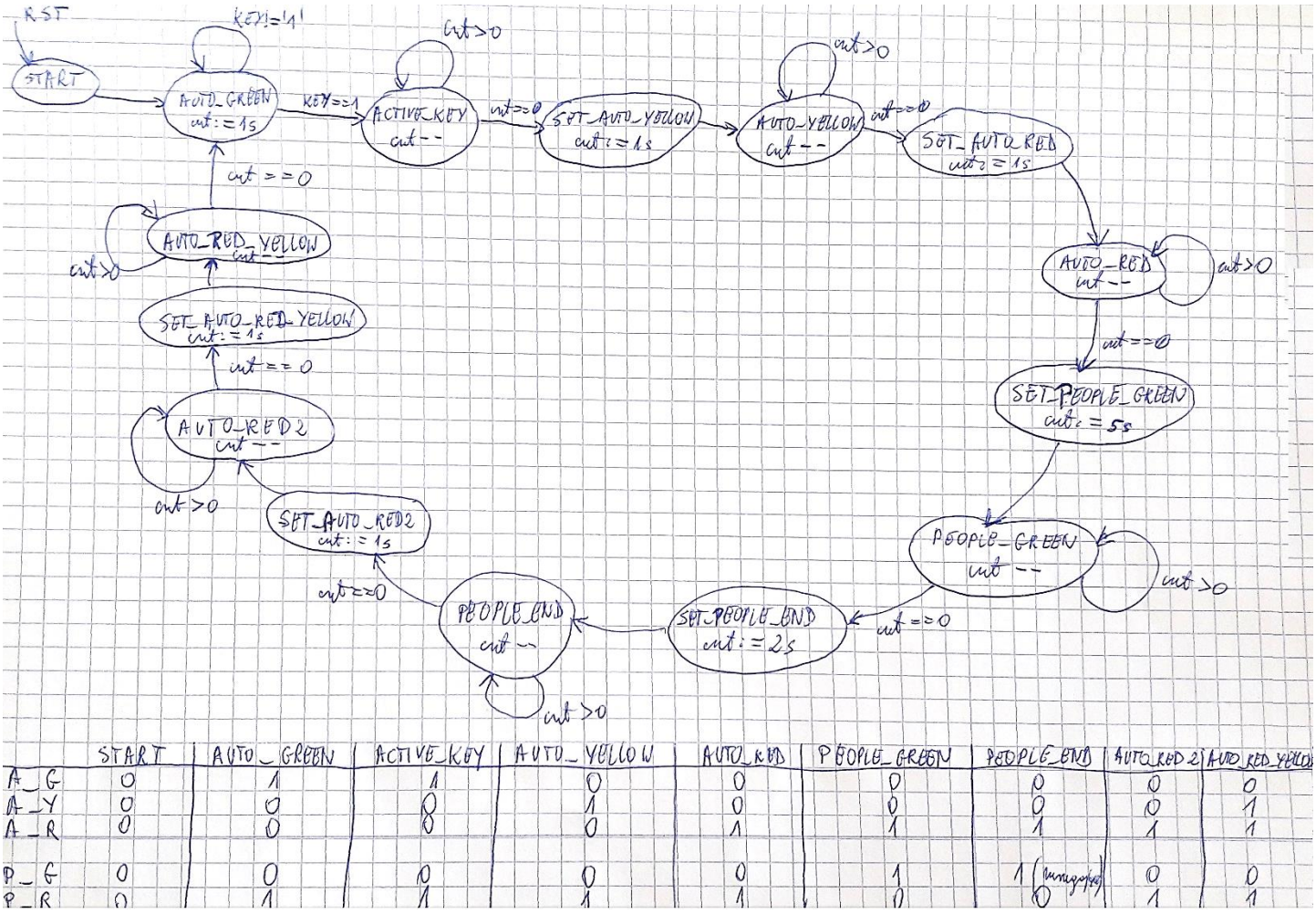
POLITECHNIKA POZNAŃSKA	Programowalne układy cyfrowe	Rok akademicki 2017/2018
Wydział Elektroniki i Telekomunikacji	Temat: Implementacja sygnalizacji świetlnej na przejściu dla pieszych w języku Verilog (światło zielone dla pieszych wyzwalanie z przycisku).	
Wykonali: **Marcin Brach** **Marcin Jeżewski** **Maciej Bęcki**	Rok studiów I Semestr II Stopień II Studia niestacjonarne	

1. Cel projektu

Celem projektu jest zaprojektowanie oraz implementacja automatu stanów w technologii FPGA z wykorzystaniem oprogramowania Lattice Diamond 3.8 oraz Active-HDL 10.3. Automat ten będzie opisywał sygnalizację świetlną na przejściu dla pieszych w przypadku gdy światło zielone dla pieszych jest wyzwalane po naciśnięciu przycisku.

2. Sposób wykonania

Projektowany automat należy opisać za pomocą grafu oraz tabeli prawdy określającej stany poszczególnych świateł na przejściu. Następnie należy dokonać implementacji w języku Verilog, napisać testbench i dokonać symulacji oraz uruchomić kod na płytce prototypowej z układem MachXO2-4000HC.



	START	AUTO_GREEN	ACTIVE_KEY	AUTO_YELLOW	AUTO_RED	PEOPLE_GREEN	PEOPLE_END	AUTO_RED2	AUTO_RED_YELLOW
A_G	0	1	1	0	0	0	0	0	0
A_Y	0	0	0	1	0	0	0	0	1
A_R	0	0	0	0	1	1	1	1	1
P_G	0	0	0	0	0	1	1 (migający)	0	0
P_R	0	1	1	1	1	0	0	1	1

Rysunek 1 Schemat projektowanego automatu.

3. Implementacja automatu w języku Verilog

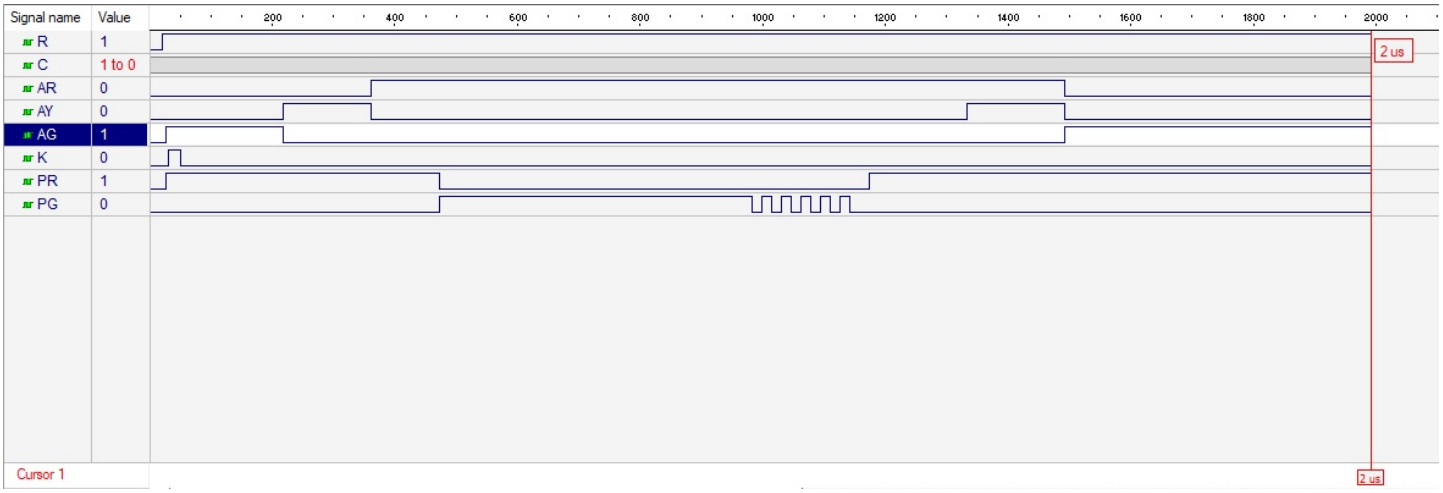
```
module lights(
input wire RST,
input wire CLOCK,
input wire KEY,
output reg P_R,
output reg P_G,
output reg A_R,
output reg A_Y,
output reg A_G
);

//wartości użyte do symulacji:
//parameter CNT_AG = 10, CNT_AY = 8, CNT_AR = 6, CNT_PG = 30, CNT_PE = 12, CNT_ARII = 9, CNT_ARY = 10;

//wartości użyte do uruchomienia na płytce uruchomieniowej (25 = 1 sekunda):
parameter CNT_AG = 25, CNT_AY = 25, CNT_AR = 25, CNT_PG = 125, CNT_PE = 50, CNT_ARII = 25, CNT_ARY = 25;

localparam START = 0, AUTO_GREEN = 1, ACTIVE_KEY = 2, SET_AUTO_YELLOW = 3, AUTO_YELLOW = 4,SET_AUTO_RED = 5,
AUTO_RED = 6, SET_PEOPLE_GREEN = 7,PEOPLE_GREEN = 8,SET_PEOPLE_END = 9, PEOPLE_END = 10,SET_AUTO_REDII = 11,
AUTO_REDII = 12, SET_AUTO_RED_YELLOW = 13, AUTO_RED_YELLOW = 14;

reg[3:0] state, next_state;
reg[31:0] CNT;
reg [31:0] period;

wire CLK;
wire nRST = !RST;


always @(posedge CLOCK or posedge nRST)
if(nRST) period = 0;
else period = period + 1;

assign CLK = period[21];  //do symulacji użyto: assign CLK = period[1];


always@ (posedge CLK or posedge nRST)
if (nRST) state <= START;
else state <= next_state;


always@ (*)
begin
case (state)
START:                      next_state <= AUTO_GREEN;
AUTO_GREEN:                 if(KEY) next_state <= ACTIVE_KEY;
                            else next_state <= AUTO_GREEN;
ACTIVE_KEY:                 if(CNT == 0) next_state <= SET_AUTO_YELLOW;
                            else next_state <= ACTIVE_KEY;
SET_AUTO_YELLOW:            next_state <=AUTO_YELLOW;
AUTO_YELLOW:                if(CNT == 0) next_state <= SET_AUTO_RED;
                            else next_state <= AUTO_YELLOW;
SET_AUTO_RED:               next_state <= AUTO_RED;
AUTO_RED:                   if(CNT == 0) next_state <= SET_PEOPLE_GREEN;
                            else next_state <= AUTO_RED;
SET_PEOPLE_GREEN:           next_state <= PEOPLE_GREEN;
PEOPLE_GREEN:               if(CNT == 0) next_state <= SET_PEOPLE_END;
                            else next_state <= PEOPLE_GREEN;
SET_PEOPLE_END:             next_state <= PEOPLE_END;
PEOPLE_END:                 if(CNT == 0) next_state <= SET_AUTO_REDII;
                            else next_state <= PEOPLE_END;
SET_AUTO_REDII:             next_state <= AUTO_REDII;
AUTO_REDII:                 if(CNT == 0) next_state <= SET_AUTO_RED_YELLOW;
                            else next_state <= AUTO_REDII;
SET_AUTO_RED_YELLOW:        next_state <= AUTO_RED_YELLOW;
AUTO_RED_YELLOW:            if(CNT == 0) next_state <= AUTO_GREEN;
                            else next_state <= AUTO_RED_YELLOW;
default :                   next_state<= START;
endcase
end
```

```
always@ (posedge CLK or posedge nRST)
if(nRST) CNT <= 'd0;
else if(next_state == AUTO_GREEN) CNT <= CNT_AG;
else if(next_state == ACTIVE_KEY) CNT <= CNT - 'd1;
else if(next_state == SET_AUTO_YELLOW) CNT <= CNT_AY;
else if(next_state == AUTO_YELLOW) CNT <= CNT - 'd1;
else if(next_state == SET_AUTO_RED) CNT <= CNT_AR;
else if(next_state == AUTO_RED) CNT <= CNT - 'd1;
else if(next_state == SET_PEOPLE_GREEN) CNT <= CNT_PG;
else if(next_state == PEOPLE_GREEN) CNT <= CNT - 'd1;
else if(next_state == SET_PEOPLE_END) CNT <= CNT_PE;
else if(next_state == PEOPLE_END) CNT <= CNT - 'd1;
else if(next_state == SET_AUTO_REDII) CNT <= CNT_ARII;
else if(next_state == AUTO_REDII) CNT <= CNT - 'd1;
else if(next_state == SET_AUTO_RED_YELLOW) CNT <= CNT_ARY;
else if(next_state == AUTO_RED_YELLOW) CNT <= CNT - 'd1;
else CNT <= 'd0;


always@ (posedge CLK or posedge nRST)
if(nRST) begin P_R <= 1'b0; P_G <= 1'b0; A_R <= 1'b0; A_Y <= 1'b0; A_G <= 1'b0; end
else if(next_state == START) begin P_R <= 1'b0; P_G <= 1'b0; A_R <= 1'b0; A_Y <= 1'b0; A_G <= 1'b0; end
else if(next_state == AUTO_GREEN) begin P_R <=1'b1; P_G <= 1'b0; A_R <= 1'b0; A_Y <= 1'b0; A_G <= 1'b1; end
else if(next_state == ACTIVE_KEY) begin P_R <= 1'b1; P_G <= 1'b0; A_R <= 1'b0; A_Y <= 1'b0; A_G <= 1'b1; end
else if(next_state == AUTO_YELLOW) begin P_R <= 1'b1; P_G <= 1'b0; A_R <= 1'b0; A_Y <= 1'b1; A_G <= 1'b0; end
else if(next_state == AUTO_RED) begin P_R <= 1'b1; P_G <= 1'b0; A_R <= 1'b1; A_Y <= 1'b0; A_G <= 1'b0; end
else if(next_state == PEOPLE_GREEN) begin P_R <= 1'b0; P_G <= 1'b1; A_R <= 1'b1; A_Y <= 1'b0; A_G <= 1'b0; end
else if(next_state == PEOPLE_END) begin P_R <= 1'b0; P_G <= period[23]; A_R <= 1'b1; A_Y <= 1'b0; A_G <= 1'b0;
end  //do symulacji użyto: P_G <= period[2];
else if(next_state == AUTO_REDII) begin P_R <= 1'b1; P_G <= 1'b0; A_R <= 1'b1; A_Y <= 1'b0; A_G <= 1'b0; end
else if(next_state == AUTO_RED_YELLOW) begin P_R <= 1'b1; P_G <= 1'b0; A_R <= 1'b1; A_Y <= 1'b1; A_G <= 1'b0; end

endmodule
```

4. Testbench

```
`default_nettype none
`timescale 1 ns / 1 ns

module testbench();

reg R, C, AR, AY, AG, K, PR, PG;

lights ligInst ( .RST(R), .CLOCK(C), .KEY(K), .P_R(PR), .P_G(PG), .A_R(AR), .A_Y(AY), .A_G(AG));


initial begin
           R=0;
       #20 R=1;
end

initial begin
           K=0;
       #30 K=1;
       #20 K=0;
end

initial begin
          C=0;
end

always #2 C = ~C;

endmodule
```

5. Wyniki symulacji



Rysunek 2 Przebiegi czasowe otrzymane w wyniku symulacji