Debugging in Verilog Vivado

Neciu Laurentiu Florin 331CC

Digori Gheorghe 331CC

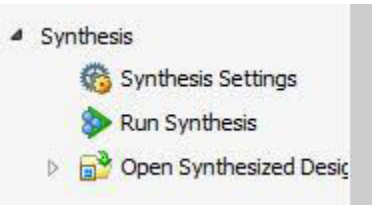
Scopul lucrarii:

Ne propunem sa depanam un program scris in verilog care calculeaza maximul a 3 numere pe 16 biti.

Se da urmatorul modul care contine diverse probleme ce trebuie depanate.

```
module max16b(a, b, c, y);

input signed [15:0] a, b, c;
output signed [15:0] y;

wire ul_lt;
assign ul_lt = (a<b);
wire max_ab;
assign max_ab = ul_lt? b:a;
wire u2_lt;
assign u2_lt = (c > max_ab);
assign y = u2_lt?c : max_ab;

endmodule
```

Se inceaca depanarea modulului. Se sintetizeaza modulul verilog dand click pe “Run synthesis module”. Dupa ce programul a terminat sinteza observam o fereastra de dialog care ne permite sa accesam raportul generat de aceasta sinteza.



WARNING: [Synth 8-3848] Net y in module/entity max16b does not have driver.

```
INFO: [Synth 8-638] synthesizing module 'max16b' [C:/Users/Florin/Documents/
WARNING: [Synth 8-3848] Net y in module/entity max16b does not have driver.
INFO: [Synth 8-256] done synthesizing module 'max16b' (1#1) [C:/Users/Florir
```

Acestu lucru arata ca y a fost declarat ca iesire insa nu a fost assignat in program. Vom rezolva eroarea punand “assign y = u1_lt?c : max_ab” deoarece variabila noastra de iesire a fost notata y nu max.

```
module max16b(a, b, c, y);

input signed [15:0] a, b, c;
output signed [15:0] y;

wire ul_lt;
assign ul_lt = (a<b);
wire max_ab;
assign max_ab = ul_lt? b:a;
wire ul_lt;
assign ul_lt = (c > max_ab);
assign y = ul_lt?c : max_ab;

endmodule
```

Aceasta eroare dispare din raport-ul generat la sinteza in momentul in care codul a fost modificat.

Pentru a testa functionalitatea codului, vom concepe un modul de simulare si un fisier cu un set de valori date. Modulul are rolul de a automatiza testarea pentru valorile din fisier.

Avem mai jos codul pentru pentru verilog pentru modulul de simulare.

```
integer fd;
integer count, status;
integer i_a, i_b, i_c, i_result;
integer errors;
max16b uut(a,b,c,y);
initial begin
    a=0; b=0; c=0;
    fd = $fopen("../../../values.txt", "r");
    if (fd == 0)
        fd = $fopen("../../../../values.txt", "r");
    count = 1;
    # 100;
    errors = 0;
    while ($fgets(align, fd))
    begin
        status = $sscanf(align, "%d %d %d %d", i_a, i_b, i_c, i_result);
        a = i_a; b = i_b; c = i_c;
        # 50;
        if (y == i_result)
            $display("%d(%t): pass, a:%d, b:%d, c:%d, y:%d\n", count, $time, a, b, c, y);
        else
        begin
            $display("%d(%t): fail, a:%d, b:%d, c:%d, y(actual):%d, y(asteptat):%d\n", count, $time, a, b, c, y, i_result);
            errors = errors + 1;
        end

        count = count + 1;
    end
end
```

```
-250 -43 10 10
60 90 60 90
-10 22 -20 22
-101 -22 -531 -22
1 55 2 55
1 4 2 4
```

Acest cod deschide un fisier denumit “values.txt” si il parseaza folosint functia de citire sscanf. Valorile intregi (integer) sunt salvate mai apoi in valori registru care sunt trimise modulului la care vrem sa facem depanarea. Se verifica daca iesirea modului nostru este in conformitate cu rezultatul citit. Rezultatele depanarii sunt afisate in consola programului.

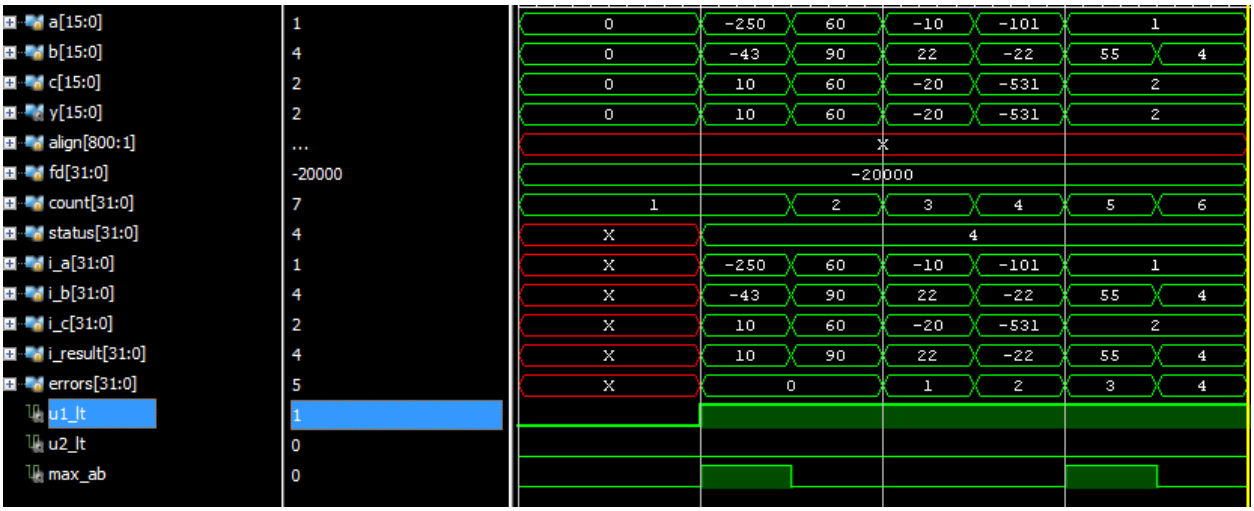
Intrarile fisierului values.txt sunt dat mai sus iar mai jos se observa iesirile consolei. Observam ca programul nostru nu trece majoritatea testelor.

```
1(              150000): pass, a:  -250, b:   -43, c:    10, y:    10
2(              200000): fail, a:    60, b:    90, c:    60, y(actual):    60, y(asteptat):        90
3(              250000): fail, a:   -10, b:    22, c:   -20, y(actual):   -20, y(asteptat):        22
4(              300000): fail, a:  -101, b:   -22, c:  -531, y(actual):  -531, y(asteptat):       -22
5(              350000): fail, a:     1, b:    55, c:     2, y(actual):     2, y(asteptat):        55
6(              400000): fail, a:     1, b:     4, c:     2, y(actual):     2, y(asteptat):         4
```

Pentru a determina care sunt variabilele problematice din program, consultam formele de unde din simulare.

Putem afisa variabile din program dand click pe modulul nostru (uut) in fereastra “Scopes”. Odata ce am facut asta putem trage din fereastra “Objects” variabilele u1_lt, u2_lt si max_ab in fereastra de simulare. Resetam apoi simularea si observam formele de unda.

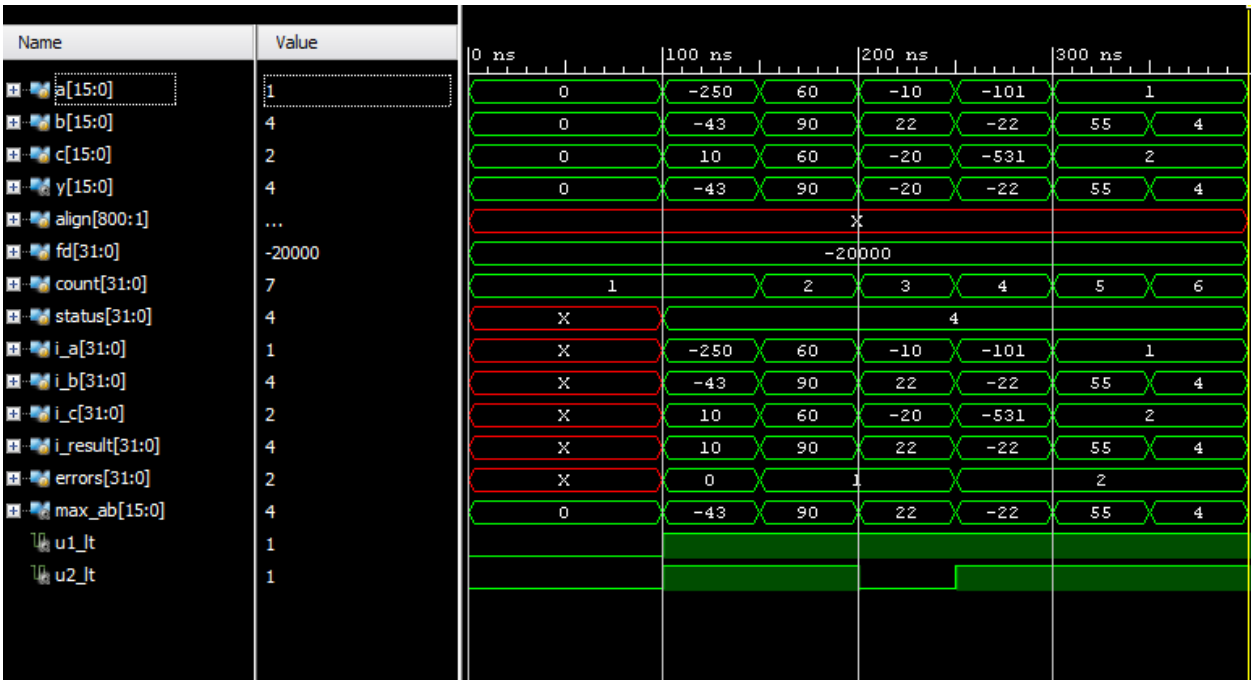
Obs: se poate face ca variabilele din simulare sa aiba un format decimal signed selectandu-le, dand click dreapta, radix si signed decimal.



```
module max16b(a,b,c,y);
    input signed [15:0] a,b,c;
    output signed [15:0] y;

    wire u1_lt;
    assign u1_lt = (a < b);

    wire [15:0] max_ab;
    assign max_ab = u1_lt ? b : a;

    wire u2_lt;
    assign u2_lt = (c < max_ab);

    assign y = u2_lt ? max_ab : c;

endmodule
```

Din aceasta simulare observam imediat o eroare la variabila max_ab. Variabila max_ab are valoare 1 sau 0. Observam ca dimensiunea wire-ului este de 1 bit. Ne uitam la modul cum este declarata variabila in programul nostru si observam ca variabila este declarata ca “wire” nu ca “wire [15:0]” cum ne-ar fi trebuit. Facem modificarile si resetam simularile. Codul modficat se afla in stanga iar simularile pentru codul modificat se afla mai jos.

```
        1(              150000): fail, a:  -250, b:   -43, c:    10, y(actual):   -43, y(asteptat):        10
        2(              200000): pass, a:    60, b:    90, c:    60, y:    90
        3(              250000): fail, a:   -10, b:    22, c:   -20, y(actual):   -20, y(asteptat):        22
        4(              300000): pass, a:  -101, b:   -22, c:  -531, y:   -22
        5(              350000): pass, a:     1, b:    55, c:     2, y:    55
        6(              400000): pass, a:     1, b:     4, c:     2, y:     4
```



Din rezultatele testelor din consola si simulari observam ca modulul nostru nu indeplineste toate testele. La primul test (pe care nu il trece) observam ca variabila

```
module max16b(a,b,c,y);
    input signed [15:0] a,b,c;
    output signed [15:0] y;

    wire ul_lt;
    assign ul_lt = (a < b);

    wire signed [15:0] max_ab;
    assign max_ab = ul_lt ? b : a;

    wire u2_lt;
    assign u2_lt = (c < max_ab);

    assign y = u2_lt ? max_ab : c;

endmodule
```

logica u2_lt este adevarata desi c (cu valoarea 10) nu este mai mic decat max_ab (cu valoarea -43). Motivul pentru care acest test esueaza este ca se compara prost numerele. Incercam sa vedem modul cum au fost declarate. Variabila “c” este un signed de 15 biti insa variabila max_ab este un wire de 15 biti nedeclarat ca signed. Consultand documentatia aferenta am aflat ca varbialele nedeclarate ca signed sunt in mod nativ unsigned. Modificam codul conform specificatiilor. Refacem simularile.

```
000ns
   1(              150000): pass, a:  -250, b:   -43, c:    10, y:    10

   2(              200000): pass, a:    60, b:    90, c:    60, y:    90

   3(              250000): pass, a:   -10, b:    22, c:   -20, y:    22

   4(              300000): pass, a:  -101, b:   -22, c:  -531, y:   -22

   5(              350000): pass, a:     1, b:    55, c:     2, y:    55

   6(              400000): pass, a:     1, b:     4, c:     2, y:     4
```

Modulul este acum functional. Se pot genera mai multe intrari in fisier pentru test (posibil folosind un alt utilitar) pentru a testa mai bine modulul, testarea fiind automatizata.