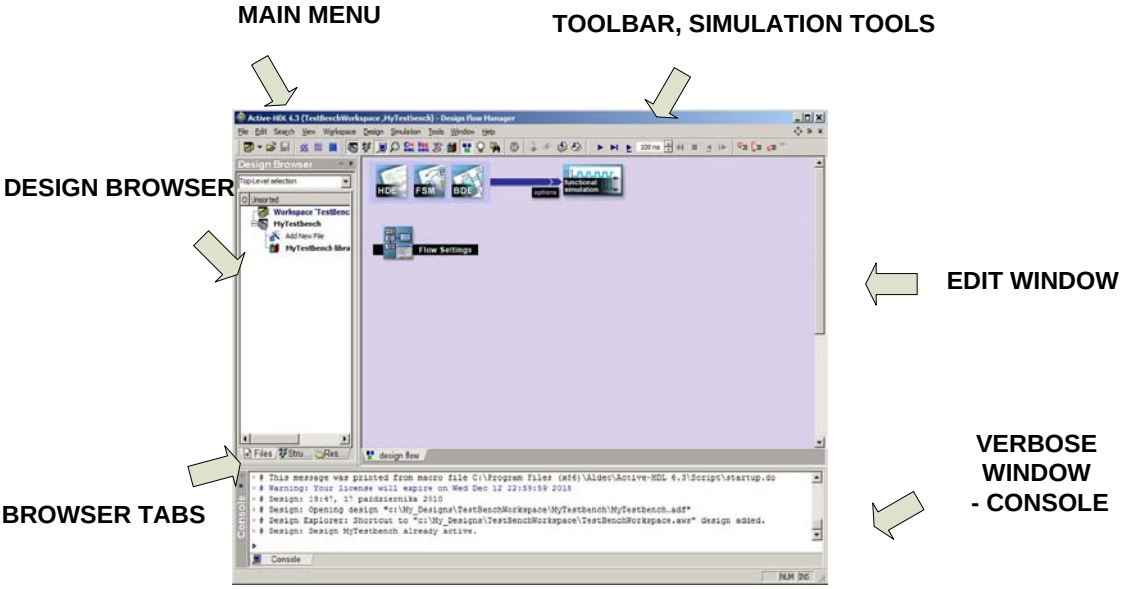
# Programmable Digital Systems – Exercise 1

Goals:

- Getting started with usage of ActiveHDL,
- Getting started with Verilog syntax,
- Circuit simulation: clock generator, data registers.

Exercise:

1. Run Active-HDL.
2. Select „Create new workspace”.
3. As a destination folder, please select „X:\”. Please select „Add New Design to Workspace”. Set name as e.g. “MyTestbenchWorkspace”.
4. Select “Create an empty design”.
5. In combo box “Default HDL Language” select “VERILOG”.
6. Give a design name e.g. „MyTestbench”.




7. Add a new file by selecting an option In design browser (Add New File), then use „Verilog Source Code”. Set the file’s name to e.g. ”mytestbench.v”. Fill the file:

```
module mytestbenchmodule();
reg CLK;
initial
begin
    $monitor(“CLK=%d”, CLK);
    CLK = 0;

    forever
    begin
            #50 CLK = ! CLK;
    end
end
endmodule
```

8. Use Design/Compile All from Main Menu. After that, compilation status should have been displayed in the console:

```
# Compile success 0 Errors 0 Warnings  Analysis time : 0[s].
# done
```

9. In Design Browser, a top-level module must be selected („Top-Level Selection”). The current project has only one module –e.g. mytestbenchmodule – so please select it.

10. With use of toolbar, run simulation for a while (▶) a then stop it (■). There should be some new lines in the console window:

```
…
# KERNEL: CLK=0
# KERNEL: CLK=1
# KERNEL: CLK=0
# KERNEL: CLK=1
….
```

11. Please add a waveform figure, by using command from Main Menu: „File”/”New”/”Waveform”.
12. In Design Browser select „Structure” tab, and then, by using mouse cursor, drag the module (e.g. „mytestbenchmodule”) and drop it onto edit window with waveform. CLK signal should show.
13. With use of toolbar, run the simulation again for a while (▶) and then stop it (■). Zoom the waveform in (  ) until the view is clear enough.

14. Reinvoke the experiment with the program below:

```
module mytestbenchmodule();
reg CLK;
initial CLK  <= 0;
always #50  CLK <= ~CLK;

initial
begin
$monitor("CLK=%d", CLK);
end
endmodule
```

15. What are the differences between these two programs?
    What is the  period of generated clock signal? Why is it such?
16. Type the following program:

```
module mytestbenchmodule();

reg CLK;
initial CLK  <= 0;
always #50  CLK <= ~CLK;

reg [7:0] A;
initial A=0;

reg B;
initial B=1;

always @(posedge CLK) A <= A + B;

initial $monitor("clk=%d A=%d B=%d", CLK, A, B);
endmodule
```

17. Compile– Main Menu– „Design”/”Compile All”.
18. WARNING: Although the project library has been compiled, Simulator still uses old data about modules (e.g. „mytestbenchmodule”). Because of that, simulator must be restarted with use of commands from Main Menu „Simulation”/”End Simulation” and then „Initialize Simulation”. Now it is possible to drag&drop signals of

„mytestbenchmodule” module on the waveform window. CLK, A and B signals should be displayed.

19. Simulate 100ns (▶).
20. Change how register A i displayed to decimal mode (right mouse button, „Properties”, „General” tab, “Values” frame : and finally - „Decimal”).
21. Add keyboard controlled stimulus to B signal (right mouse button, „Stimulators…”, „Signals” tab, Type = „Hotkey”, then „Press New hotkey”, and press a key on the keyboard e.g. „B”, - finally, press„Apply”).
22. Alternatively keep pressing the selected key (e.g. „B”) i and simulating 50ps fragments (▶). Check how does the stimulus work, and what is its impact on counting register „A”.
23. Add a new VERILOG file to the design e.g. „increment.v” and use the following program sources:

„increment.v” file

```
module increment(
    input wire RST,
    input wire CLK,
    input wire B,
    output wire [7:0] CNT
);
reg [7:0] a;

always @(posedge CLK or posedge RST)
    if (RST) a <= 0; else a <= a + B;

assign CNT = a;

endmodule
```

„mytestbench.v” file

```
module mytestbenchmodule();
reg CLK;
initial CLK  <= 0;
always #50  CLK <= ~CLK;

reg RST;
initial
begin
    RST <= 0;
    RST <= #100 1;
    RST <= #500 0;
end

increment my_increment1(
    .CLK(CLK),
    .RST(RST),
    .B(1),
    .CNT()
);

Endmodule
```

24. Use “Compile All”, restart the simulation (”End Simulation” and again:„Initialize Simulation”).
25. Drag&drop both modules („mytestbenchmodule” and „my_increment1”) on the wavefrom –CLK, RST, A, B, CLK and RST signals should show.
26. Repeat the experiment with assigning a keyboard stimulus.

27. How does this design work?
    How register are initialized in each of those modules?
    From when module „my_increment” starts counting?

28. Add a new file to the design:

„sum.v” file

```
module sum
(
input wire [7:0] A,
input wire [7:0] B,
output wire [8:0] C
);

assign C = A+B;

endmodule
```

….and modify „mytestbench.v” file as follows:

```
module mytestbenchmodule();
reg CLK;
initial CLK  <= 0;
always #50  CLK <= ~CLK;

reg RST;
initial
begin
    RST <= 0;
    RST <= #100 1;
    RST <= #500 0;
end

wire [7:0] cnt1, cnt2;
wire [8:0] cnt12;

increment my_increment1(
    .CLK(CLK),
    .RST(RST),
    .B(1),
    .CNT(cnt1)
);

increment my_increment2(
    .CLK(CLK),
    .RST(RST),
    .B(cnt1[0]),
    .CNT(cnt2)
);

sum sum1(
.A(cnt1),
.B(cnt2),
.C(cnt12)
);

endmodule
```

29. „Compile All”, restart the simulation  (”End Simulation” and again: „Initialize Simulation”). Drag&drop all modules on the waveform.
30. Which of the modules counts faster and which of them counts slower?
    What invokes counting in „my_increment2” module ?
    What value is outputted from module „sum1” (C port) ?

31. Run tutorial from Main Menu: „Help”/”Interactive Verilog Tutorial”.