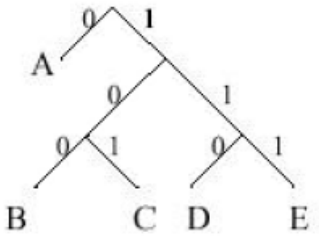
Problema:

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: