

VHDL Testbench



PREP #1 Example



VHDL Testbench



- NO Synthesis
- Test Vectors
 - ◆ Input Vector
 - ◆ Output Vector
 - ◆ Expected Vector
- UUT : Unit Under Test
- Clock Generator
- Error Report Utility



Library : STD



Library for Testbench

use std.standard.all; -- Char. & String type define

use std.textio.all; -- Text IO Buffer & File IO

Testbench Entity

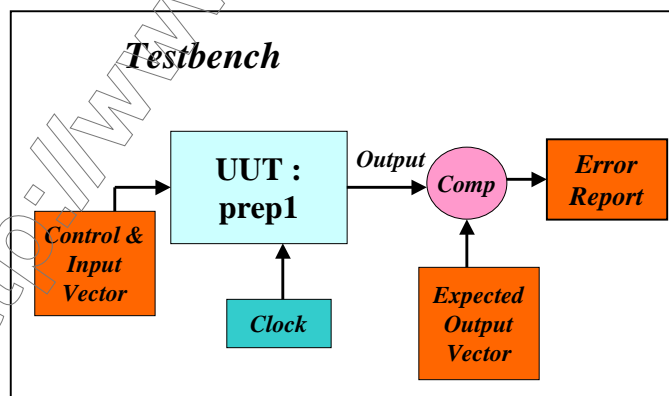
ENTITY *test_prep1* IS

-- Testbench has NOT need Port.....

END *test_prep1*;



Testbench Component





Testbench SIGNALs



ARCHITECTURE behave of test_prep1 IS

-- Control Signals

SIGNAL CLK: std_logic := '0'; *-- Clock Generator*

SIGNAL RST,S_L: std_logic; *-- Reset, Shift Left Control*

SIGNAL S: std_logic_VECTOR(1 DOWNTO 0); *-- Input Select*

-- Input/Output Signals of UUT

SIGNAL d0,d1,d2,d3: std_logic_VECTOR(7 DOWNTO 0); *-- Input to UUT*

SIGNAL Q: std_logic_VECTOR(7 DOWNTO 0); *-- Output from UUT*

-- Type define for Test Vector Container

CONSTANT numvecs: INTEGER := 20;

TYPE cntlrom is array(0 to (numvecs - 1)) of std_logic_VECTOR(3 DOWNTO 0);

TYPE datarom is array(0 to (numvecs - 1)) of std_logic_VECTOR(31 DOWNTO 0);

TYPE outrom is array(0 to (numvecs - 1)) of std_logic_VECTOR(7 DOWNTO 0);

-- Test Counter and Error Report

SIGNAL loopindex: integer := 0;

SIGNAL numerrors: integer := 0;



UUT Component



COMPONENT prep1

PORT (

CLK : IN std_logic;

RST : IN std_logic;

S_L : IN std_logic;

S : IN std_logic_VECTOR(1 DOWNTO 0);

d0,d1,d2,d3 : IN std_logic_VECTOR(7 DOWNTO 0);

Q : OUT std_logic_VECTOR(7 DOWNTO 0)

);

END COMPONENT;



Test Vector : Control Signal



CONSTANT cntl: cntlrom := -- RST,S_L, S(1:0)

```
(
  "1000", -- #0 Reset
  "1000", -- #1 Reset
  "1000", -- #2 Reset
  "0001", -- #3 Select d1
  "0010", -- #4 Select d2
  "0011", -- #5 Select d3
  "0011", -- #6 Select d3
  "0000", -- #7 Clocking
  "0011", -- #8 Select d3
  "0011", -- #9 Select d3
  "0111", -- #10 Select d3, Shift Left
  "0111", -- #11 Select d3, Shift Left
  "0111", -- #12 Select d3, Shift Left
  "0111", -- #13 Select d3, Shift Left
  "0111", -- #14 Select d3, Shift Left
  "0111", -- #15 Select d3, Shift Left
  "0111", -- #16 Select d3, Shift Left
  "0111", -- #17 Select d3, Shift Left
  "0111", -- #18 Select d3, Shift Left
  "1111" -- #19 Reset
);
```



Test Vector : Input



CONSTANT invec: datarom := -- Input Vector

```
(
  -- d3_d2_d1_d0
  "00000000000000000000000000000000", -- #0 00_00_00_00
  "11111100000000100000000000000000", -- #1 fe_01_00_00
  "00010001001000101010101011111111", -- #2 11_22_aa_ff
  "00010001001000101010101011111111", -- #3 11_22_aa_ff
  "00010001001000101010101011111111", -- #4 11_22_aa_ff
  "00010001001000101010101011111111", -- #5 11_22_aa_ff
  "11111110010001010101010111111111", -- #6 FF_22_aa_ff
  "000100010010001010101000000001", -- #7 11_22_aa_01
  "00000001000100011010101011111111", -- #8 01_11_aa_ff
  "11111110010001010101010111111111", -- #9 xx_22_aa_ff
  "00010001001000101010101011111111", -- #10 11_22_aa_ff
  "00010001001000101010101011111111", -- #11 11_22_aa_ff
  "00010001001000101010101011111111", -- #12 11_22_aa_ff
  "00010001001000101010101011111111", -- #13 11_22_aa_ff
  "00010001001000101010101011111111", -- #14 11_22_aa_ff
  "00010001001000101010101011111111", -- #15 11_22_aa_ff
  "00010001001000101010101011111111", -- #16 11_22_aa_ff
  "00010001001000101010101011111111", -- #17 11_22_aa_ff
  "00010001001000101010101011111111", -- #18 11_22_aa_ff
  "00010001001000101010101011111111" -- #19 11_22_aa_ff
);
```



Test Vector : Expected Output



```
CONSTANT outvec: outrom := -- Output Vector
(
  "UUUUUUUU", -- #0 Q=0x00 UNKNOWN
  "00000000", -- #1 Q=0x00 RST
  "00000000", -- #2 Q=0x00
  "00000000", -- #3 Q=0x00
  "10101010", -- #4 Q=0xAA invec(3)
  "00100010", -- #5 Q=0x22 invec(4)
  "00010001", -- #6 Q=0x11 invec(5)
  "11111111", -- #7 Q=0xFF invec(6)
  "00000001", -- #8 Q=0x01 invec(7)
  "00000001", -- #9 Q=0x01 invec(8) comes out
  "00000010", -- #10 Q=0x02 rotate
  "00000100", -- #11 Q=0x04 rotate
  "00001000", -- #12 Q=0x08 rotate
  "00010000", -- #13 Q=0x10 rotate
  "00100000", -- #14 Q=0x20 rotate
  "01000000", -- #15 Q=0x40 rotate
  "10000000", -- #16 Q=0x80 rotate
  "00000001", -- #17 Q=0x01 rotate
  "00000010", -- #18 Q=0x02 rotate
  "00000000" -- #19 Q=0x00 Reset all bits
);
```



UUT Instanciate



```
BEGIN
  I1: prep1 -- UUT "prep1"
  port map ( -- Map UUT Ports to Testbench's Signals
    CLK => CLK,
    RST => RST,
    S_L => S_L,
    S => S,
    d0 => d0,
    d1 => d1,
    d2 => d2,
    d3 => d3,
    Q => Q
  );
```



Clock Generator



```
toggleclk: PROCESS (CLK)
VARIABLE I: LINE;
BEGIN
  IF loopindex >= numvecs THEN
    -- go to finish routine
    qatestdone(numerrors); -- Error Reporting Utility
  ELSE
    CLK <= NOT CLK AFTER 50 ns; -- Toggling Clock
    -- CLK initialized when declared
  END IF;
END PROCESS toggleclk;
```



Test Jig



```
testjig: PROCESS (CLK)
BEGIN
  IF CLK = '0' THEN -- UUT was RISING-Triggered, Output Check when Clock Falling
    IF Q /= outvec(loopindex) THEN -- Compare UUT output Q with Expected output vector
      numerrors <= numerrors + 1; -- If Error
      qafail(outvec(loopindex), q, "UUT out Q"); -- Display an error message
    END IF;
    loopindex <= loopindex + 1; -- scheduled, so bump in next assignments
  IF (loopindex + 1) < numvecs THEN -- Assign Next Test Data from Test Vectors
    RST <= cntl(loopindex + 1)(3); -- Control Vector ROM
    S_L <= cntl(loopindex + 1)(2);
    S <= cntl(loopindex + 1)(1 DOWNT0 0);
    d3 <= invec(loopindex + 1)(31 DOWNT0 24); -- Input Vector ROM
    d2 <= invec(loopindex + 1)(23 DOWNT0 16);
    d1 <= invec(loopindex + 1)(15 DOWNT0 8);
    d0 <= invec(loopindex + 1)(7 DOWNT0 0);
  END IF;
  ELSIF CLK = '1' THEN -- Nothing to do..
  ELSE -- CLK is unknown
  END IF;
END PROCESS testjig;
END behave;
```



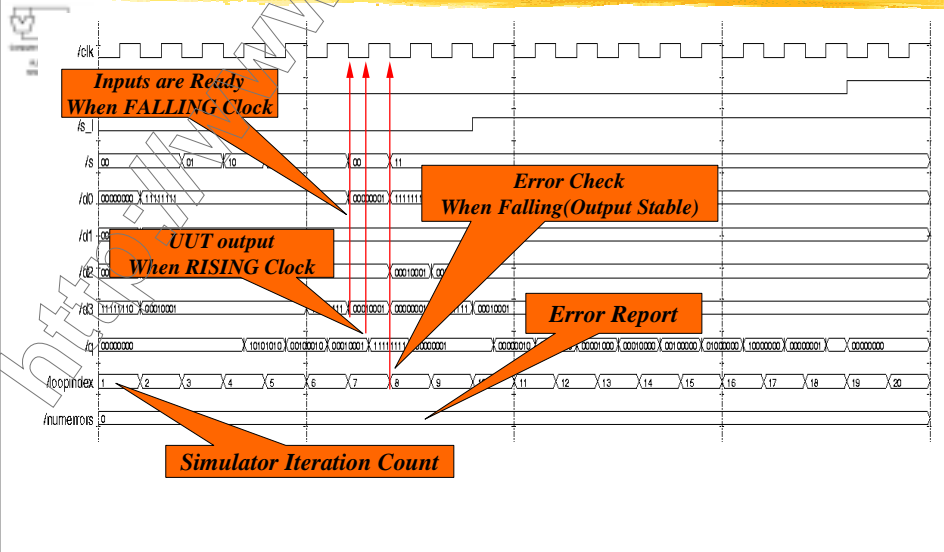
Simulation Procedure (MTI)



- **Make Working library**
 - ◆ vlib work
- **Compile UUT**
 - ◆ vcom UUT
- **Compile Testbench**
 - ◆ vcom testbench
- **Simulate testbench**
 - ◆ vsim testbench
- **Simulate**
 - ◆ run



Simulation Result





Error Reporting Utility



```
LIBRARY std, IEEE;  
USE ieee.std_logic_1164.all;  
USE std.standard.all;  
USE std.textio.all;
```

```
PACKAGE qatools IS
```

```
    type stdlogic_to_char_t is array(std_logic) of character; -- Char, Table
```

```
    PROCEDURE qatestdone (numerrors: IN INTEGER);
```

```
    PROCEDURE qafail (expected,actual: IN std_logic; what:IN string);
```

```
    PROCEDURE qafail (expected,actual: IN std_logic_vector; what:IN string);
```

```
    FUNCTION to_string(inp : std_logic_vector) return string;
```

```
END qatools;
```



Error Reporting Utility



```
PACKAGE BODY qatools IS
```

```
    PROCEDURE qatestdone(numerrors: IN INTEGER) IS
```

```
        VARIABLE I: LINE; -- Text IO Buffer
```

```
    BEGIN
```

```
        IF (numerrors = 0) THEN -- Was Error ?
```

```
            WRITE(I,string("End of Good Simulation!")); -- Write String
```

```
        ELSE
```

```
            -- into Buffer
```

```
            WRITE(I,string("Error: Simulation failed with "));
```

```
            WRITE(I,numerrors);
```

```
            WRITE(I,string(" errors. "));
```

```
        END IF;
```

```
        WRITELINE(OUTPUT,I); -- String Output into the Transcript Window
```

```
    END qatestdone;
```




Error Reporting Utility



```
constant to_char : std_logic_to_char_t := (  
    'U' => 'U',  
    'X' => 'X',  
    '0' => '0',  
    '1' => '1',  
    'Z' => 'Z',  
    'W' => 'W',  
    'L' => 'L',  
    'H' => 'H',  
    '.' => '.');  
  
--  
-- convert a std_logic_vector to a string  
--  
function to_string(inp : std_logic_vector) return string is  
    alias vec : std_logic_vector(1 to inp'length) is inp;  
    variable result : string(vec'range);  
  
begin  
    for i in vec'range loop  
        result(i) := to_char(vec(i));  
    end loop;  
    return result;  
end;
```



Error Reporting Utility



```
PROCEDURE qafail (expected,actual: IN std_logic; what : IN string ) IS  
BEGIN  
    assert false  
        report "Failure for " & what & ",Expected: " & to_char(expected)  
            & " Actual: " & to_char(actual)  
        severity error;  
END qafail;  
  
PROCEDURE qafail (expected,actual: IN std_logic_vector; what:IN string) IS  
BEGIN  
    assert false  
        report "Failure for " & what & ",Expected: " & to_string(expected)  
            & " Actual: " & to_string(actual)  
        severity error;  
END qafail;  
  
END qatools;
```



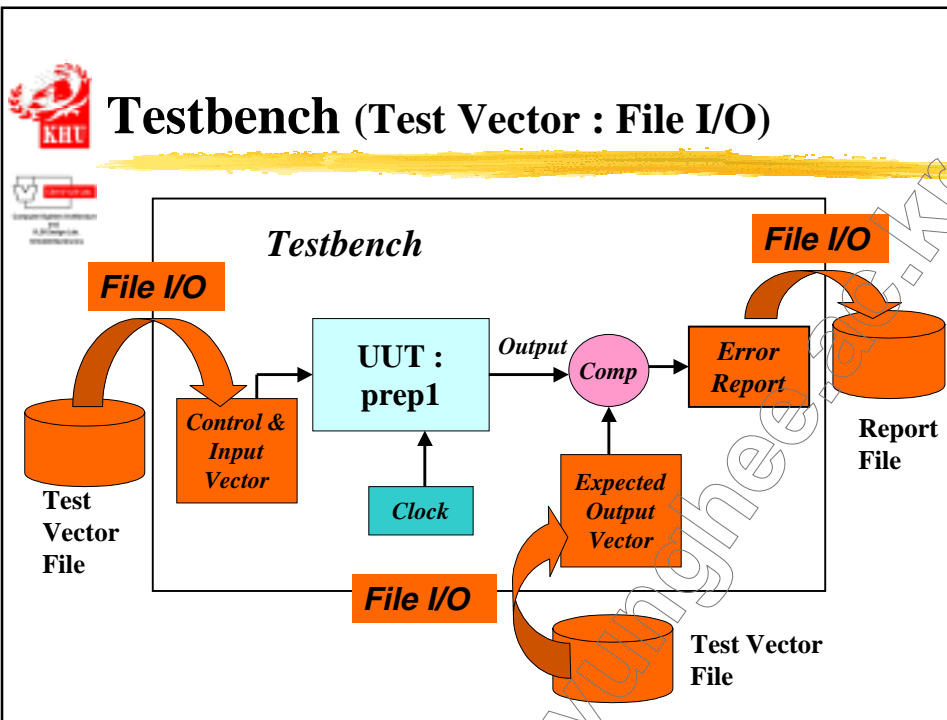
Error Reporting (Transcript-Success Message)

Simulation Success Report on Transcript Window



Error Reporting (Transcript-Fail Message)

Error Reporting on Transcript Window



File IO

- **LIBRARY**

```
USE ieee.std_logic_1164.ALL;
USE std.textio.ALL;
```
- **File Declaration**

```
FILE file_id : sub_type IS [IN|OUT] "filename";
```
- **File IO Function**

```
readline(file_id, line_buff), writeline(file_id, line_buff),
read(line_buff, var, bool), write(line_buff, var, bool);
```
- **Helper Function**

```
endfile(file_id)
endline(line_buff)
```



Example : Test Vector File



■ Test Vector File

```

1000 00000000000000000000000000000000 UUUUUUUU
1000 11111110000000010000000000000000 00000000
1000 00010001001000101010101011111111 00000000
0001 00010001001000101010101011111111 00000000
0000 00010001001000101 .....
0011 000000010001000 .....
0011 111111110010 .....
.....

```

■ Read String

```
"1000 11111110000000010000000000000000 00000000"
```

■ Convert to 44-bit Std_logic_vector

```
| rst(1) | s(1) | s_l(2) | d0(8) | d1(8) | d2(8) | d3(8) | outvect(8) |
= 44 bits
```



Testbench : SIGNAL ALIAS



```

SIGNAL loopindex : integer := 0;
SIGNAL numerrors : integer := 0;
SIGNAL CLK : std_logic := '0';
SIGNAL Q : std_logic_VECTOR(7 DOWNT0 0);
-- SIGNALs for Test Vector Handling
SIGNAL ports : std_logic_vector(44 DOWNT0 1) := (OTHERS => 'Z');
-- Alias signals easy to connect to the component
ALIAS RST : std_logic IS ports(44);
ALIAS S_L : std_logic IS ports(43);
ALIAS S : std_logic_vector(1 downto 0) IS ports(42 downto 41);
ALIAS d0 : std_logic_vector(7 DOWNT0 0) IS ports(40 downto 33);
ALIAS d1 : std_logic_vector(7 DOWNT0 0) IS ports(32 downto 25);
ALIAS d2 : std_logic_vector(7 DOWNT0 0) IS ports(24 downto 17);
ALIAS d3 : std_logic_vector(7 DOWNT0 0) IS ports(16 downto 9);
ALIAS outvect : std_logic_VECTOR(7 DOWNT0 0) IS ports(8 downto 1);

```



Testbench : Test Vector File I/O



```
testjig: PROCESS (CLK)
FILE vector_file : text IS IN "testvect.dat"; -- Test Vector File Declare
VARIABLE l      : line; -- Line Buffer
VARIABLE signo  : integer;
VARIABLE good_number : boolean := true;
BEGIN
  IF CLK = '0' THEN
    IF endfile(vector_file) THEN -- EOF ?
      qatstdone(numerrors); -- go to finish routine
    ELSE
      -- Read string from test vector file
      -- and store into LINE buffer ; "l"
      readline(vector_file, l);
```



Testbench : Convert to std_logic



```
IF good_number THEN
  signo := 44; -- Test Vector Length
  FOR i IN 1'RANGE LOOP
    CASE l(i) IS
      WHEN '0' => -- Drive 0
        ports(signo) <= '0';
      WHEN '1' => -- Drive 1
        ports(signo) <= '1';
      WHEN 'h' | 'H' => -- Test for 1
        ASSERT ports(signo) = '1';
      WHEN 'l' | 'L' => -- Test for 0
        ASSERT ports(signo) = '0';
      WHEN 'x' | 'X' => -- Don't care
        NULL;
      WHEN 'u' | 'U' => -- Unknown
        NULL;
```



Testbench : Feed Test vector



```
WHEN ' ' | ht => -- Skip white space
  NEXT;
WHEN OTHERS => -- Illegal character
  ASSERT false
  REPORT "Illegal character in vector file: " & I(i);
  EXIT;
END CASE;
signo := signo - 1;
END LOOP;
ELSE
  -- Compare UUT output with Output expected
  IF Q /= outvect THEN
    numerrors <= numerrors + 1;
    qafail(outvect, q, "UUT out Q"); -- display an error message
  END IF;
  loopindex <= loopindex + 1;
END IF;
END IF;
```

<http://www.csvlsi.kyunghee.ac.kr>