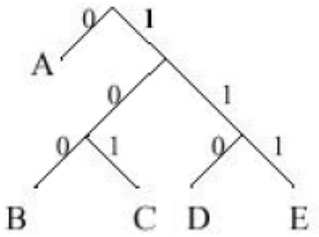
Problema:

Sa se proiecteze un sistem care decodifica un sir de biti conform arborelui prezentat in continuare (bazat pe probabilitatile de aparitie):



Se cere:

1) Descrierea solutiei in limbaj natural
2) Modul verilog ce implementeaza algoritmul
3) Sa se prezinte formele de unda pentru semnalele aferente modulului de test urmator:

```
`timescale 1ns / 1ps
module test;

	// Inputs
	reg clk;
	reg clr;
	reg din;

	// Outputs
	wire valid;
	wire [2:0] dout;

	// Instantiate the Unit Under Test (UUT)
	rezolvare uut (
		.clk(clk),
		.clr(clr),
		.din(din),
		.valid(valid),
		.dout(dout)
	);

	always #5 clk <= ~clk;

	initial begin
		// Initialize Inputs
		clk = 0;
		clr = 0;
		din = 0; //"01000111101"
		#10;
		clr = 1;
```

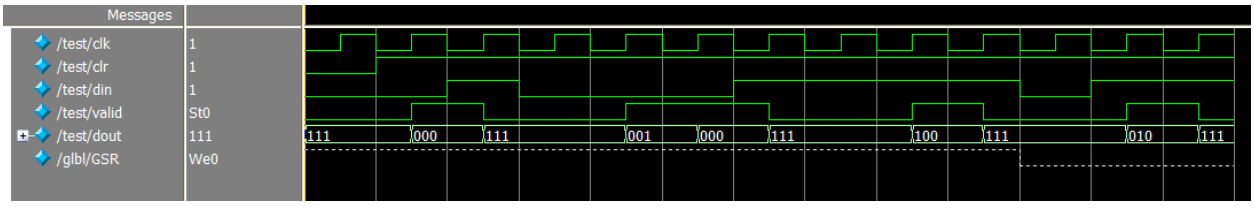
```
                #10;
                din = 1;
                #10;
                din = 0;
                #30;
                din = 1;
                #40;
                din = 0;
                #10;
                din = 1;
                #20;
                $stop;
        end

endmodule
```

Rezolvare (modulul verilog):

```
`timescale 1ns / 1ps
module rezolvare(
   input clk,
   input clr,
   input din,
   output reg valid,
   output reg [2:0] dout
   );

        localparam A=3'd0;
        localparam B=3'd1;
        localparam C=3'd2;
        localparam D=3'd3;
        localparam E=3'd4;
        localparam invalid=3'd7;

        reg [1:0] state;

        always@(posedge clk or negedge clr)
                if (~clr) begin
                        state <= 0;
                        valid <= 0;
                        dout <= invalid;
                end else case (state)
                        2'd0 : if (din) begin
                                                valid <= 0;
                                                dout <= invalid;
                                                state <= 1;
                                        end else begin
                                                valid <= 1;
```

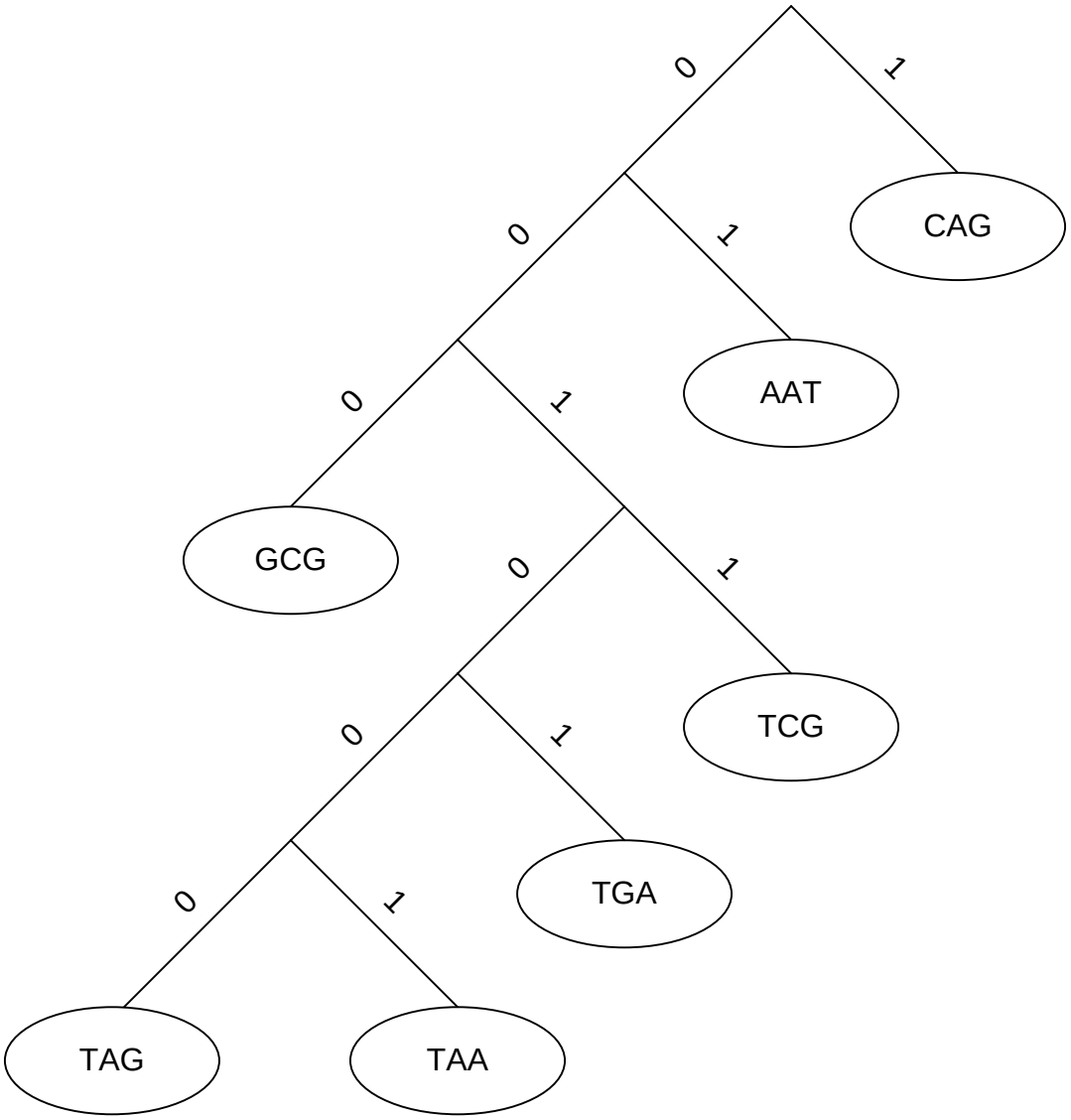
```
                    dout <= A;
                    state <= 0;
                end
        2'd1 : if (din) begin
                    valid <= 0;
                    dout <= invalid;
                    state <= 3;
                end else begin
                    valid <= 0;
                    dout <= invalid;
                    state <= 2;
                end
        2'd2 : if (din) begin
                    valid <= 1;
                    dout <= C;
                    state <= 0;
                end else begin
                    valid <= 1;
                    dout <= B;
                    state <= 0;
                end
        2'd3 : if (din) begin
                    valid <= 1;
                    dout <= E;
                    state <= 0;
                end else begin
                    valid <= 1;
                    dout <= D;
                    state <= 0;
                end
    endcase

endmodule
```

Formele de unda:



Problema:

In ADN-ul uman se gasesc 64 de codoni (secvente de trei nucleotide ce codifica un aminoacid specific – nucleotidele sunt A, C, G si T) cu probabilitati de aparitie diferite.
Sa se proiecteze un sistem care decodifica un sir de biti conform arborelui prezentat in continuare (subarbore al arborelui pentru ADN-ul uman):



Se cere:

1) Descrierea solutiei in limbaj natural
2) Modul verilog ce implementeaza algoritmul
3) Sa se prezinte formele de unda pentru semnalele aferente modulului de test urmator:

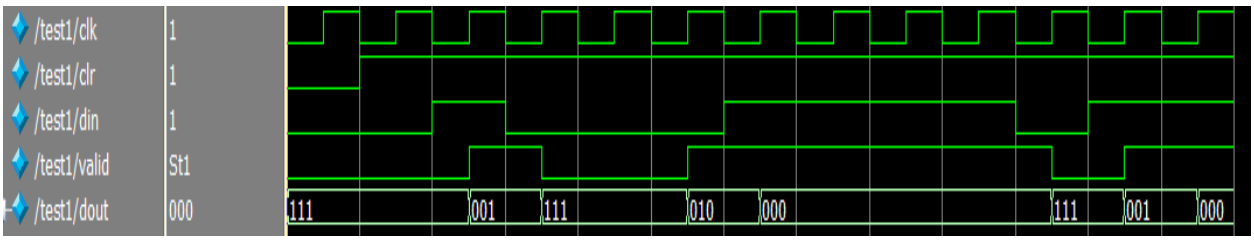
```
`timescale 1ns / 1ps
module test;

	// Inputs
	reg clk;
	reg clr;
	reg din;

	// Outputs
	wire valid;
	wire [2:0] dout;

	// Instantiate the Unit Under Test (UUT)
	rezolvare uut (
		.clk(clk),
		.clr(clr),
		.din(din),
		.valid(valid),
		.dout(dout)
	);

	always #5 clk <= ~clk;

	initial begin
		// Initialize Inputs
		clk = 0;
		clr = 0;
		din = 0; //"01000111101"
		#10;
		clr = 1;
		#10;
		din = 1;
		#10;
		din = 0;
		#30;
		din = 1;
		#40;
		din = 0;
		#10;
		din = 1;
		#20;
		$stop;
	end

endmodule
```

Rezolvare (modulul verilog):

```
`timescale 1ns / 1ps
module rezolvare(
    input clk,
    input clr,
    input din,
    output reg valid,
    output reg [2:0] dout
    );

        localparam CAG=3'd0;
        localparam AAT=3'd1;
        localparam GCG=3'd2;
        localparam TCG=3'd3;
        localparam TGA=3'd4;
        localparam TAG=3'd5;
        localparam TAA=3'd6;
        localparam invalid=3'd7;

        reg [2:0] state;

        always@(posedge clk or negedge clr)
            if (~clr) begin
                state <= 0;
                valid <= 0;
                dout <= invalid;
            end else case (state)
                3'd0 : if (din) begin
                                valid <= 1;
                                dout <= CAG;
                                state <= 0;
                            end else begin
                                valid <= 0;
                                dout <= invalid;
                                state <= 1;
                            end
                3'd1 : if (din) begin
                                valid <= 1;
                                dout <= AAT;
                                state <= 0;
                            end else begin
                                valid <= 0;
                                dout <= invalid;
                                state <= 2;
                            end
                3'd2 : if (din) begin
                                valid <= 0;
```

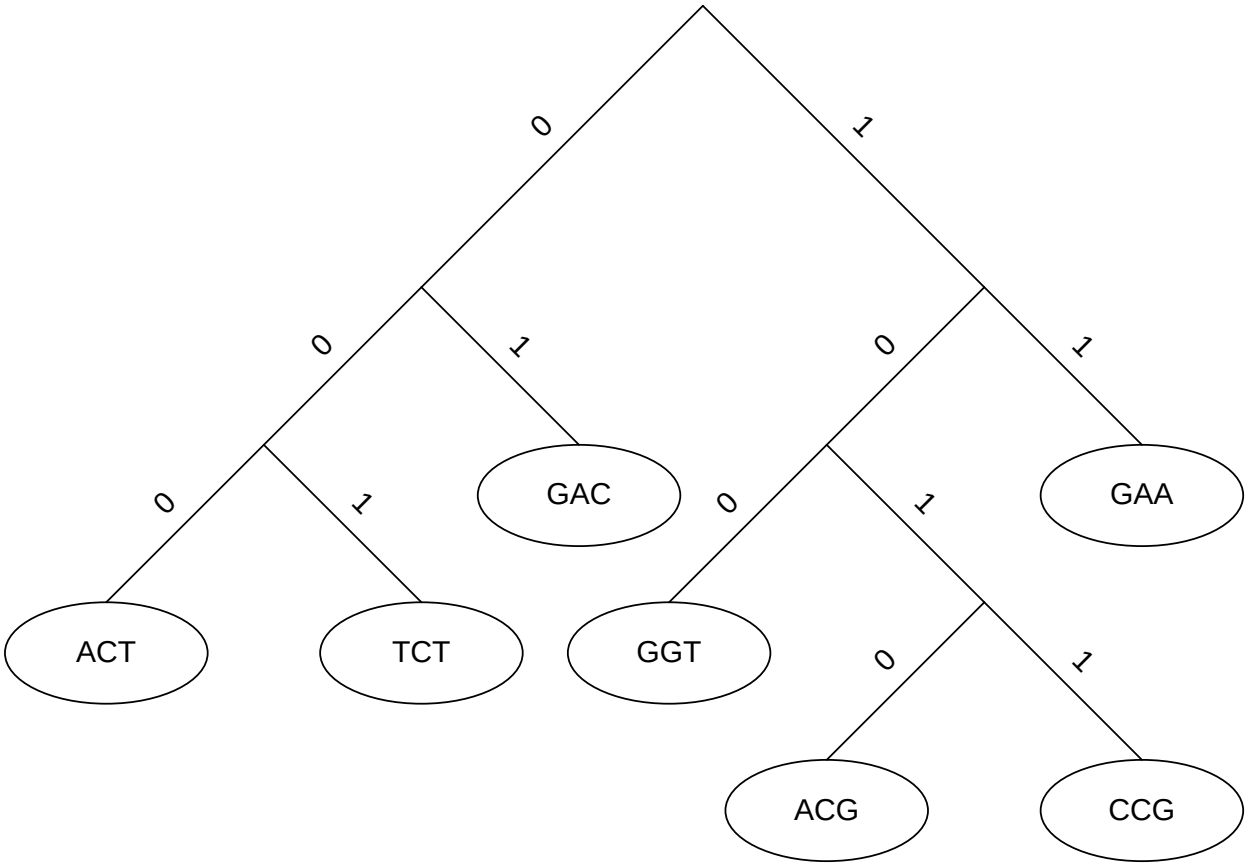
```
                                        dout <= invalid;
                                        state <= 3;
                                    end else begin
                                        valid <= 1;
                                        dout <= GCG;
                                        state <= 0;
                                    end
                3'd3 : if (din) begin
                                        valid <= 1;
                                        dout <= TCG;
                                        state <= 0;
                                    end else begin
                                        valid <= 0;
                                        dout <= invalid;
                                        state <= 4;
                                    end
                3'd4 : if (din) begin
                                        valid <= 1;
                                        dout <= TGA;
                                        state <= 0;
                                    end else begin
                                        valid <= 0;
                                        dout <= invalid;
                                        state <= 5;
                                    end
                3'd5 : if (din) begin
                                        valid <= 1;
                                        dout <= TAA;
                                        state <= 0;
                                    end else begin
                                        valid <= 1;
                                        dout <= TAG;
                                        state <= 0;
                                    end
        endcase

endmodule
```

Formele de unda:



Problema:

In ADN-ul uman se gasesc 64 de codoni (secvente de trei nucleotide ce codifica un aminoacid specific – nucleotidele sunt A, C, G si T) cu probabilitati de aparitie diferite.
Sa se proiecteze un sistem care decodifica un sir de biti conform arborelui prezentat in continuare (subarbore al arborelui pentru ADN-ul uman):



Se cere:

1) Descrierea solutiei in limbaj natural
2) Modul verilog ce implementeaza algoritmul
3) Sa se prezinte formele de unda pentru semnalele aferente modulului de test urmator:

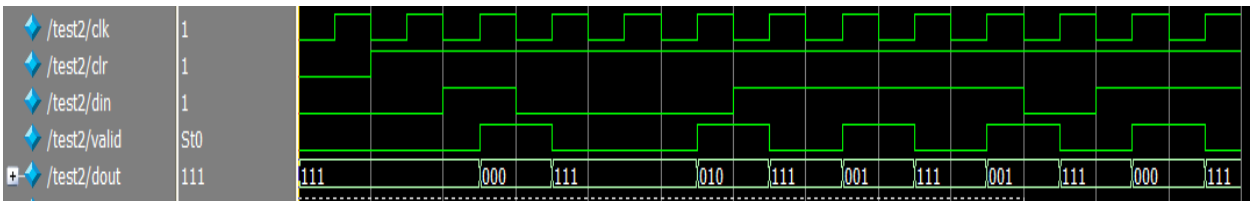
```
`timescale 1ns / 1ps
module test;

	// Inputs
	reg clk;
	reg clr;
	reg din;

	// Outputs
	wire valid;
	wire [2:0] dout;

	// Instantiate the Unit Under Test (UUT)
	rezolvare uut (
		.clk(clk),
		.clr(clr),
		.din(din),
		.valid(valid),
		.dout(dout)
	);

	always #5 clk <= ~clk;

	initial begin
		// Initialize Inputs
		clk = 0;
		clr = 0;
		din = 0; //"01000111101"
		#10;
		clr = 1;
		#10;
		din = 1;
		#10;
		din = 0;
		#30;
		din = 1;
		#40;
		din = 0;
		#10;
		din = 1;
		#20;
		$stop;
	end

endmodule
```

Rezolvare (modulul verilog):

```
`timescale 1ns / 1ps
module rezolvare(
    input clk,
    input clr,
    input din,
    output reg valid,
    output reg [2:0] dout
    );

        localparam GAC=3'd0;
        localparam GAA=3'd1;
        localparam ACT=3'd2;
        localparam TCT=3'd3;
        localparam GGT=3'd4;
        localparam ACG=3'd5;
        localparam CCG=3'd6;
        localparam invalid=3'd7;

        reg [2:0] state;

        always@(posedge clk or negedge clr)
            if (~clr) begin
                state <= 0;
                valid <= 0;
                dout <= invalid;
            end else case (state)
                3'd0 : if (din) begin
                                valid <= 0;
                                dout <= invalid;
                                state <= 2;
                            end else begin
                                valid <= 0;
                                dout <= invalid;
                                state <= 1;
                            end
                3'd1 : if (din) begin
                                valid <= 1;
                                dout <= GAC;
                                state <= 0;
                            end else begin
                                valid <= 0;
                                dout <= invalid;
                                state <= 3;
                            end
                3'd2 : if (din) begin
                                valid <= 1;
```

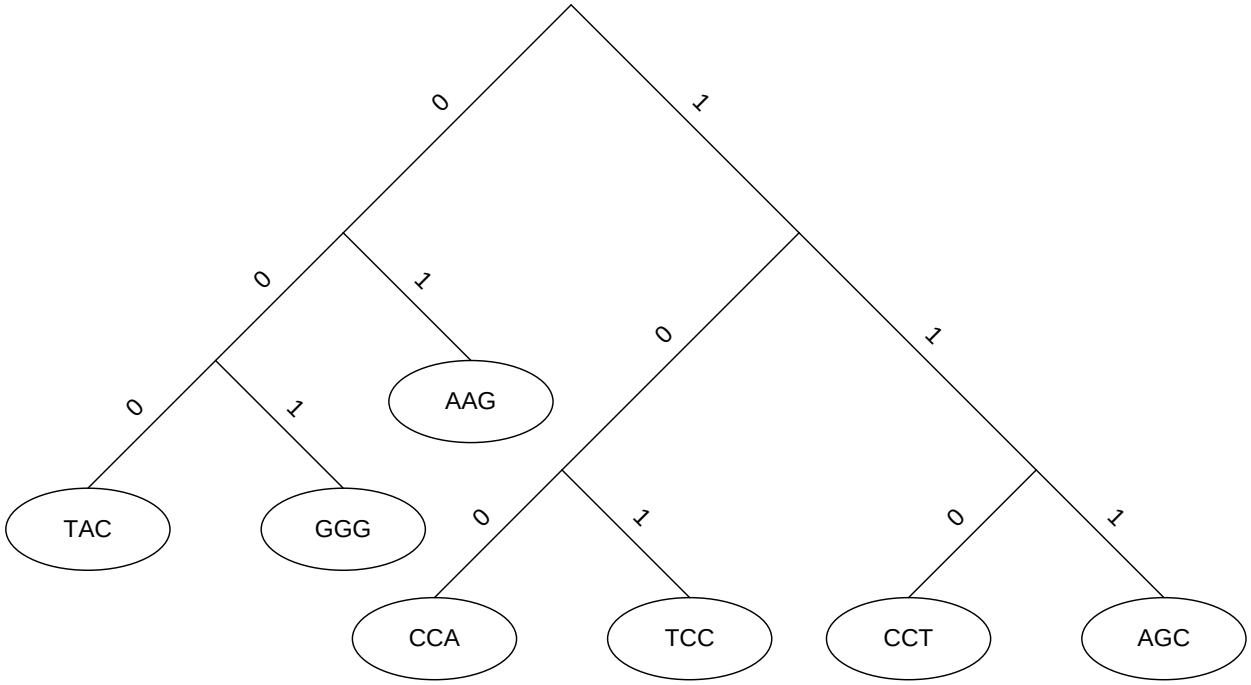
```
                                        dout <= GAA;
                                        state <= 0;
                                    end else begin
                                        valid <= 0;
                                        dout <= invalid;
                                        state <= 4;
                                    end
                            3'd3 : if (din) begin
                                        valid <= 1;
                                        dout <= TCT;
                                        state <= 0;
                                    end else begin
                                        valid <= 1;
                                        dout <= ACT;
                                        state <= 0;
                                    end
                            3'd4 : if (din) begin
                                        valid <= 0;
                                        dout <= invalid;
                                        state <= 5;
                                    end else begin
                                        valid <= 1;
                                        dout <= GGT;
                                        state <= 0;
                                    end
                            3'd5 : if (din) begin
                                        valid <= 1;
                                        dout <= CCG;
                                        state <= 0;
                                    end else begin
                                        valid <= 1;
                                        dout <= ACG;
                                        state <= 0;
                                    end
                    endcase

endmodule
```

Formele de unda:



Problema:

In ADN-ul uman se gasesc 64 de codoni (secvente de trei nucleotide ce codifica un aminoacid specific – nucleotidele sunt A, C, G si T) cu probabilitati de aparitie diferite.
Sa se proiecteze un sistem care decodifica un sir de biti conform arborelui prezentat in continuare (subarbore al arborelui pentru ADN-ul uman):



Se cere:

1) Descrierea solutiei in limbaj natural
2) Modul verilog ce implementeaza algoritmul
3) Sa se prezinte formele de unda pentru semnalele aferente modulului de test urmator:

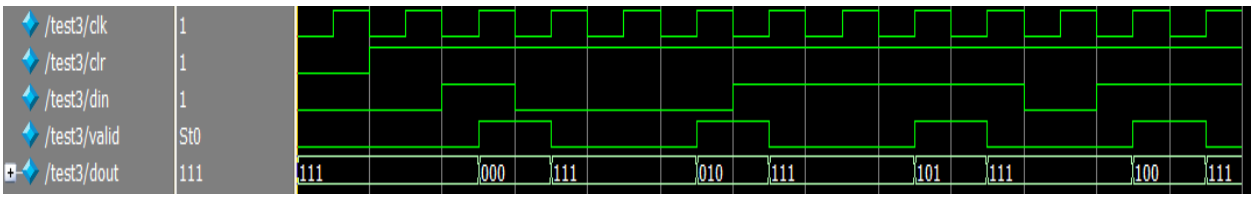
```
`timescale 1ns / 1ps
module test;

	// Inputs
	reg clk;
	reg clr;
	reg din;

	// Outputs
	wire valid;
	wire [2:0] dout;

	// Instantiate the Unit Under Test (UUT)
	rezolvare uut (
		.clk(clk),
		.clr(clr),
		.din(din),
		.valid(valid),
		.dout(dout)
	);

	always #5 clk <= ~clk;

	initial begin
		// Initialize Inputs
		clk = 0;
		clr = 0;
		din = 0; //"01000111101"
		#10;
		clr = 1;
		#10;
		din = 1;
		#10;
		din = 0;
		#30;
		din = 1;
		#40;
		din = 0;
		#10;
		din = 1;
		#20;
		$stop;
	end

endmodule
```

Rezolvare (modulul verilog):

```
`timescale 1ns / 1ps
module rezolvare3(
   input clk,
   input clr,
   input din,
   output reg valid,
   output reg [2:0] dout
   );

            localparam AAG=3'd0;
            localparam TAC=3'd1;
            localparam GGG=3'd2;
            localparam CCA=3'd3;
            localparam TCC=3'd4;
            localparam CCT=3'd5;
            localparam AGC=3'd6;
            localparam invalid=3'd7;

            reg [2:0] state;

            always@(posedge clk or negedge clr)
                  if (~clr) begin
                              state <= 0;
                              valid <= 0;
                              dout <= invalid;
                  end else case (state)
                              3'd0 : if (din) begin
                                                      valid <= 0;
                                                      dout <= invalid;
                                                      state <= 2;
                                          end else begin
                                                      valid <= 0;
                                                      dout <= invalid;
                                                      state <= 1;
                                          end
                              3'd1 : if (din) begin
                                                      valid <= 1;
                                                      dout <= AAG;
                                                      state <= 0;
                                          end else begin
                                                      valid <= 0;
                                                      dout <= invalid;
                                                      state <= 3;
                                          end
                              3'd2 : if (din) begin
                                                      valid <= 0;
```

```
                                        dout <= invalid;
                                        state <= 5;
                                    end else begin
                                        valid <= 0;
                                        dout <= invalid;
                                        state <= 4;
                                    end
                            3'd3 : if (din) begin
                                        valid <= 1;
                                        dout <= TAC;
                                        state <= 0;
                                    end else begin
                                        valid <= 1;
                                        dout <= GGG;
                                        state <= 0;
                                    end
                            3'd4 : if (din) begin
                                        valid <= 1;
                                        dout <= TCC;
                                        state <= 0;
                                    end else begin
                                        valid <= 1;
                                        dout <= CCA;
                                        state <= 0;
                                    end
                            3'd5 : if (din) begin
                                        valid <= 1;
                                        dout <= CCT;
                                        state <= 0;
                                    end else begin
                                        valid <= 1;
                                        dout <= AGC;
                                        state <= 0;
                                    end
                    endcase

endmodule
```

Formele de unda:



Problema:

Sa se proiecteze un sistem care codifica un character ASCII (reprezentat pe 7 biti) generand cei 11 biti ai codului Hamming corespunzator.

Se cere:

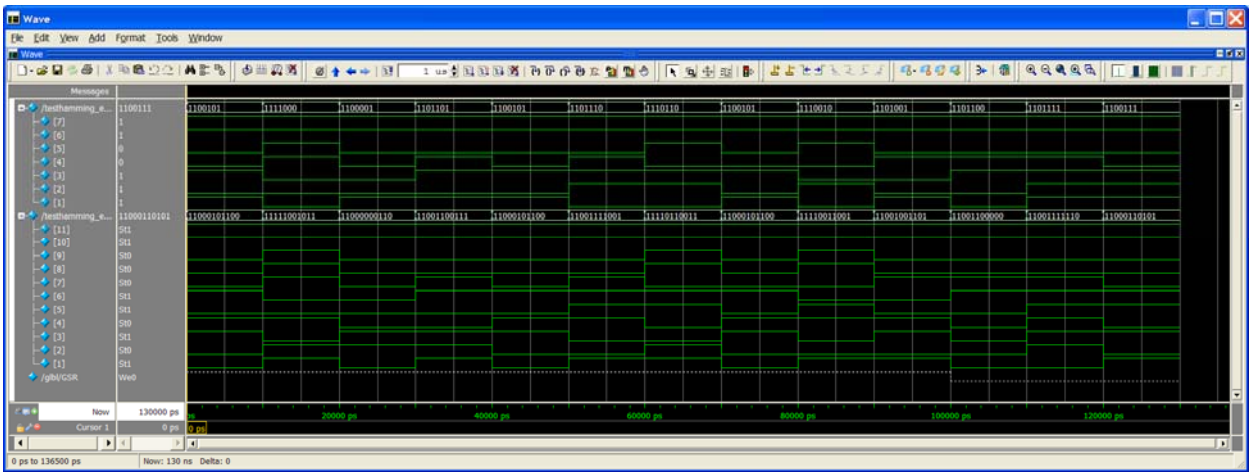
1) Descrierea solutiei in limbaj natural
2) Modul verilog ce implementeaza algoritmul
3) Sa se prezinte formele de unda pentru semnalele aferente modulului de test urmator:

```
`timescale 1ns / 1ps
module testhamming_enc;
	reg [7:1] c;
	wire [11:1] h;

	hamming_enc uut (.c(c), .h(h));

	initial begin  //examenverilog
		c=7'b1100101;
		#10;
		c=7'b1111000;
		#10;
		c=7'b1100001;
		#10;
		c=7'b1101101;
		#10;
		c=7'b1100101;
		#10;
		c=7'b1101110;
		#10;
		c=7'b1110110;
		#10;
		c=7'b1100101;
		#10;
		c=7'b1110010;
		#10;
		c=7'b1101001;
		#10;
		c=7'b1101100;
		#10;
		c=7'b1101111;
		#10;
		c=7'b1100111;
		#10;
		$stop;
	end
endmodule
```

Rezolvare (modulul verilog):

```
`timescale 1ns / 1ps
module hamming_enc(
    input [7:1] c,
    output [11:1] h
    );

            wire p1,p2,p4,p8;

            assign p1=d[1]^^d[2]^^d[4]^^d[5]^^d[7];
            assign p2=d[1]^^d[3]^^d[4]^^d[6]^^d[7];
            assign p4=d[2]^^d[3]^^d[4];
            assign p8=d[5]^^d[6]^^d[7];
            assign h={d[7:5],p8,d[4:2],p4,d1,p2,p1};

endmodule
```

Formele de unda:



Problema:

Sa se proiecteze un sistem care decodifica un cod Hamming pe 11 biti si obtine caracterul corect ASCII (reprezentat pe 7 biti) si semnalarea eventualei erori detectate.

Se cere:

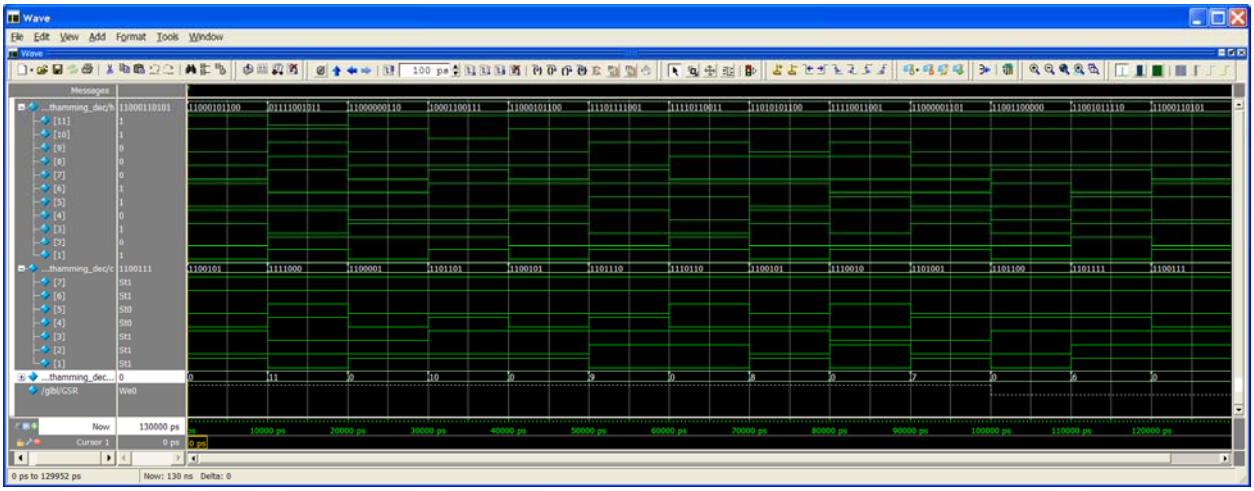
1) Descrierea solutiei in limbaj natural
2) Modul verilog ce implementeaza algoritmul
3) Sa se prezinte formele de unda pentru semnalele aferente modulului de test urmator:

```
`timescale 1ns / 1ps
module testhamming_dec;
	reg [11:1] h;
	wire [7:1] c;
	wire [3:0] eb;
	hamming_dec uut (.h(h), .c(c), .eb(eb));

	initial begin    //examenverilog
		h=11'b11000101100;
		#10;
		h=11'b01111001011;
		#10;
		h=11'b11000000110;
		#10;
		h=11'b10001100111;
		#10;
		h=11'b11000101100;
		#10;
		h=11'b11101111001;
		#10;
		h=11'b11110110011;
		#10;
		h=11'b11010101100;
		#10;
		h=11'b11110011001;
		#10;
		h=11'b11000001101;
		#10;
		h=11'b11001100000;
		#10;
		h=11'b11001011110;
		#10;
		h=11'b11000110101;
		#10;
		$stop;
	end
endmodule
```

Rezolvare (modulul verilog):

```
`timescale 1ns / 1ps
module hamming_dec(
    input [11:1] h,
    output [7:1] c,
    output [3:0] eb
    );

            assign eb[0]=h[1]^^h[3]^^h[5]^^h[7]^^h[9]^^h[11];
            assign eb[1]=h[2]^^h[3]^^h[6]^^h[7]^^h[10]^^h[11];
            assign eb[2]=h[4]^^h[5]^^h[6]^^h[7];
            assign eb[3]=h[8]^^h[9]^^h[10]^^h[11];

            assign c[1]=(eb==4'd3)? ~h[3] : h[3];
            assign c[2]=(eb==4'd5)? ~h[5] : h[5];
            assign c[3]=(eb==4'd6)? ~h[6] : h[6];
            assign c[4]=(eb==4'd7)? ~h[7] : h[7];
            assign c[5]=(eb==4'd9)? ~h[9] : h[9];
            assign c[6]=(eb==4'd10)? ~h[10] : h[10];
            assign c[7]=(eb==4'd11)? ~h[11] : h[11];

endmodule
```

Formele de unda:



Probleme:

1) Sa se proiecteze un automat cu stari finite Moore ce analizeaza un sir de caractere si accepta doar combinatia MURMUR. Alfabetul utilizat este: M R U

```
`timescale 1ns / 1ps
module testp1;

	localparam M=2'd0;
	localparam R=2'd1;
	localparam U=2'd2;

	// Inputs
	reg clk;
	reg clr;
	reg [1:0] din;

	// Outputs
	wire valid;

	// Instantiate the Unit Under Test (UUT)
	problema1 uut (.clk(clk), .clr(clr), .din(din), .valid(valid));

	always #5 clk <= ~clk;

	initial begin
		// Initialize Inputs
		clk = 0;
		clr = 1;
		din = M;
		#10;
		clr = 0;
		#10;
		din = U;
		#10;
		din = R;
		#10;
		din = M;
		#10;
		din = U;
		#10;
		din = R;
		#30;
		$stop;
	end

endmodule
```

2) Sa se proiecteze un automat cu stari finite Moore ce analizeaza un sir de caractere si accepta combinatia FROG cu orice prefix. Alfabetul utilizat este: F G O R

```
`timescale 1ns / 1ps
module testp2;

	localparam F=2'd0;
	localparam G=2'd1;
	localparam O=2'd2;
	localparam R=2'd3;

	// Inputs
	reg clk;
	reg clr;
	reg [1:0] din;

	// Outputs
	wire valid;

	// Instantiate the Unit Under Test (UUT)
	problema2 uut (.clk(clk), .clr(clr), .din(din), .valid(valid));

	always #5 clk <= ~clk;

	initial begin
		// Initialize Inputs
		clk = 0;
		clr = 1;
		din = F;
		#10;
		clr = 0;
		#10;
		din = R;
		#10;
		din = F;
		#10;
		din = R;
		#10;
		din = O;
		#10;
		din = G;
		#30;
		$stop;
	end

endmodule
```

3) Sa se proiecteze un automat cu stari finite Moore ce analizeaza un sir de caractere si accepta combinatiile CAT sau DOG oriunde. Alfabetul utilizat este: A C D G O T

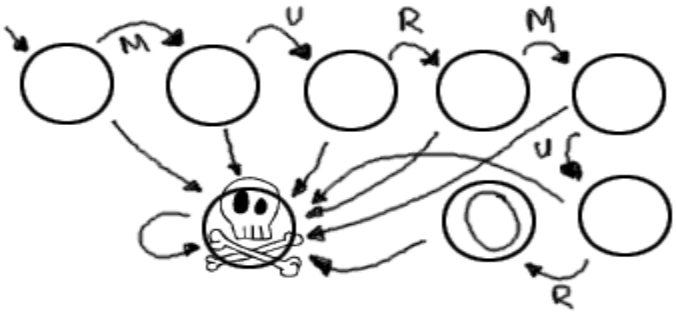
```
`timescale 1ns / 1ps
module testp3;

	localparam A=3'd0;
	localparam C=3'd1;
	localparam D=3'd2;
	localparam G=3'd3;
	localparam O=3'd4;
	localparam T=3'd5;

	// Inputs
	reg clk;
	reg clr;
	reg [2:0] din;

	// Outputs
	wire valid;

	// Instantiate the Unit Under Test (UUT)
	problema3 uut (.clk(clk), .clr(clr), .din(din), .valid(valid));

	always #5 clk <= ~clk;

	initial begin
		// Initialize Inputs
		clk = 0;
		clr = 1;
		din = D;
		#10;
		clr = 0;
		#10;
		din = C;
		#10;
		din = A;
		#10;
		din = D;
		#10;
		din = O;
		#10;
		din = G;
		#30;
		$stop;
	end

endmodule
```

Se cere:

a) Diagram de stari
b) Descrierea in verilog a automatului
c) Formele de unda obtinute cu ajutorul testului descris de codul verilog

Rezolvari:

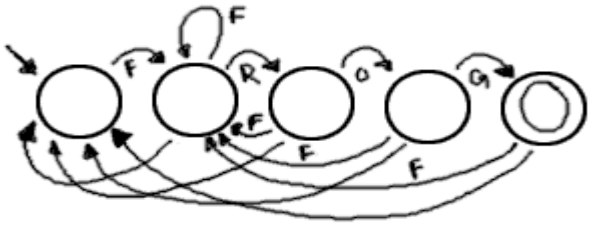
1)




```
`timescale 1ns / 1ps
module problema1(
   input clk,
   input clr,
   input [1:0] din,
   output valid
   );

            localparam M=2'd0;
            localparam R=2'd1;
            localparam U=2'd2;


            reg [2:0] state;

            assign valid = (state==6);

            always@(posedge clk or posedge clr)
                  if (clr) state <= 0;
                  else case (state)
                        3'd0 : if (din==M) state <= 1;
                                          else state <= 7;
                        3'd1 : if (din==U) state <= 2;
                                          else state <= 7;
                        3'd2 : if (din==R) state <= 3;
                                          else state <= 7;
                        3'd3 : if (din==M) state <= 4;
                                          else state <= 7;
                        3'd4 : if (din==U) state <= 5;
                                          else state <= 7;
```

```
                3'd5 : if (din==R) state <= 6;
                                   else state <= 7;
                3'd6 : state <= 7;
                3'd7 : state <= 7;
        endcase

endmodule
```

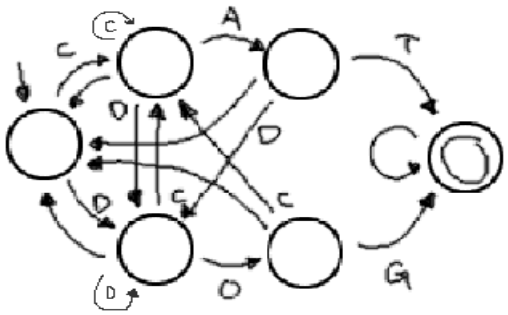
2)




```
`timescale 1ns / 1ps
module problema2(
    input clk,
    input clr,
    input [1:0] din,
    output valid
    );

        localparam F=2'd0;
        localparam G=2'd1;
        localparam O=2'd2;
        localparam R=2'd3;

        reg [2:0] state;

        assign valid = (state==4);

        always@(posedge clk or posedge clr)
            if (clr) state <= 0;
            else case (state)
                3'd0 : if (din==F) state <= 1;
                                   else state <= 0;
                3'd1 : if (din==R) state <= 2;
                                   else begin
                                    if (din==F) state <= 1;
                                    else state <= 0;
                                   end
```

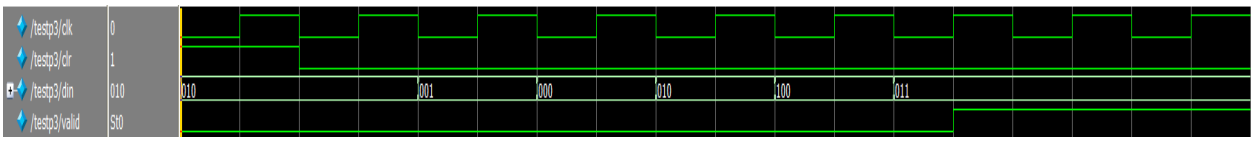
```
                3'd2 : if (din==O) state <= 3;
                                   else begin
                                    if (din==F) state <= 1;
                                    else state <= 0;
                                   end
                3'd3 : if (din==G) state <= 4;
                                   else begin
                                    if (din==F) state <= 1;
                                    else state <= 0;
                                   end
                3'd4 : if (din==F) state <= 1;
                                   else state <= 0;
                default: state <= 0;
        endcase

endmodule
```

3)




```
`timescale 1ns / 1ps
module problema3(
   input clk,
   input clr,
   input [2:0] din,
   output valid
   );

            localparam A=3'd0;
            localparam C=3'd1;
            localparam D=3'd2;
            localparam G=3'd3;
            localparam O=3'd4;
            localparam T=3'd5;
```

```
reg [2:0] state;

assign valid = (state==5);

always@(posedge clk or posedge clr)
        if (clr) state <= 0;
        else case (state)
                3'd0 : if (din==C) state <= 1;
                                   else begin
                                    if (din==D) state <= 2;
                                    else state <= 0;
                                   end
                3'd1 : if (din==A) state <= 3;
                                   else begin
                                    if (din==D) state <= 2;
                                    else begin
                                         if (din==C) state <= 1;
                                         else state <= 0;
                                        end
                                   end
                3'd2 : if (din==O) state <= 4;
                                   else begin
                                    if (din==C) state <= 1;
                                    else begin
                                         if (din==D) state <= 2;
                                         else state <= 0;
                                        end
                                   end
                3'd3 : if (din==T) state <= 5;
                                   else begin
                                    if (din==D) state <= 2;
                                    else state <= 0;
                                   end
                3'd4 : if (din==G) state <= 5;
                                   else begin
                                    if (din==C) state <= 1;
                                    else state <= 0;
                                   end
                3'd5 : state <= 5;
                default: state <= 0;
        endcase

endmodule
```



Problema:

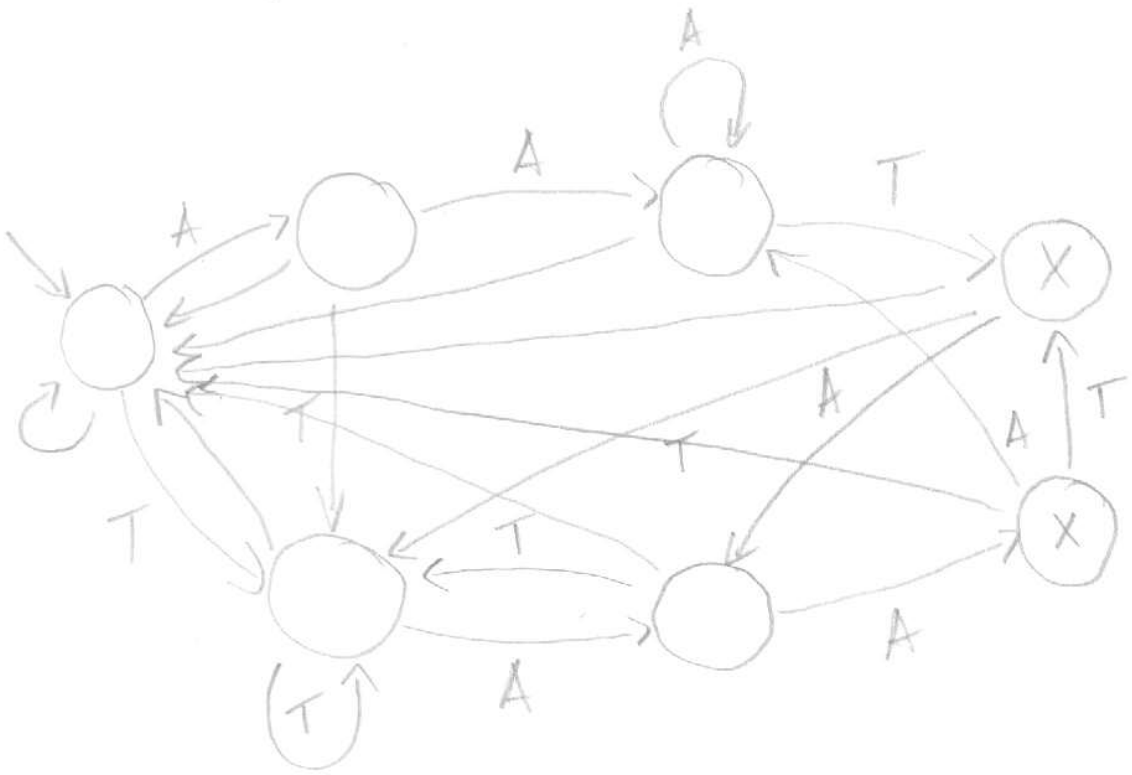
Sa se proiecteze un automat cu stari finite Moore ce analizeaza o secventa ADN formata din cele patru nucleotide (A, C, G si T) si identifica aparitia codonilor (secvente de trei nucleotide ce codifica un aminoacid specific): AAT si TAA

Se cere:
a) Diagram de stari
b) Descrierea in verilog a automatului
c) Formele de unda obtinute cu ajutorul testului descris de codul verilog

```
`timescale 1ns / 1ps
module testpe1;
	localparam A=2'd0;
	localparam C=2'd1;
	localparam G=2'd2;
	localparam T=2'd3;

	// Inputs
	reg clk;
	reg clr;
	reg [1:0] din;

	// Outputs
	wire valid;

	// Instantiate the Unit Under Test (UUT)
	pe1 uut (.clk(clk), .clr(clr), .din(din), .valid(valid));

	always #5 clk <= ~clk;

	initial begin
		// Initialize Inputs
		clk = 0;
		clr = 1;
		din = A;
		#10;
		clr = 0;
		#10;
		din = T;
		#10;
		din = A;
		#30;
		din = T;
		#10;
		din = C;
		#10;
		din = A;
		#40;
		din = T;
		#10;
		$stop;
	end
endmodule
```

Rezolvare:

a)

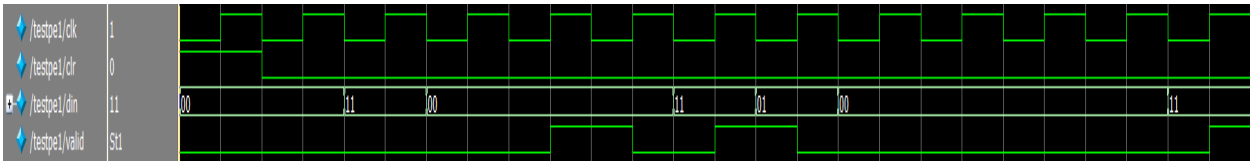


b)

```
`timescale 1ns / 1ps
module pe1(
   input clk,
   input clr,
   input [1:0] din,
   output valid
   );

        localparam A=2'd0;
        localparam C=2'd1;
        localparam G=2'd2;
        localparam T=2'd3;

        reg [2:0] state;

        assign valid = (state==5)||(state==6);

        always@(posedge clk or posedge clr)
            if (clr) state <= 0;
            else case (state)
                3'd0 : if (din==A) state <= 1;
                                   else if (din==T) state <= 2;
                                   else state <= 0;
                3'd1 : if (din==A) state <= 3;
                                   else if (din==T) state <= 2;
                                   else state <= 0;
```

```
                3'd2 : if (din==A) state <= 4;
                                else if (din==T) state <= 2;
                                else state <= 0;
                3'd3 : if (din==T) state <= 5;
                                else if (din==A) state <= 3;
                                else state <= 0;
                3'd4 : if (din==A) state <= 6;
                                else if (din==T) state <= 2;
                                else state <= 0;
                3'd5 : if (din==A) state <= 4;
                                else if (din==T) state <= 2;
                                else state <= 0;
                3'd6 : if (din==A) state <= 3;
                                else if (din==T) state <= 5;
                                else state <= 0;
                default: state <= 0;
        endcase

endmodule
```

c)



Problema:

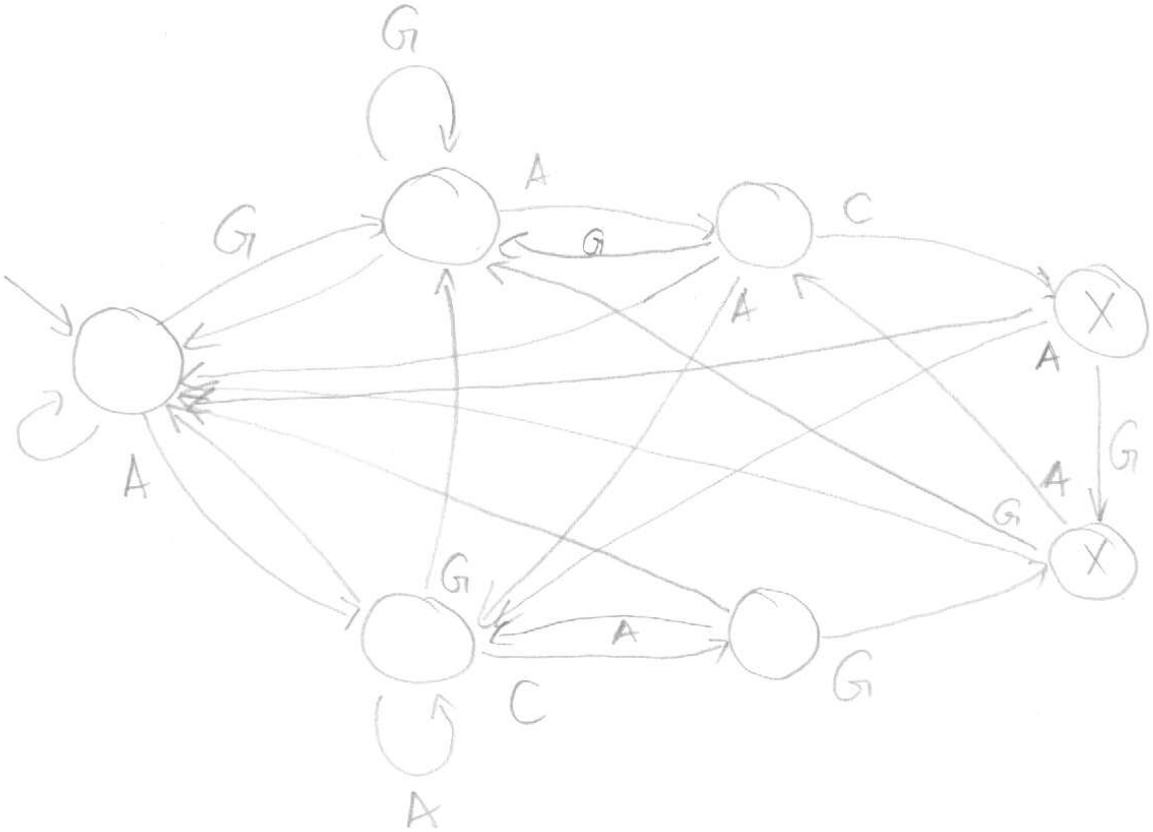
Sa se proiecteze un automat cu stari finite Moore ce analizeaza o secventa ADN formata din cele patru nucleotide (A, C, G si T) si identifica aparitia codonilor (secvente de trei nucleotide ce codifica un aminoacid specific): GAC si ACG

Se cere:
a) Diagram de stari
b) Descrierea in verilog a automatului
c) Formele de unda obtinute cu ajutorul testului descris de codul verilog

```
`timescale 1ns / 1ps
module testpe2;
	localparam A=2'd0;
	localparam C=2'd1;
	localparam G=2'd2;
	localparam T=2'd3;

	// Inputs
	reg clk;
	reg clr;
	reg [1:0] din;

	// Outputs
	wire valid;

	// Instantiate the Unit Under Test (UUT)
	pe2 uut (.clk(clk), .clr(clr), .din(din), .valid(valid));

	always #5 clk <= ~clk;

	initial begin
		// Initialize Inputs
		clk = 0;
		clr = 1;
		din = A;
		#10;
		clr = 0;
		#10;
		din = G;
		#10;
		din = A;
		#30;
		din = C;
		#10;
		din = G;
		#40;
		din = A;
		#10;
		din = C;
		#10;
		$stop;
	end
endmodule
```

Rezolvare:

a)

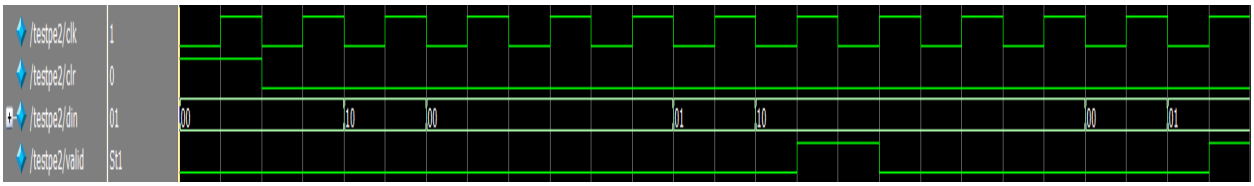


b)

```
`timescale 1ns / 1ps
module pe2(
   input clk,
   input clr,
   input [1:0] din,
   output valid
   );

            localparam A=2'd0;
            localparam C=2'd1;
            localparam G=2'd2;
            localparam T=2'd3;

            reg [2:0] state;

            assign valid = (state==5)||(state==6);

            always@(posedge clk or posedge clr)
                  if (clr) state <= 0;
                  else case (state)
                        3'd0 : if (din==G) state <= 1;
                                          else if (din==A) state <= 2;
                                          else state <= 0;
```

```
                3'd1 : if (din==A) state <= 3;
                                   else if (din==G) state <= 1;
                                   else state <= 0;
                3'd2 : if (din==A) state <= 2;
                                   else if (din==G) state <= 1;
                                   else if (din==C) state <= 4;
                                   else state <= 0;
                3'd3 : if (din==C) state <= 5;
                                   else if (din==G) state <= 1;
                                   else if (din==A) state <= 2;
                                   else state <= 0;
                3'd4 : if (din==G) state <= 6;
                                   else if (din==A) state <= 2;
                                   else state <= 0;
                3'd5 : if (din==A) state <= 2;
                                   else if (din==G) state <= 6;
                                   else state <= 0;
                3'd6 : if (din==A) state <= 3;
                                   else if (din==G) state <= 1;
                                   else state <= 0;
                default: state <= 0;
        endcase

endmodule
```

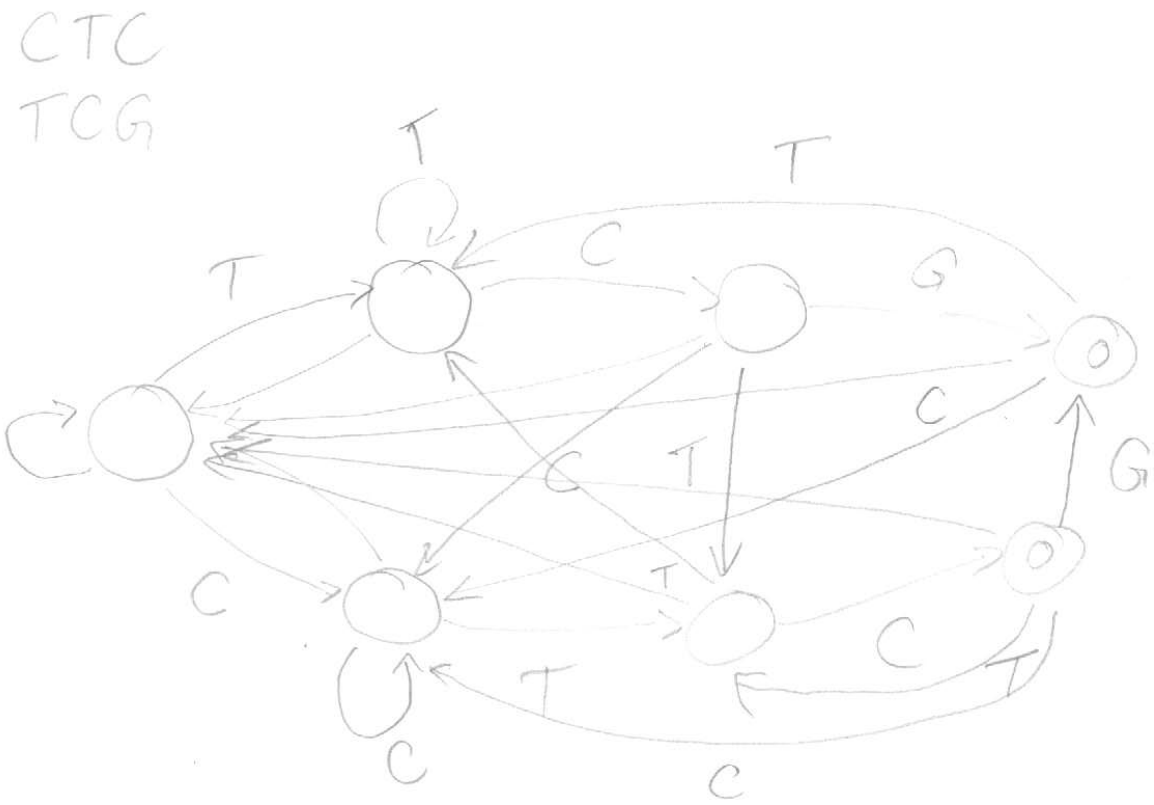
c)



Problema:
Sa se proiecteze un automat cu stari finite Moore ce analizeaza o secventa ADN formata din cele patru nucleotide (A, C, G si T) si identifica aparitia codonilor (secvente de trei nucleotide ce codifica un aminoacid specific): CTC si TCG

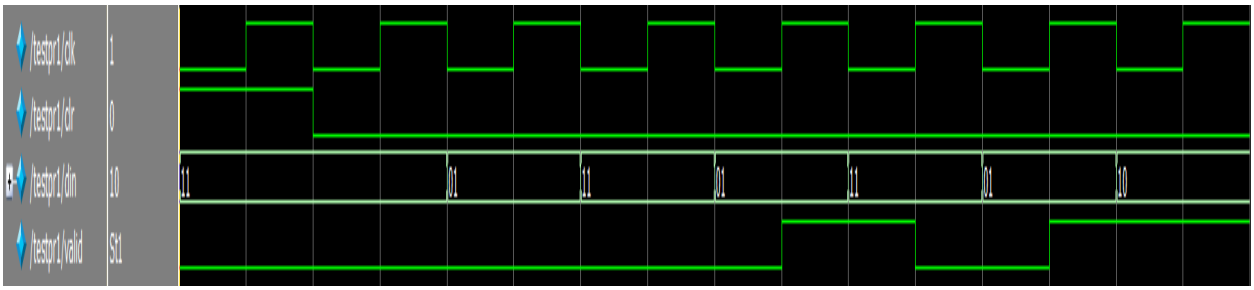
Se cere:
a) Diagram de stari
b) Descrierea in verilog a automatului
c) Formele de unda obtinute cu ajutorul testului descris de codul verilog

```
`timescale 1ns / 1ps
module testpr1;
	localparam A=2'd0;
	localparam C=2'd1;
	localparam G=2'd2;
	localparam T=2'd3;

	// Inputs
	reg clk;
	reg clr;
	reg [1:0] din;

	// Outputs
	wire valid;

	// Instantiate the Unit Under Test (UUT)
	pr1 uut ( .clk(clk), .clr(clr), .din(din),  .valid(valid));

	always #5 clk <= ~clk;

	initial begin
		// Initialize Inputs
		clk = 0;
		clr = 1;
		din = T;
		#10;
		clr = 0;
		#10;
		din = C;
		#10;
		din = T;
		#10;
		din = C;
		#10;
		din = T;
		#10;
		din = C;
		#10;
		din = G;
		#10;
		$stop;
	end
endmodule
```




```
`timescale 1ns / 1ps
module pr1(
   input clk,
   input clr,
   input [1:0] din,
   output valid
   );

        localparam A=2'd0;
        localparam C=2'd1;
        localparam G=2'd2;
        localparam T=2'd3;

        reg [2:0] state;

        assign valid = (state==5)||(state==6);

        always@(posedge clk or posedge clr)
                if (clr) state <= 0;
                else case (state)
                        3'd0 : if (din==T) state <= 1;
                                        else if (din==C) state <= 2;
```

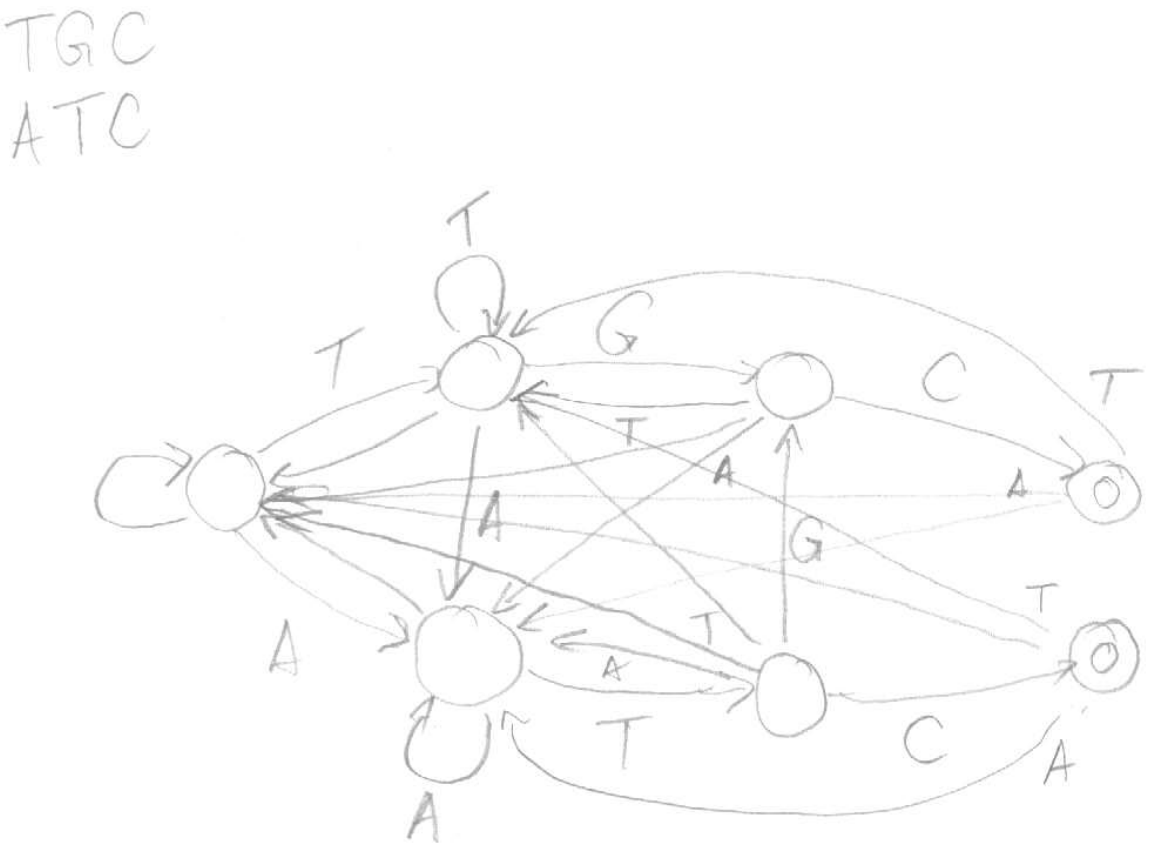
```
                    else state <= 0;
            3'd1 : if (din==C) state <= 3;
                    else if (din==T) state <= 1;
                    else state <= 0;
            3'd2 : if (din==C) state <= 2;
                    else if (din==T) state <= 4;
                    else state <= 0;
            3'd3 : if (din==G) state <= 5;
                    else if (din==T) state <= 4;
                    else if (din==C) state <= 2;
                    else state <= 0;
            3'd4 : if (din==C) state <= 6;
                    else if (din==T) state <= 1;
                    else state <= 0;
            3'd5 : if (din==C) state <= 2;
                    else if (din==T) state <= 1;
                    else state <= 0;
            3'd6 : if (din==G) state <= 5;
                    else if (din==T) state <= 4;
                    else if (din==C) state <= 2;
                    else state <= 0;
            default: state <= 0;
        endcase

endmodule
```



Problema:
Sa se proiecteze un automat cu stari finite Moore ce analizeaza o secventa ADN formata din cele patru nucleotide (A, C, G si T) si identifica aparitia codonilor (secvente de trei nucleotide ce codifica un aminoacid specific): TGC si ATC

Se cere:
a) Diagram de stari
b) Descrierea in verilog a automatului
c) Formele de unda obtinute cu ajutorul testului descris de codul verilog

```
`timescale 1ns / 1ps
module testpr2;

	localparam A=2'd0;
	localparam C=2'd1;
	localparam G=2'd2;
	localparam T=2'd3;

	// Inputs
	reg clk;
	reg clr;
	reg [1:0] din;

	// Outputs
	wire valid;

	// Instantiate the Unit Under Test (UUT)
	pr2 uut (.clk(clk), .clr(clr), .din(din), .valid(valid));

	always #5 clk <= ~clk;

	initial begin
		// Initialize Inputs
		clk = 0;
		clr = 1;
		din = T;
		#10;
		clr = 0;
		#10;
		din = A;
		#10;
		din = T;
		#10;
		din = G;
		#10;
		din = A;
		#10;
		din = T;
		#10;
		din = C;
		#10;
		$stop;
	end
endmodule
```




```
`timescale 1ns / 1ps
module pr2(
    input clk,
    input clr,
    input [1:0] din,
    output valid
    );

        localparam A=2'd0;
        localparam C=2'd1;
        localparam G=2'd2;
        localparam T=2'd3;

        reg [2:0] state;

        assign valid = (state==5)||(state==6);

        always@(posedge clk or posedge clr)
            if (clr) state <= 0;
```

```
else case (state)
        3'd0 : if (din==T) state <= 1;
                        else if (din==A) state <= 2;
                        else state <= 0;
        3'd1 : if (din==G) state <= 3;
                        else if (din==T) state <= 1;
                        else if (din==A) state <= 2;
                        else state <= 0;
        3'd2 : if (din==A) state <= 2;
                        else if (din==T) state <= 4;
                        else state <= 0;
        3'd3 : if (din==C) state <= 5;
                        else if (din==T) state <= 1;
                        else if (din==A) state <= 2;
                        else state <= 0;
        3'd4 : if (din==C) state <= 6;
                        else if (din==T) state <= 1;
                        else if (din==A) state <= 2;
                        else if (din==G) state <= 3;
                        else state <= 0;
        3'd5 : if (din==A) state <= 2;
                        else if (din==T) state <= 1;
                        else state <= 0;
        3'd6 : if (din==T) state <=1;
                        else if (din==A) state <= 2;
                        else state <= 0;
        default: state <= 0;
endcase

endmodule
```