Functional Verification

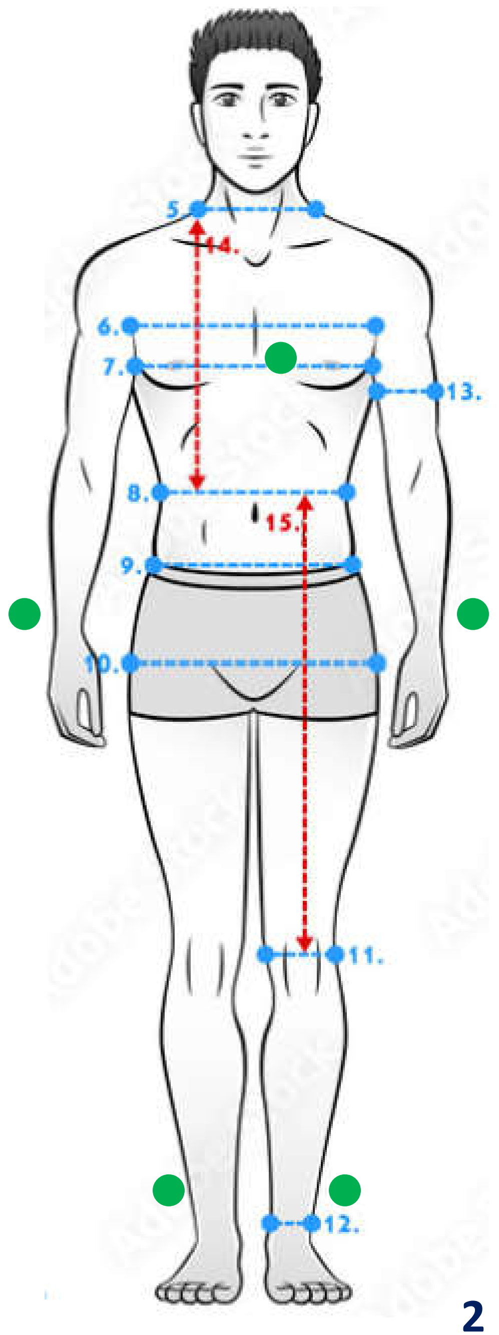
Tuan Nguyen-viet

DUT (full_adder)

```
module full_adder (a_in, b_in, c_in, c_out, sum_out);

input [3:0] a_in, b_in;
input c_in;
output carry_out;
output [3:0] sum_out;

reg carry_out;
reg [3:0] sum_out;

always @ (a_in or b_in or c_in)
    begin
        {carry_out, sum_out} = a_in + b_in + c_in;
    end

endmodule
```

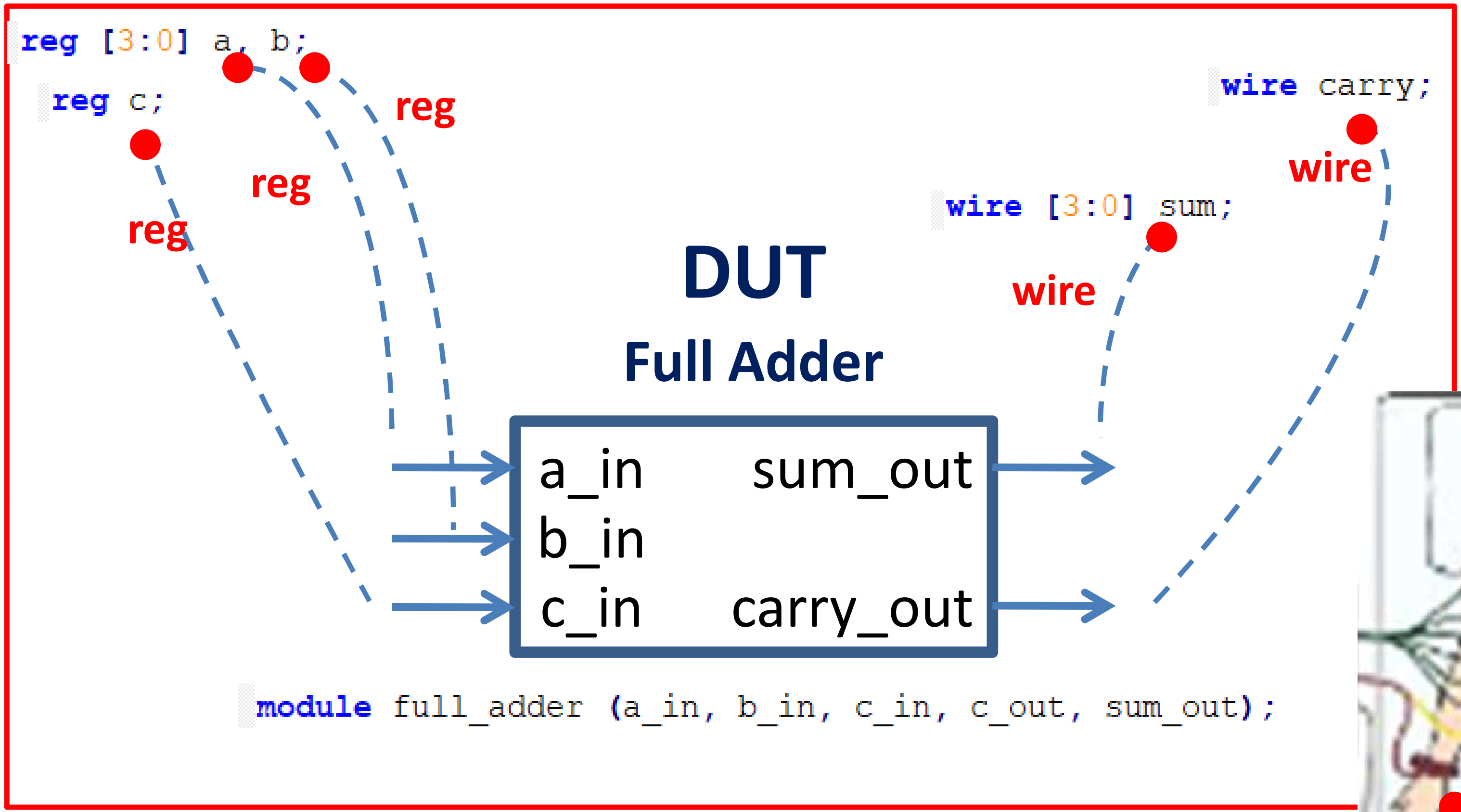
Blank/Empty

```
full_adder dut (
                .a_in(    ),
                .b_in(    ),
                .c_in(    ),
                .carry_out(    ),
                .sum_out(    )
                );
```

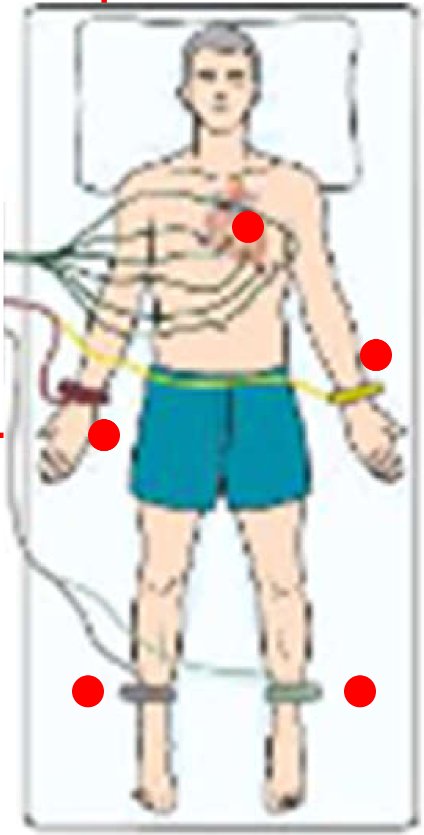


Test Bench

```
module full_adder_test; // 4-bit
```



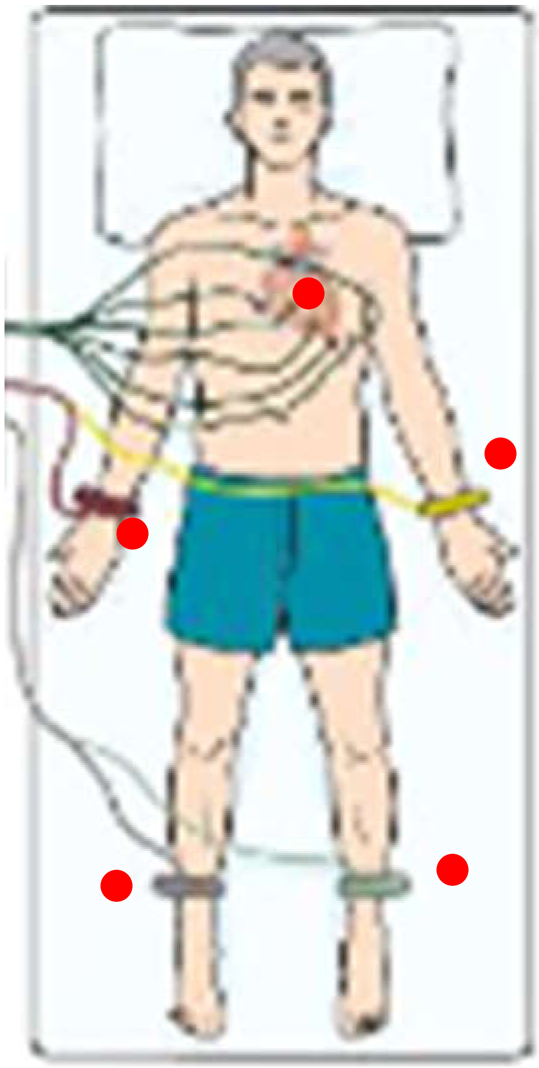
```
reg [3:0] a, b;
reg c;
wire carry;
wire [3:0] sum;
module full_adder (a_in, b_in, c_in, c_out, sum_out);
```



Test Bench (full_adder_test)

```
`timescale 1 ns/10 ps

module full_adder_test; // 4-bit

reg [3:0] a, b;
reg c;
wire [3:0] sum;
wire carry;

full_adder dut (
                    .a_in(a),
                    .b_in(b),
                    .c_in(c),
                    .carry_out(carry),
                    .sum_out(sum)
                    );
initial
    begin
        a = 4'b0000;
        b = 4'b0000;
        c = 1'b0;
        #50;
        a = 4'b0101;
        b = 4'b1010;
        // => sum=1111, carry=0
        #50;
        a = 4'b1111;
        b = 4'b0001;
        // => sum=0000, carry=1
        #50;
        a = 4'b0000;
        b = 4'b1111;
        c = 1'b1;
        // => sum=0000, carry=1
        #50;
        a = 4'b0110;
        b = 4'b0001;
        // => sum=1000, carry=0
    end
endmodule
```

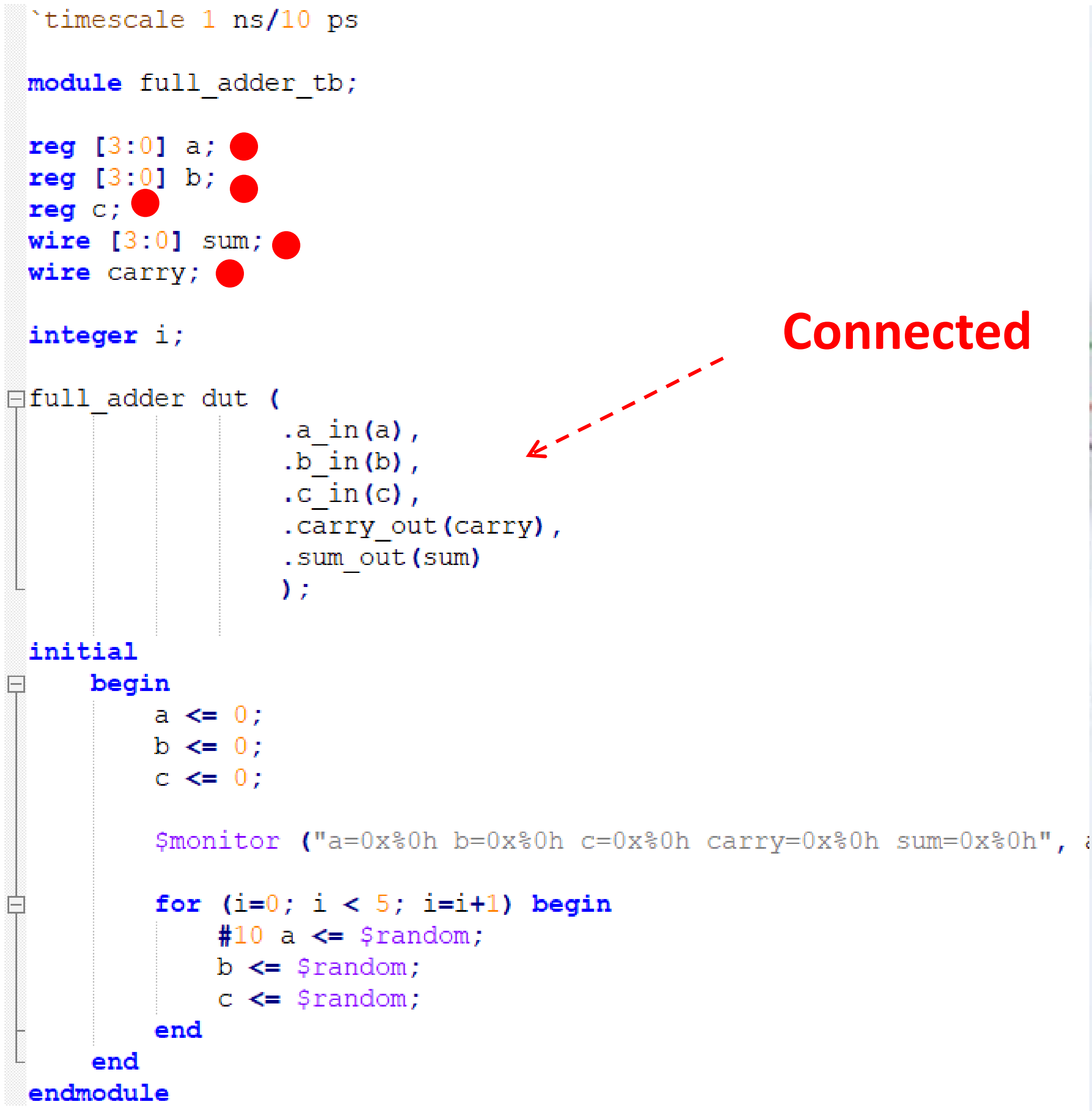
Connected



Blank/Empty

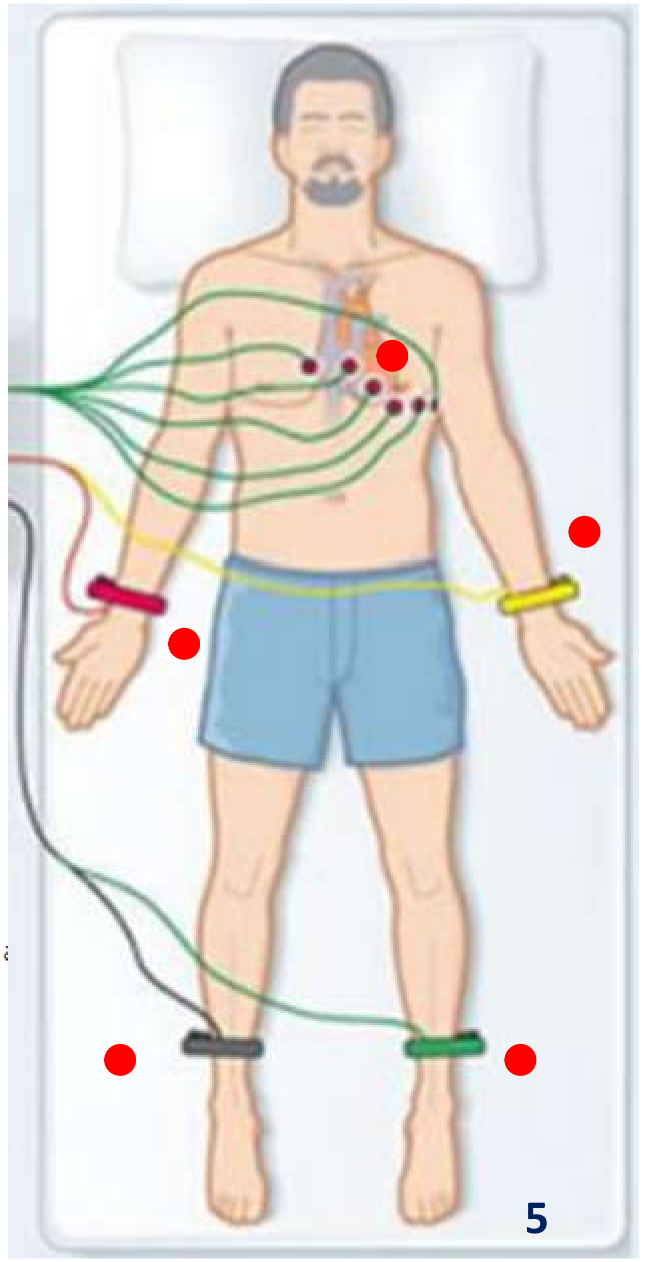
```
full_adder dut (
                    .a_in(   ),
                    .b_in(   ),
                    .c_in(   ),
                    .carry_out(   ),
                    .sum_out(   )
                    );
```

Test Bench (full_adder_tb)

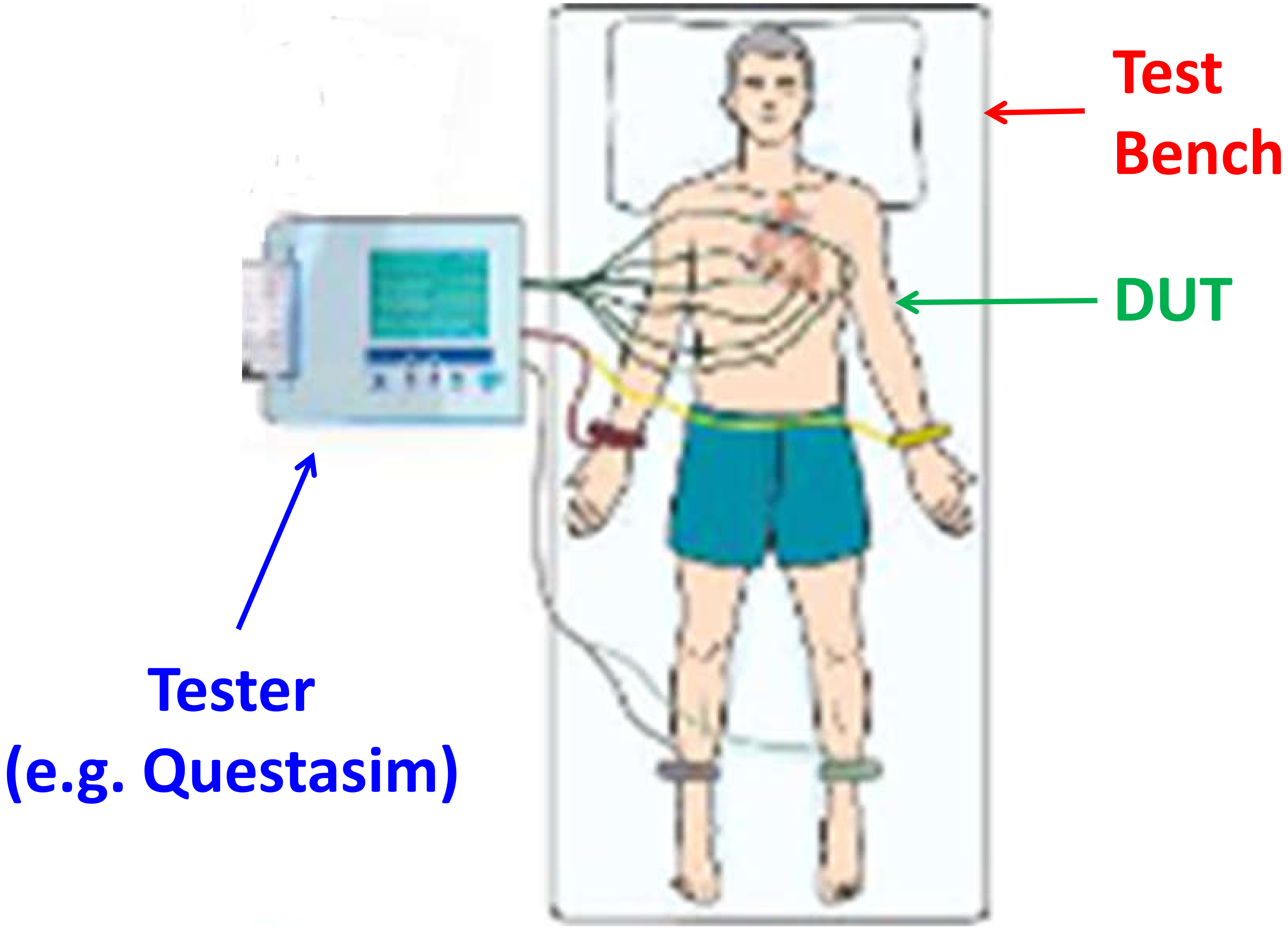


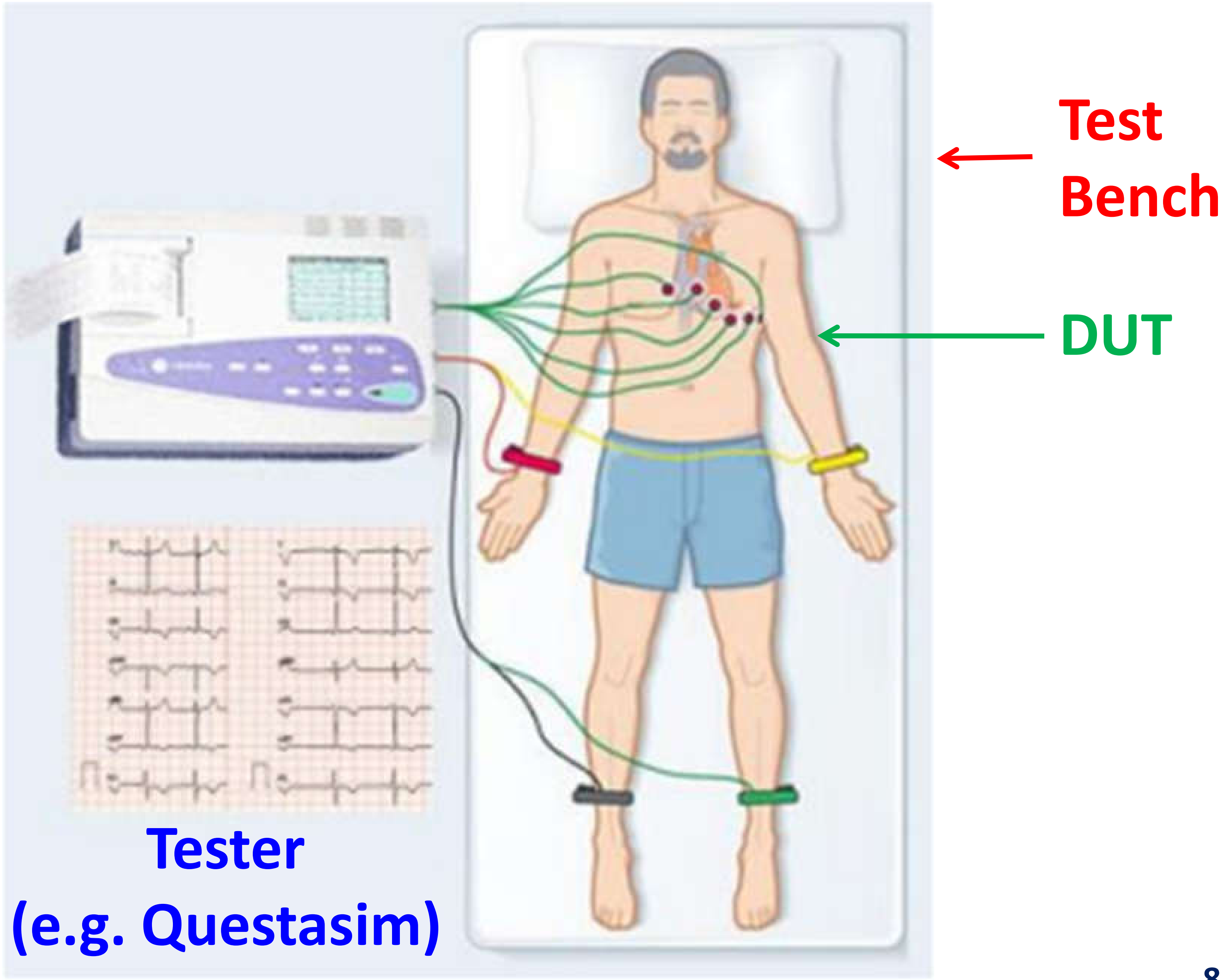
```
`timescale 1 ns/10 ps

module full_adder_tb;

reg [3:0] a;
reg [3:0] b;
reg c;
wire [3:0] sum;
wire carry;

integer i;

full_adder dut (
                .a_in(a),
                .b_in(b),
                .c_in(c),
                .carry_out(carry),
                .sum_out(sum)
                );

initial
    begin
        a <= 0;
        b <= 0;
        c <= 0;

        $monitor ("a=0x%0h b=0x%0h c=0x%0h carry=0x%0h sum=0x%0h", a

        for (i=0; i < 5; i=i+1) begin
            #10 a <= $random;
            b <= $random;
            c <= $random;
        end
    end
endmodule
```

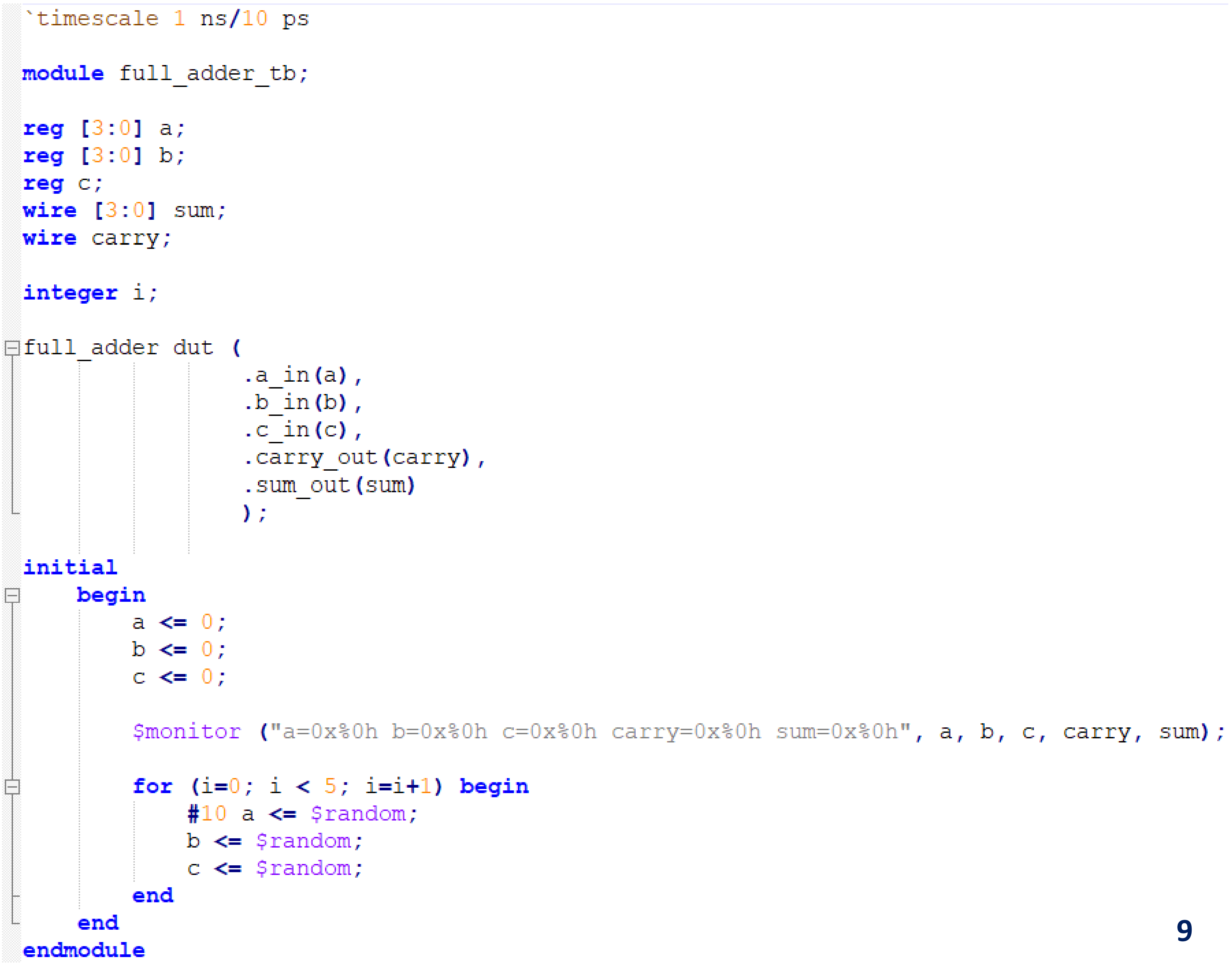
Connected



Modelsim/Questasim (with PC)


Test Bench
DUT
Tester (e.g. Questasim)


Test Bench
DUT
Tester
(e.g. Questasim)



```verilog
`timescale 1 ns/10 ps

module full_adder_tb;

reg [3:0] a;
reg [3:0] b;
reg c;
wire [3:0] sum;
wire carry;

integer i;

full_adder dut (
                .a_in(a),
                .b_in(b),
                .c_in(c),
                .carry_out(carry),
                .sum_out(sum)
                );

initial
    begin
        a <= 0;
        b <= 0;
        c <= 0;

        $monitor ("a=0x%0h b=0x%0h c=0x%0h carry=0x%0h sum=0x%0h", a, b, c, carry, sum);

        for (i=0; i < 5; i=i+1) begin
            #10 a <= $random;
            b <= $random;
            c <= $random;
        end
    end
endmodule
```

QuestaSim

- QuestaSim is a **functional verification simulator** of **Mentor Graphics**,
  - developed as an integrated platform under Questa Advanced Simulator software,
    - which is the core simulation and debugging engine.
  - powered by ModelSim tool,
    - which both are the software products belonging to the **same company**.

- **QuestaSim** offers
    - higher capacity and supports larger FPGA and SoC,
  - where **ModelSim** supports smaller designs.

QuestaSim (2)

- Because of its **verification** supportive nature, QuestaSim
  - is compatible with a large array of languages containing
    - **Verilog** HDL,
    - **SystemVerilog**
    - and VHDL;
  - and offers support for SystemVerilog based **UVM libraries** in hardware description and verification.

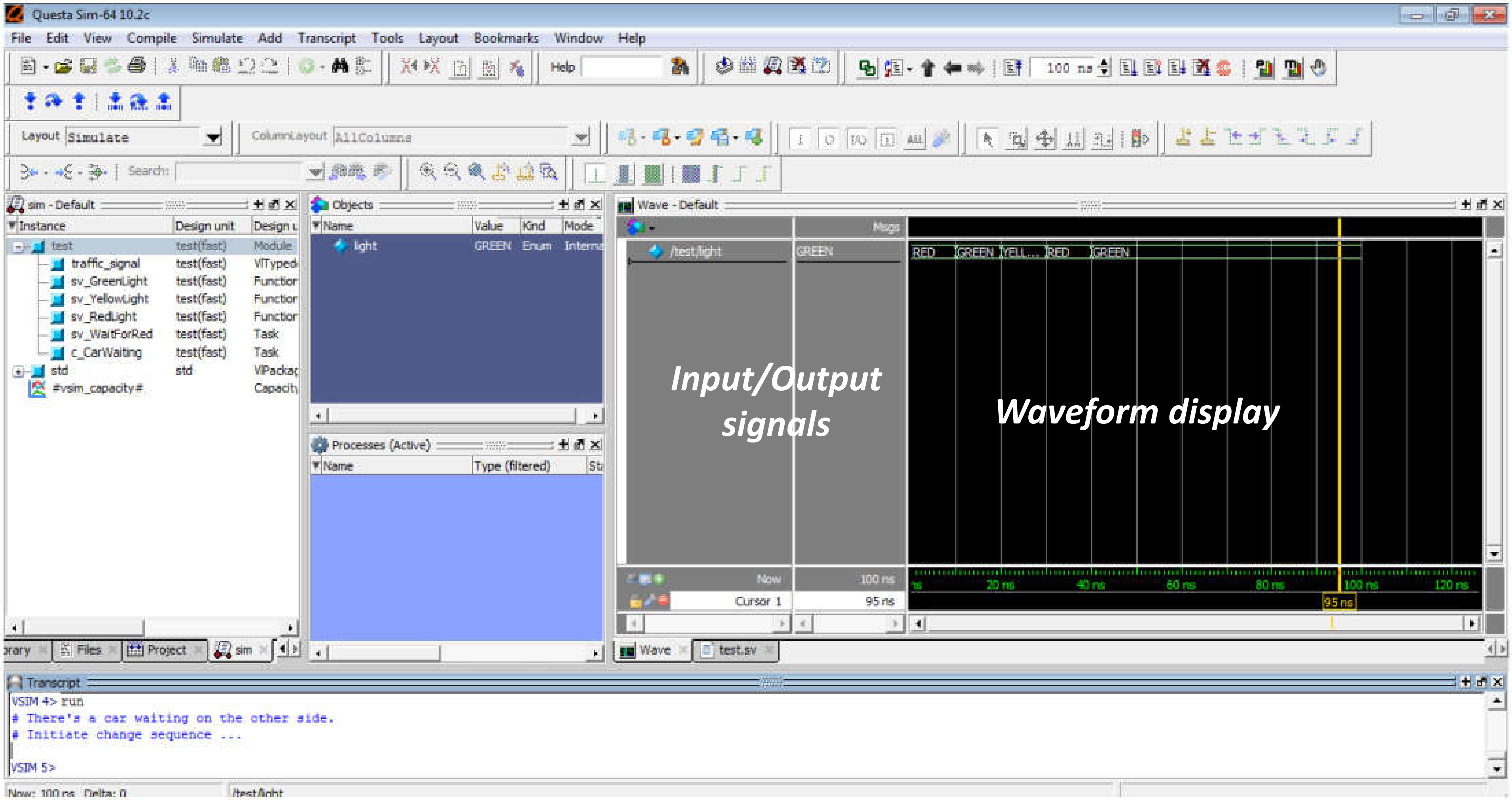
- QuestaSim was used for running the *complete UVM testbench*
  - in order to observe the *verification results*.

QuestaSim (3)

- One of the main differences between QuestaSim and Modelsim (besides performance/capacity) is that
  - QuestaSim is the simulation engine for the Questa Platform which includes integration of
    - Verification Management,
    - Formal based technologies,
    - Questa Verification IP,
    - Low Power Simulation and
    - Accelerated Coverage Closure technologies.

- QuestaSim natively supports SystemVerilog for **Testbench**, **UPF**, UCIS, OVM/**UVM**.

A screenshot from QuestaSim simulation environment



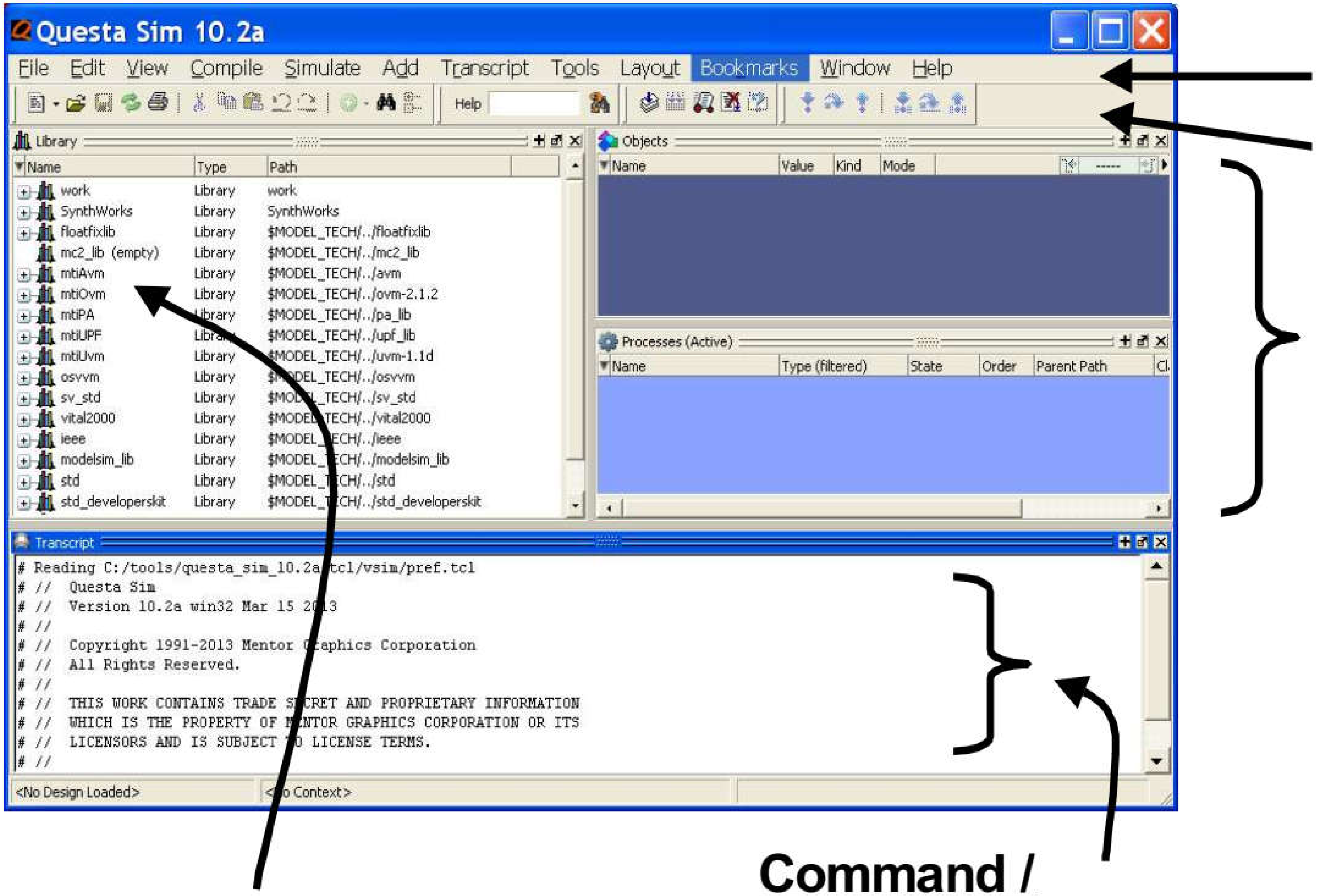
Instance	Design unit	Design u
test	test(fast)	Module
traffic_signal	test(fast)	VITyped
sv_GreenLight	test(fast)	Function
sv_YellowLight	test(fast)	Function
sv_RedLight	test(fast)	Function
sv_WaitForRed	test(fast)	Task
c_CarWaiting	test(fast)	Task
std	std	VIPackag
#vsim_capacity#		Capacity

Name	Value	Kind	Mode
light	GREEN	Enum	Interna

	Msgs
/test/light	GREEN

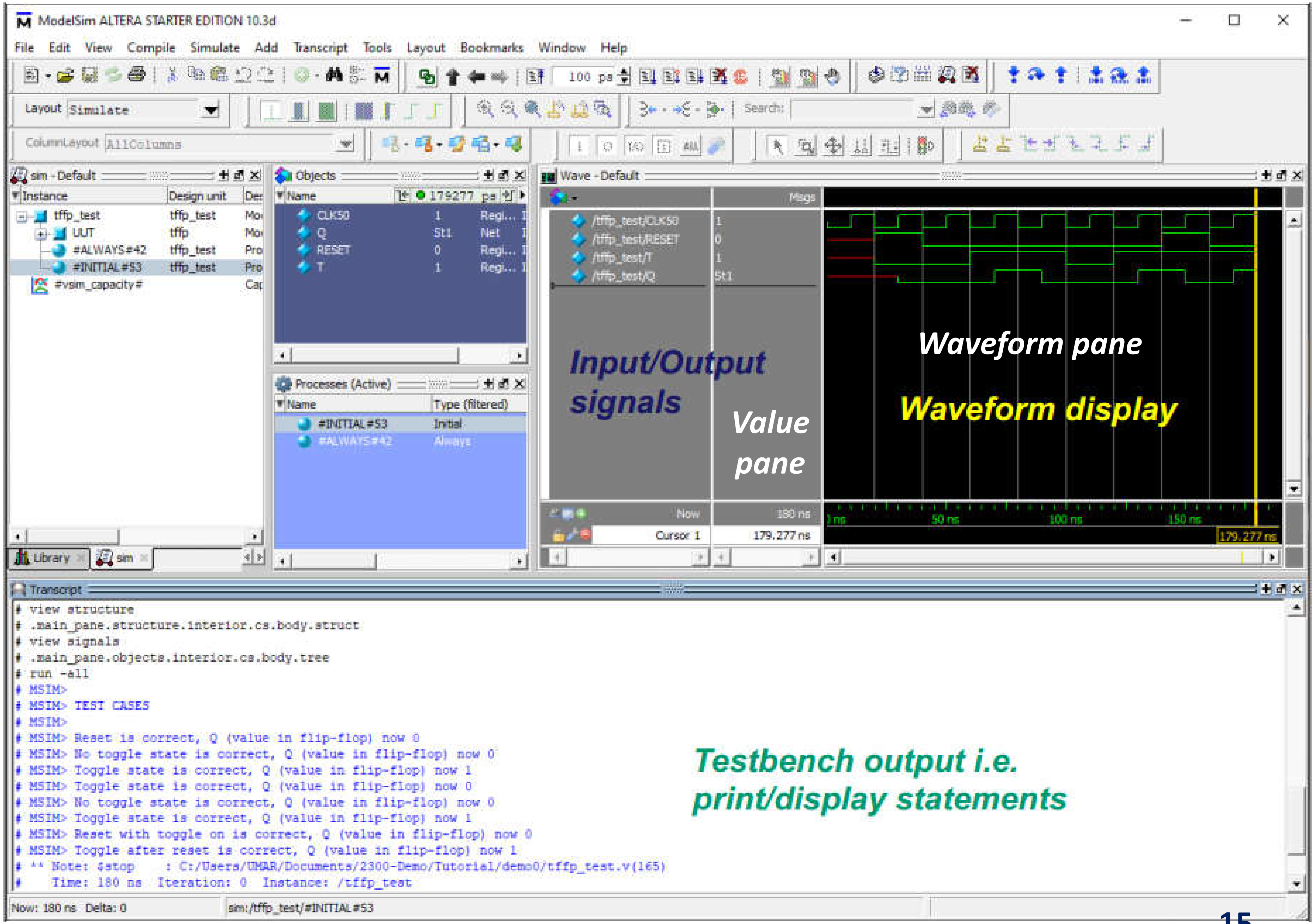
```
VSIM 4> run
# There's a car waiting on the other side.
# Initiate change sequence ...

VSIM 5>
```

Now: 100 ns Delta: 0 /test/light


Questa Sim 10.2a
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
Help
Library
Objects
Name Value Kind Mode
Processes (Active)
Name Type (filtered) State Order Parent Path
Transcript
<No Design Loaded>
<No Context>
Menu
Buttons
Other Windows
Library Window
Command / Transcript Window

Name	Type	Path
work	Library	work
SynthWorks	Library	SynthWorks
floatfixlib	Library	$MODEL_TECH/../floatfixlib
mc2_lib (empty)	Library	$MODEL_TECH/../mc2_lib
mtiAvm	Library	$MODEL_TECH/../avm
mtiOvm	Library	$MODEL_TECH/../ovm-2.1.2
mtiPA	Library	$MODEL_TECH/../pa_lib
mtiUPF	Library	$MODEL_TECH/../upf_lib
mtiUvm	Library	$MODEL_TECH/../uvm-1.1d
osvvm	Library	$MODEL_TECH/../osvvm
sv_std	Library	$MODEL_TECH/../sv_std
vital2000	Library	$MODEL_TECH/../vital2000
ieee	Library	$MODEL_TECH/../ieee
modelsim_lib	Library	$MODEL_TECH/../modelsim_lib
std	Library	$MODEL_TECH/../std
std_developerskit	Library	$MODEL_TECH/../std_developerskit

```
# Reading C:/tools/questa_sim_10.2a/tcl/vsim/pref.tcl
# //  Questa Sim
# //  Version 10.2a win32 Mar 15 2013
# //
# //  Copyright 1991-2013 Mentor Graphics Corporation
# //  All Rights Reserved.
# //
# //  THIS WORK CONTAINS TRADE SECRET AND PROPRIETARY INFORMATION
# //  WHICH IS THE PROPERTY OF MENTOR GRAPHICS CORPORATION OR ITS
# //  LICENSORS AND IS SUBJECT TO LICENSE TERMS.
# //
```



```
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# MSIM>
# MSIM> TEST CASES
# MSIM>
# MSIM> Reset is correct, Q (value in flip-flop) now 0
# MSIM> No toggle state is correct, Q (value in flip-flop) now 0
# MSIM> Toggle state is correct, Q (value in flip-flop) now 1
# MSIM> Toggle state is correct, Q (value in flip-flop) now 0
# MSIM> No toggle state is correct, Q (value in flip-flop) now 0
# MSIM> Toggle state is correct, Q (value in flip-flop) now 1
# MSIM> Reset with toggle on is correct, Q (value in flip-flop) now 0
# MSIM> Toggle after reset is correct, Q (value in flip-flop) now 1
# ** Note: $stop    : C:/Users/UMAR/Documents/2300-Demo/Tutorial/demo0/tffp_test.v(165)
#    Time: 180 ns  Iteration: 0  Instance: /tffp_test
```

Now: 180 ns Delta: 0   sim:/tffp_test/#INITIAL#53

Buttons	Description*	Command Line

	Compile this file
	open the Compile Source File dialog; in a project, compile the file
		To compile:
		`vlog -work [working library] [filename.v]`
		`ex) vlog -work work tb_tutorial.v`
		
		To recompile:
		`vlog -work [working library] -refresh`

	Compile All compile all files in the open project	Compile > Compile All

	Simulate
	load the selected design unit or simulation configuration object
		`vsim -c [working library].[file name]`
		
		`Ex) vsim -c work.tb_tutorial`

	Restart reload the design elements and reset the simulation time to zero, with the option of using current formatting, breakpoints, WLF file, virtual definitions, and assertion settings	`restart -f`
	Run run the current simulation for the specified run length	`run`
	Open Wave Viewer	Window > Wave
	Zoom Mode set mouse to Zoom Mode – drag left mouse button to zoom, click middle mouse button to select	N/A
	Insert Cursor add a cursor to the waveform pane	Insert > Cursor

Thank You