Laborator AC Săptămâna 7

Planificatorul de mesaje pentru o aplicaţie criptografică

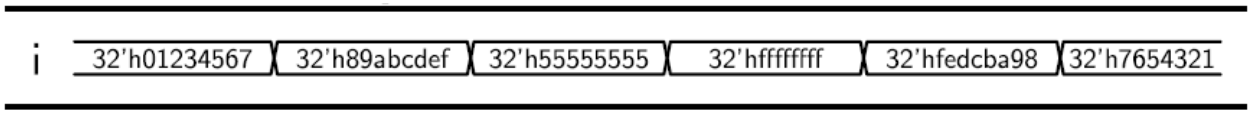
P.7.1 Implementați, folosind Verilog, operatorul $\sigma 0^{\{256\}}$ (*) în conformitate cu ecuația dată mai jos ca referință:

$$\sigma_0^{\{256\}}(x) = \boldsymbol{ROTR^7}(x) \oplus \boldsymbol{ROTR^{18}}(x) \oplus \boldsymbol{SHR^3}$$

***Soluție**:*

```
module ssgm_0 (

input [31:0] i , // cuvântul inițial

output [31:0] o // cuvântul generat

);

assign o = {i[6:0], i[31:7]} ^ {i[17:0], i[31:18]} ^ {3'd0, i[31:3]};

endmodule
```

P.7.2 Testați modulul ***ssgm_0*** cu ajutorul unui testbench de verificare non-exhaustivă construit conform cronogramei :



Soluție:

```
module ssgm_0_tb (

output reg [31:0] i , // cuvântul inițial

output [31:0] o // cuvântul generat

);

ssgm_0 sigma0 ( .i(i), .o(o) );

initial begin

i=32'h01234567;

#20 i=32'h89abcdef;

#20 i=32'h55555555;

#20 i=32'hffffffff;

#20 i=32'hfedcba98;

#20 i=32'h76543210;

end
```

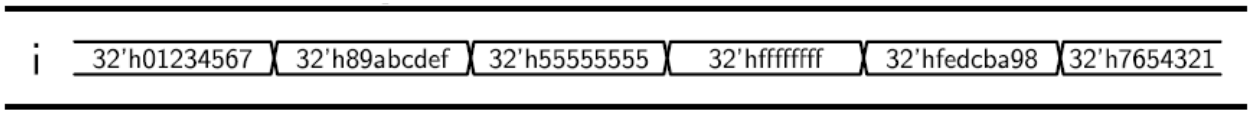
P.7.3 Implementați, folosind Verilog, operatorul $\sigma 1^{\{256\}}$ (*) în conformitate cu ecuația dată mai jos ca referință:

$$\boldsymbol{\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}}$$

Soluție:

```
module ssgm_1 (

input [31:0] i , // cuvântul inițial

output [31:0] o // cuvântul generat

);

assign o = {i[16:0], i[31:17]} ^ {i[18:0], i[31:19]} ^ {10'd0, i[31:10]};

endmodule
```

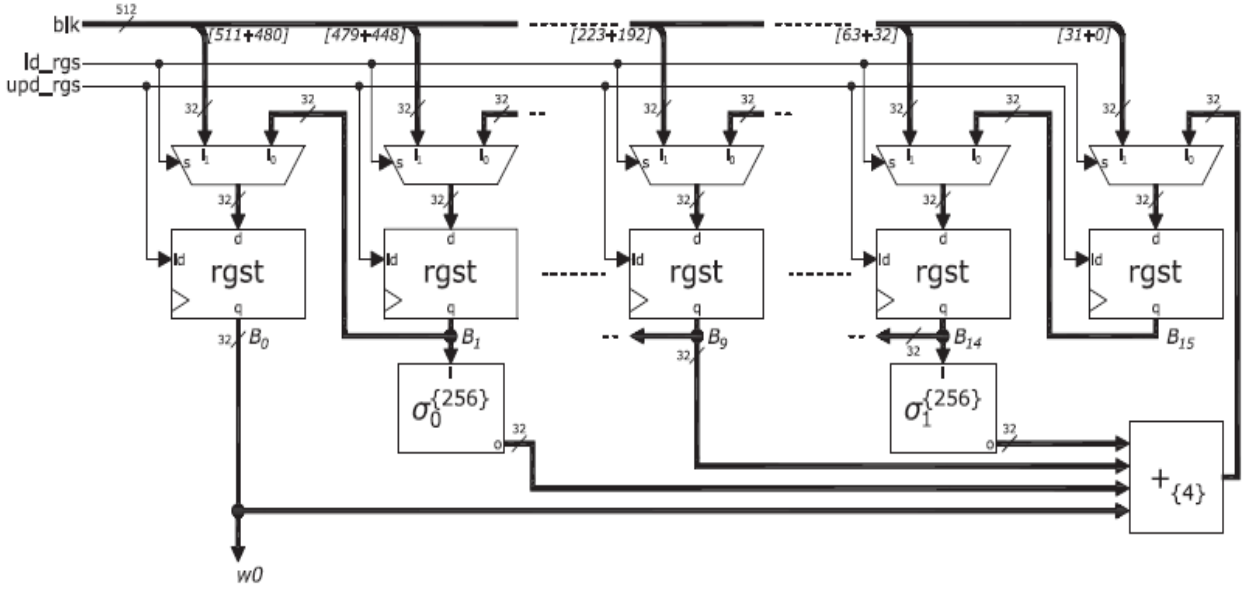
P.7.4 Verificați modulul ***ssgm_1*** cu ajutorul unui testbench de verificare non-exhaustivă construit conform cronogramei:



Solution:

```
module ssgm_1_tb (

output reg [31:0] i , // cuvântul inițial

output [31:0] o // cuvântul generat

);

ssgm_1 sigma1 ( .i(i), .o(o) );

initial begin

i=32'h01234567;

#20 i=32'h89abcdef;

#20 i=32'h55555555;

#20 i=32'hffffffff;

#20 i=32'hfedcba98;

#20 i=32'h76543210;

end
```

P.7.5 Construiți, folosind Verilog, arhitectura dată mai jos, a planificatorului de mesaje care prelucrează blocul pe 512-biți, livrând la ieșirea sa cuvântul cel mai semnificativ W0.




```
module msg_sch (

input clk ,

input rst_b ,

input [511:0] blk ,

input ld_rgs ,

input upd_rgs ,

output [31:0] w0

);
```

```
wire [511:0] a, b;

wire [31:0] s0, s1, s;

assign s = b[511:480] + b0 + b[223:192] + s1;

mux # (.w(32)) multiplexers[15:0] (.i1(blk),
.i0({b[479:0],s}), .s(ld_rgs), .o(a));

rgst # (.w(32)) registers[15:0] (.clk(clk), .rst_b(rst_b),

.d(a), .ld(upd_rgs), clr(1'd0), .q(b));

ssgm_0 sigma0 (.i(b[479:448]), .o(s0));

ssgm_1 sigma1 (.i(b[63:32], .o(s1));

assign wo = b[511:480];

endmodule
```