Task-uri și funcții

Task-uri

- se definesc în interiorul modulelor în care se folosesc. Se poate defini un task într-un fișier separat și să se folosească `include.
- pot conține elemente de timing precum posedge, negedge sau întârzieri
- pot avea un număr de intrări și un număr de ieșiri. Ordinea în care se face declararea intrărilor și ieșirilor definește ordinea în care variabilele sunt furnizate la apelul task-ului.
- se pot declara variabile în interior și acestea vor fi locale
- pot lucra cu variabile globale, definie în modulul în care este definit task-ul.Se pot folosi variabilele globale în expresii sau li se pot da valori
- când sunt folosite variabile locale, valorile ieșirilor sunt asignate la sfârșitul execuției task-ului
- pot apela un alt task sau o funcție
- pot modelalogică secvențială sau combinațională
- trebuie apelate explicit
- nu pot face parte din expresii
- sintaxa:

task <nume_task>;

[declaratii_intrari;]

[declaratii_iesiri;]

[declaratii_variabile_locale;]

[begin]

[asignari;]

[end]

endtask

Task care folosește variabile globale:

```
module_task_variabile_globale(…….);
    reg [7:0] a, b;
    reg [7:0] a_and_b, a_or_b, a_xor_b;
    always @(a or b)
    begin
        oper_bit;
    end
    ……..
    task oper_bit;
        begin
            a_and_b = a & b;
            a_or_b = a | b;
            a_xor_b = a ^ b;
```

```
            end
        endtask
        ……….....
    endmodule
```

Task cu intrări și ieșiri proprii:

```
    module adder8(a, b, cin, s, cout);
        input [7:0] a, b;
        input cin;
        output [7:0] s;
        output cout;
            assign {cout,s} = a + b + cin;
    endmodule

    module test_adder8;
        reg [7:0] at, bt;
        reg cint;
        wire [7:0] st;
        wire coutt;
        adder8 adder8_inst(at, bt, cint, st, coutt);
        task stimuli;
            input [7:0] val_at, val_bt;
            input val_cint;
            begin
            at = val_at;
            bt = val_bt;
            cint = val_cint;
            end
        endtask
        initial
        begin
            stimuli(8'haa, 8'h55, 1'b0);
            stimuli(8'haa, 8'h55, 1'b1);
            stimuli(8'hf0, 8'h0f, 1'b0);
            stimuli(8'hf0, 8'h0f, 1'b1);
            stimuli(8'h34, 8'he7, 1'b1);
            .....
        end
    endmodule
```

Funcții

- sunt definite în modulul în care se folosesc. Se pot defini funcții în fișiere separate și să se folosească `include.
- nu pot fi incluse elemente de timing. O funcție este executată în timpde simulare 0
- pot avea oricâte intrări, dar o singură ieșire
- variabilele declarate în interior sunt locale funcției. Ordinea declarării intrărilor furnizează ordinea în care variabilele sunt pasate funcției la apelarea acesteia
- se pot folosi variabile globale, declarate în modulul în care apare descrierea funcției

- atunci când se folosesc variabile locale, rezultatul este asignat la sfârșitul execuției funcției
- pot fi folosite pentru modelarea de logică combinațională
- pot apela alte funcții, dar nu pot apela task-uri
- sintaxa:

function <nume_functie>;

[declaratii_intrari;]

[declaratii_variabile_locale;]

[begin]

[asignari;]

[end]

endfunction

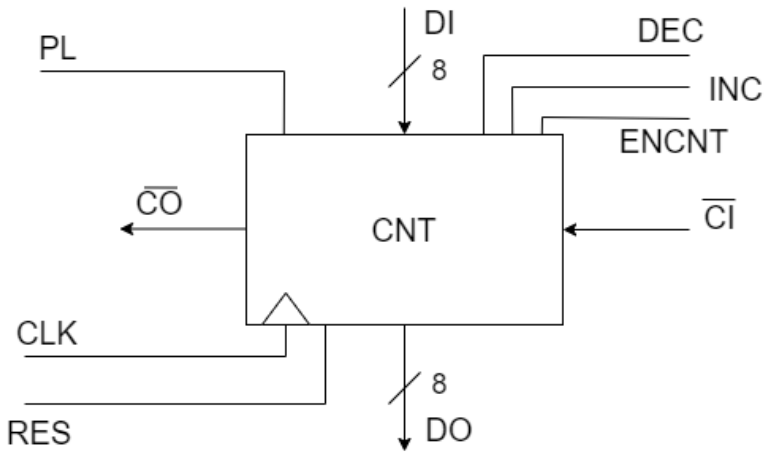
Exemplu de descriere și utilizare a unei funcții

```
module exemplu_functie(a, b, c, d, e, z);
    input a, b, c, d, e;
    output z;
    function functie_logica;
        input val_a, val_b, val_c, val_d;
        begin
            functie_logica = ((val_a & val_b) | (val_c & val_d));
        end
    endfunction
    assign z = e ? functie_logica(a, b, c, d) : 0;
endmodule
```

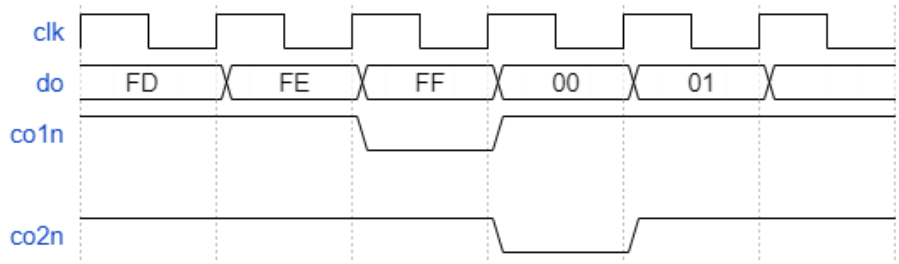
Am2940

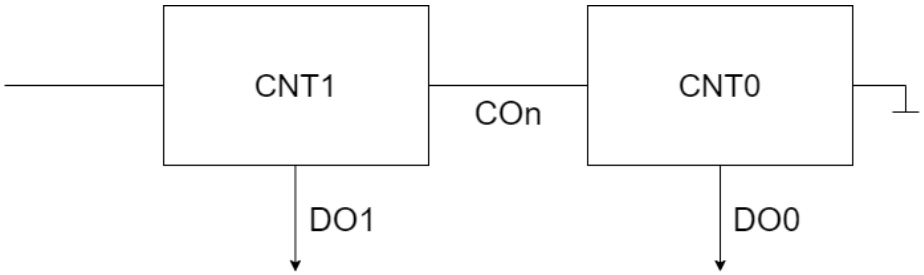
Numărătoarele

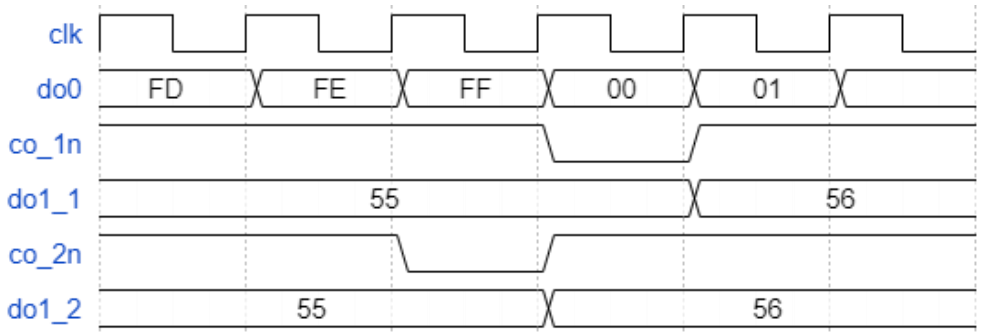
Porturi:

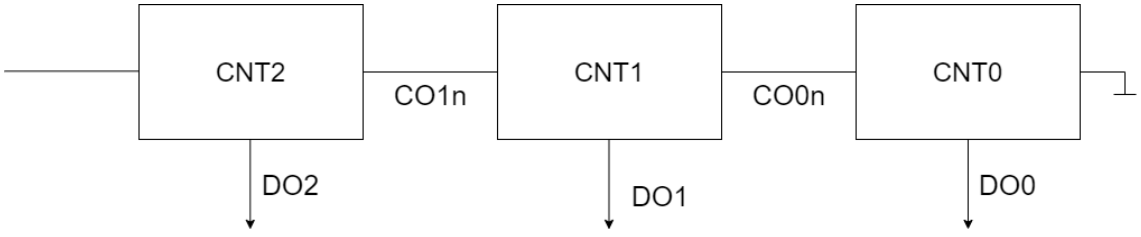


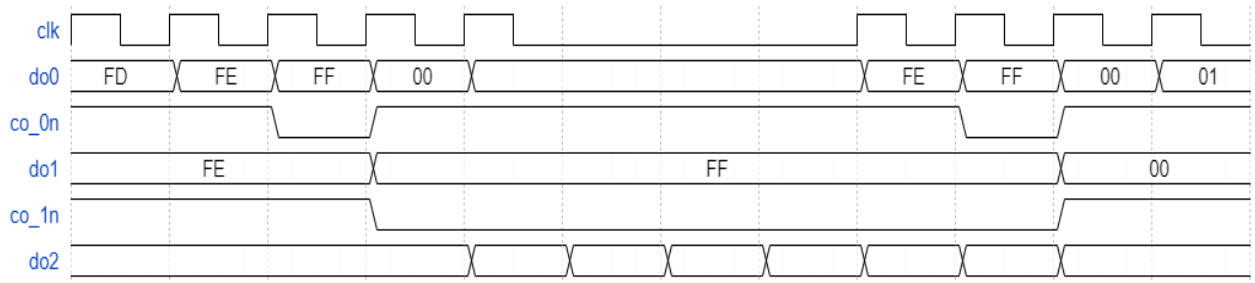
Generare COn:










clk
do0
FD
FE
FF
00
FE
FF
00
01
co_0n
do1
FE
FF
00
co_1n
do2