
                '0',  -- Forcing  0
                '1',  -- Forcing  1
                'Z',  -- High Impedance
                'W',  -- Weak     Unknown
                'L',  -- Weak     0
                'H',  -- Weak     1
                '-'   -- Don't care
               );
```

Std_logic a defacto industry standard,

Integer Types

Operations on signals (variables) of the integer types:

INTEGER, NATURAL,

and their sybtypes, such as

TYPE day_of_month IS **RANGE 1 TO 31;**

are synthesizable in the range

$-(2^{31}-1)$ .. 2^{31} -1 for INTEGERs and their subtypes

0 .. 2^{31} -1 for NATURALs and their subtypes

AGSTU Arbete Genom STUdier Utbildning

Typing motivation

- **Standard**
  - boolean true, false
  - character 191 / 256 characters
  - integer $-(2^{31}-1)$ to $(2^{31}-1)$
  - real -1.0E38 to 1.0E38
  - time 1 fs to 1 hr
  - Signed, unsigned
  - ….
- Strong Typing = Strong Error Checking Built into the Compiler
  - Less debugging

AGSTU Arbete Genom STUdier Utbildning

Type Conversion, ieee.numeric_std.all

- **Declarations**
  - signal A_unsigned_vector : unsigned (7 downto 0) ;
  - signal Unsigned_int : integer range 0 to 255 ;
  - signal B_signed_vector : signed(7 downto 0) ;
  - signal Signed_int : integer range -128 to 127;
- **Type Conversion**
  - Unsigned_int <= TO_INTEGER (A_unsigned_vector) ;
  - Signed_int <= TO_INTEGER (B_signed_vector) ;
  - A_unsigned_vector <= TO_UNSIGNED (Unsigned_int, 8) ;
  - B_signed_vector <= TO_SIGNED (Signed_int, 8) ;

AGSTU Arbete Genom STUdier Utbildning

Conversion Functions, ieee.numeric_std.all

```
function TO_INTEGER (ARG: UNSIGNED) return NATURAL;
  -- Result subtype: NATURAL. Value cannot be negative since parameter is an
  --          UNSIGNED vector.
  -- Result: Converts the UNSIGNED vector to an INTEGER.

function TO_INTEGER (ARG: SIGNED) return INTEGER;
  -- Result subtype: INTEGER
  -- Result: Converts a SIGNED vector to an INTEGER.

function TO_UNSIGNED (ARG, SIZE: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(SIZE-1 downto 0)
  -- Result: Converts a non-negative INTEGER to an UNSIGNED vector with
  --       the specified SIZE.

function TO_SIGNED (ARG: INTEGER; SIZE: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(SIZE-1 downto 0)
  -- Result: Converts an INTEGER to a SIGNED vector of the specified SIZE.
```

AGSTU Arbete Genom STUdier Utbildning

Conversion Functions, std_logic_arith

Old lib
The book
page 154

```
function CONV_INTEGER(ARG: INTEGER) return INTEGER;
function CONV_INTEGER(ARG: UNSIGNED) return INTEGER;
function CONV_INTEGER(ARG: SIGNED) return INTEGER;
function CONV_INTEGER(ARG: STD_ULOGIC) return SMALL_INT;

function CONV_UNSIGNED(ARG: INTEGER; SIZE: INTEGER) return UNSIGNED;
function CONV_UNSIGNED(ARG: UNSIGNED; SIZE: INTEGER) return UNSIGNED;
function CONV_UNSIGNED(ARG: SIGNED; SIZE: INTEGER) return UNSIGNED;
function CONV_UNSIGNED(ARG: STD_ULOGIC; SIZE: INTEGER) return UNSIGNED;

function CONV_SIGNED(ARG: INTEGER; SIZE: INTEGER) return SIGNED;
function CONV_SIGNED(ARG: UNSIGNED; SIZE: INTEGER) return SIGNED;
function CONV_SIGNED(ARG: SIGNED; SIZE: INTEGER) return SIGNED;
function CONV_SIGNED(ARG: STD_ULOGIC; SIZE: INTEGER) return SIGNED;

function CONV_STD_LOGIC_VECTOR(ARG: INTEGER; SIZE: INTEGER) return STD_LOGIC_VECTOR;
function CONV_STD_LOGIC_VECTOR(ARG: UNSIGNED; SIZE: INTEGER) return STD_LOGIC_VECTOR;
function CONV_STD_LOGIC_VECTOR(ARG: SIGNED; SIZE: INTEGER) return STD_LOGIC_VECTOR;
function CONV_STD_LOGIC_VECTOR(ARG: STD_ULOGIC; SIZE: INTEGER) return STD_LOGIC_VECTOR;
```

AGSTU Arbete Genom STUdier Utbildning

Conversion functions

Good to have that lazybones

		numeric_std	std_logic_arith
Type Conversion			
std_logic_vector	-> unsigned	unsigned(*arg*)	unsigned(*arg*)
std_logic_vector	-> signed	signed(*arg*)	signed(*arg*)
unsigned	-> std_logic_vector	std_logic_vector(*arg*)	std_logic_vector(*arg*)
signed	-> std_logic_vector	std_logic_vector(*arg*)	std_logic_vector(*arg*)
integer	-> unsigned	to_unsigned(*arg*, *size*)	conv_unsigned(*arg*, *size*)
integer	-> signed	to_signed(*arg*, *size*)	conv_signed(*arg*, *size*)
unsigned	-> integer	to_integer(*arg*)	conv_integer(*arg*)
signed	-> integer	to_integer(*arg*)	conv_integer(*arg*)
integer	-> std_logic_vector	integer -> unsigned/signed ->std_logic_vector	
std_logic_vector	-> integer	std_logic_vector -> unsigned/signed ->integer	
unsigned + unsigned	-> std_logic_vector	std_logic_vector(*arg1* + *arg2*)	*arg1* + *arg2*
signed + signed	-> std_logic_vector	std_logic_vector(*arg1* + *arg2*)	*arg1* + *arg2*
Resizing			
unsigned		resize(*arg*, *size*)	conv_unsigned(*arg*, *size*)
signed		resize(*arg*, *size*)	conv_signed(*arg*, *size*)

From: http://dz.ee.ethz.ch/support/ic/vhdl/vhdlsources.en.html

AGSTU Arbete Genom STUdier Utbildning

”Puls” - Example

Specification

-- Clock out pulses (LEDR[0]), every 40 clock cycles following frame.

```
-- -------------__---------------___________-----
-- 0          10              30        35   40
```



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

entity pulser is
port (
reset_n, CLOCK_50  : in std_logic; -- reset_n kopplas till SW[17]
LEDR  : out std_logic_vector(0 downto 0)
            );
end entity;
```

```
architecture rtl of pulser is
signal count_local : integer range 0 to 40;
begin
process (CLOCK_50, reset_n)
begin
        if reset_n = '0' then
                LEDR(0) <= '0';
        elsif (rising_edge(CLOCK_50)) then

        -- Clock out pulses (LEDR[0]), every 40 clock cycles following frame.
        -- -------------__---------------_____________-----
        -- 0        10          30      35   40

                if count_local >= 40 then
                        count_local <= 0;
                 else
                        count_local <= count_local + 1;
                end if;
-------------------------------------------------------------
                if      count_local = 10 or (count_local > 30 and count_local < 40) then
                        LEDR(0) <= '0';
                else
                        LEDR(0) <= '1';
                end if;
-------------------------------------------------
                end if;
        end process;
end rtl;
```



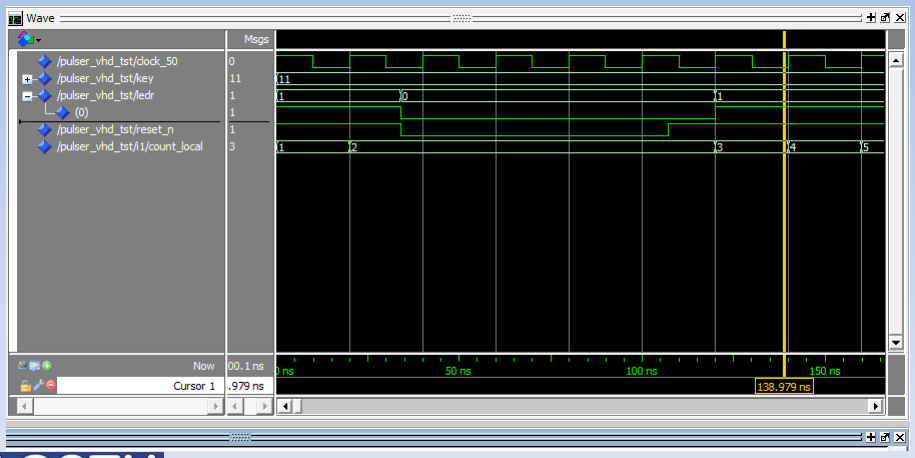
```
-----------------------------------

        reset_n <= '1', '0' AFTER 34 NS, '1' AFTER 107 NS;

----------------------------------------
init : PROCESS
-- variable declarations
BEGIN
    -- code that executes only once
WAIT;
END PROCESS init;
always : PROCESS

BEGIN
        for i in 1 to 200 loop
            CLOCK_50 <= not CLOCK_50;
                wait for 10 ns;
        end loop;

-- code executes for every event on sensitivity list
WAIT;
END PROCESS always;
-------------------------------
END pulser_arch;
```

Testbench

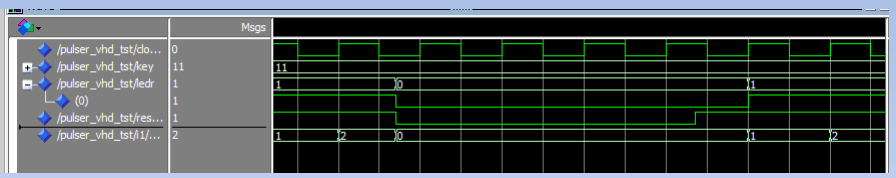
Reset

```
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
add wave \
{sim:/pulser_vhd_tst/i1/count_local }
VSIM 8> restart -force
VSIM 9> run
VSIM 10> run 1ms
```

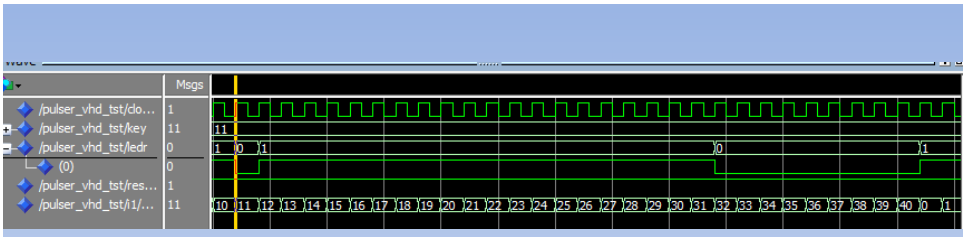
```
if reset_n = '0' then
        LEDR(0) <= '0';
elsif (rising_edge(CLOCK_50)) ..
```



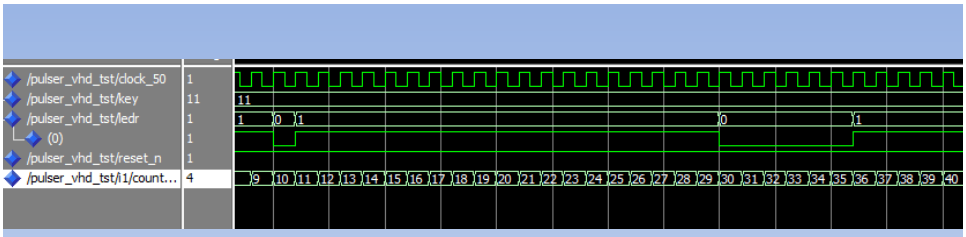
Reset




```
if reset_n = '0' then
        LEDR(0) <= '0';
        count_local <= 0;

elsif (rising_edge(CLOCK_50)) then
```




```
-- Clock out pulses (LEDR[0]), every 40 clock cycles following frame.
        -- -------------__----------------_____________-----
        -- 0            10             30        35   40

if  count_local = 10 or (count_local > 30 and count_local < 40) then
        LEDR(0) <= '0';
else
        LEDR(0) <= '1';
end if;
```


```
if count_local = 9 or (count_local >= 29 and count_local <= 34)
then
                LEDR(0) <= '0';
else
                LEDR(0) <= '1';
end if;
```

Unsigned - Unsigned = Signed?

- signal X_uv, Y_uv : unsigned (6 downto 0) ;
- signal Z_sv : signed (7 downto 0) ;
- . . .
- Z_sv <= signed('0' & X_uv) - signed('0' & Y_uv) ;

AGSTU Arbete Genom STUdier Utbildning