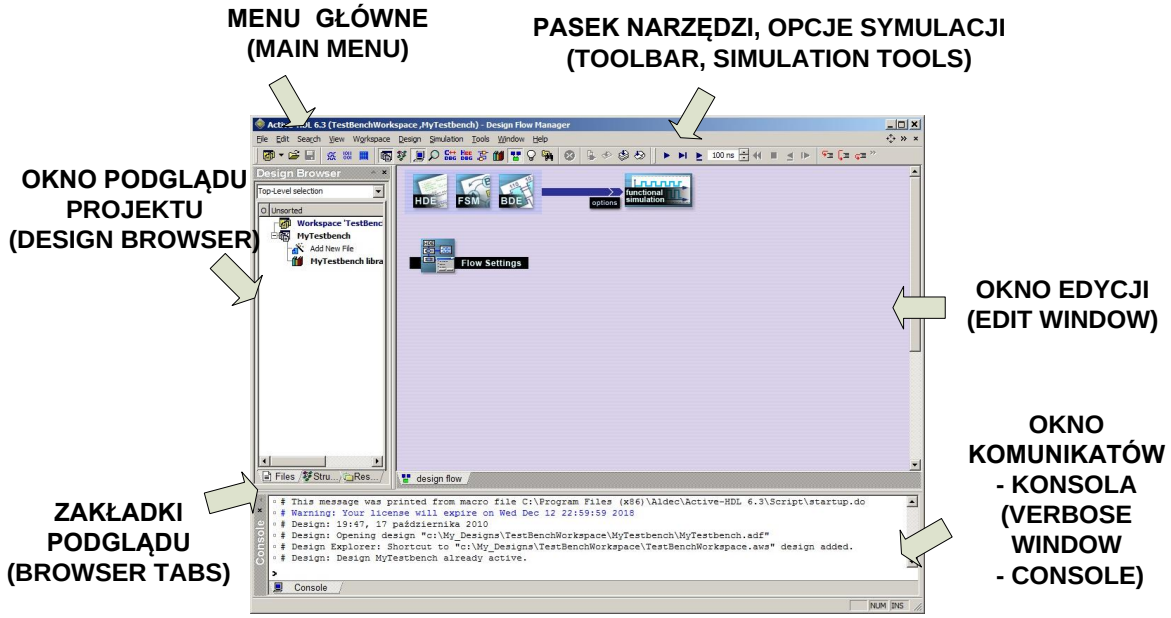
Programowalne Układy Cyfrowe – Ćwiczenie 1

Cele:

- zapoznanie z obsługą programu ActiveHDL,
- zapoznanie ze składnią języka Verilog,
- symulacja układów: generujących przebieg zegarowy, rejestr danych.

Przebieg ćwiczenia:

1. Uruchomić program Active-HDL.
2. Wybrać opcję „Create new workspace”.
3. Jako katalog docelowy wybrać dysk „X:\”. Zaznaczyć opcję „Add New Design to Workspace”. Nadać nazwę np. “MyTestbenchWorkspace”.
4. Wybrać “Create an empty design”.
5. W opcji “Default HDL Language” wybrać “VERILOG”.
6. Nadać nazwę projektowi (design name) np. „MyTestbench”.



7. Dodać nowy plik w oknie podglądu projektu (Add New File), „Verilog Source Code” o nazwie np. ”mytestbench.v” i następującej treści:

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

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

8. Z menu głównego wybrać Design/Compile All. W konsoli (okno komunikatów) powinien wyświetlić się status:

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

9. W oknie podglądu projektu należy wybrać najwyższy moduł w hierarchii projektu „Top-Level Selection". Rozpatrywany projekt ma tylko jeden moduł i należy go wybrać (np. mytestbenchmodule).

10. Korzystając z paska narzędzi, uruchomić symulację na chwilę (▶) a następnie zatrzymać (■). W konsoli powinny wyświetlić się zmiany sygnału zegarowego:

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

11. Dodać do projektu wykres – menu główne: „File"/"New"/"Waveform".
12. Na pasku podglądu wybrać „Structure", a następnie kursorem myszy przeciągnąć moduł (np. „mytestbenchmodule") na okno edycji zawierające wykres. Powinien pojawić się sygnał CLK.
13. Korzystając z paska narzędzi, ponownie uruchomić symulację na chwilę (▶) a następnie zatrzymać (■). Zbliżyć okno wykresu () aż do uzyskania widocznego przebiegu.

14. Powtórzyć eksperyment z programem:

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

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

15. Czy i jakie są różnice w działaniu tych dwóch programów?
    Z czego wynika okres wygenerowanego przebiegu zegarowego?
16. Wprowadzić 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. Skompilować – menu główne – „Design"/"Compile All".

18. UWAGA: Mimo, iż biblioteka projektu została skompilowana, symulator dalej wykorzystuje poprzednie (stare) informacje na temat sygnałów modułu np. „mytestbenchmodule”. Dlatego należy wybrać z menu głównego „Simulation”/”End Simulation” i ponownie „Initialize Simulation”. Teraz można ponownie przeciągnąć kursorem myszy sygnały modułu „mytestbenchmodule” na okno z przebiegiem czasowym. Powinny na nim pojawić się sygnały CLK, A, B.
19. Dokonać krótkiej symulacji o długości 100ns (▶).
20. Zmienić sposób wyświetlania wartości rejestru A na dziesiętny (prawy klawisz, „Properties”, zakładka „General”, ramka „Values” : „Decimal”).
21. Dodać wymuszenie dla sygnału B sterowane z pomocą klawiatury (prawy klawisz, „Stimulators…”, zakładka „Signals”, Type = „Hotkey”, następnie „Press New hotkey”, wcisnąć jakiś klawisz np. „B”, i dalej „Apply”).
22. Na zmianę wciskając wybrany klawisz (np. „B”) i symulując kolejne fragmenty 50ps (▶) sprawdzić działanie wymuszenia oraz to, czy steruje ono działaniem rejestru zliczającego „A”.
23. Do projektu dodać nowy plik VERILOG np. „increment.v” i wykorzystać następujące programy:

Plik „increment.v”

```
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
```

Plik „mytestbench.v”

```
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. Skompilować (Compile All), zrestartować symulację (”End Simulation” i ponownie „Initialize Simulation”).
25. Na przebieg czasowy przeciągnąć oba moduły („mytestbenchmodule” i „my_increment1”) – powinny pojawić się sygnały CLK, RST, A, B, CLK, RST.
26. Powtórzyć eksperyment z wymuszeniem przypisanym do klawisza.

27. Jak działa napisany układ?
    W jaki sposób dokonywana jest inicjalizacja rejestrów w każdym z modułów?
    Od którego momentu moduł „my_increment” zaczyna zliczać kolejne wartości?

28. Do projektu dodać plik

Plik „sum.v”

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

assign C = A+B;

endmodule
```

oraz zmodyfikować plik „mytestbench.v”

```
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. Skompilować (Compile All), zrestartować symulację (”End Simulation” i ponownie „Initialize Simulation”), za na przebiegu czasowym umieścić wszystkie moduły.
30. Który z modułów zlicza szybciej a który wolniej?
    Co „wyzwala” zliczanie w module „my_increment2” ?
    Jaka wartość znajduje się na wyjściu modułu „sum1” (port C) ?

31. Uruchomić z menu głównego „Help”/”Interactive Verilog Tutorial”.