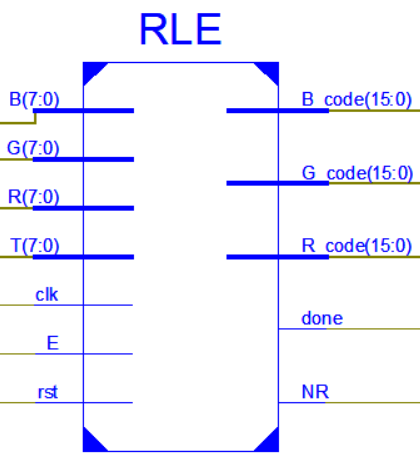
**به نام خدا**

**پیاده‌سازی الگوریتم فشرده‌سازی RLE بر بستر FPGA**

**فایل‌های ورودی و خروجی برنامه به همراه تصاویر اولیه و نتایج در پوشه test قراردارند. ابعاد تصاویر ورودی باید 128 در 128 پیکسل باشد.**

|  |  |
| --- | --- |
| **تصویر ورودی** | **تصویر خروجی T = 30** |
|  |  |
|  |  |
|  |  |

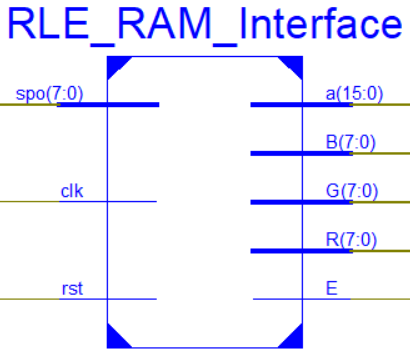
**نمای شماتیک RLE**

**

**کد RLE**

|  |
| --- |
| *module RLE(rst, clk, NR, E, T, R, G, B, R\_code, G\_code, B\_code, done);*  *input rst, clk, E;*  *input [7:0] T;*  *input [7:0] R, G, B;*  *output reg [15:0] R\_code, G\_code, B\_code;*  *output reg done, NR;*    *reg[7:0] r\_cnt, g\_cnt, b\_cnt;*  *reg[8:0] r\_cur, g\_cur, b\_cur;*  *reg [8:0] i, j;*  *parameter [8:0] MAX\_CUR = 511;*  *parameter [6:0] MAX\_I = 127;*    *always @ (posedge clk or posedge rst) begin*  *if (rst) begin*  *R\_code = 0;*  *G\_code = 0;*  *B\_code = 0;*  *done = 0;*  *r\_cnt = 0;*  *g\_cnt = 0;*  *b\_cnt = 0;*  *r\_cur = MAX\_CUR;*  *g\_cur = MAX\_CUR;*  *b\_cur = MAX\_CUR;*  *i = 0;*  *j = 0;*  *NR = 0;*  *end*  *else begin*  *NR = 0;*  *if (E) begin*  *if (r\_cur == MAX\_CUR) begin*  *R\_code = 0;*  *r\_cnt = 1;*  *r\_cur = R;*  *end*  *else if (i == MAX\_I && ((R-r\_cur >= 0 && R-r\_cur <= T) || (r\_cur-R >= 0 && r\_cur-R <= T))) begin*  *r\_cnt = r\_cnt + 1;*  *R\_code = {r\_cnt, r\_cur[7:0]};*  *r\_cnt = 0;*  *r\_cur = MAX\_CUR;*  *end*  *else if ((R-r\_cur >= 0 && R-r\_cur <= T) || (r\_cur-R >= 0 && r\_cur-R <= T)) begin*  *r\_cnt = r\_cnt + 1;*  *R\_code = 0;*  *end*  *else begin*  *R\_code = {r\_cnt, r\_cur[7:0]};*  *r\_cnt = 1;*  *r\_cur = R;*  *end*    *if (g\_cur == MAX\_CUR) begin*  *G\_code = 0;*  *g\_cnt = 1;*  *g\_cur = G;*  *end*  *else if (i == MAX\_I && ((G-g\_cur >= 0 && G-g\_cur <= T) || (g\_cur-G >= 0 && g\_cur-G <= T))) begin*  *g\_cnt = g\_cnt + 1;*  *G\_code = {g\_cnt, g\_cur[7:0]};*  *g\_cnt = 0;*  *g\_cur = MAX\_CUR;*  *end*  *else if ((G-g\_cur >= 0 && G-g\_cur <= T) || (g\_cur-G >= 0 && g\_cur-G <= T)) begin*  *g\_cnt = g\_cnt + 1;*  *G\_code = 0;*  *end*  *else begin*  *G\_code = {g\_cnt, g\_cur[7:0]};*  *g\_cnt = 1;*  *g\_cur = G;*  *end*  *if (b\_cur == MAX\_CUR) begin*  *B\_code = 0;*  *b\_cnt = 1;*  *b\_cur = B;*  *end*  *else if (i == MAX\_I && ((B-b\_cur >= 0 && B-b\_cur <= T) || (b\_cur-B >= 0 && b\_cur-B <= T))) begin*  *b\_cnt = b\_cnt + 1;*  *B\_code = {b\_cnt, b\_cur[7:0]};*  *b\_cnt = 0;*  *b\_cur = MAX\_CUR;*  *end*  *else if ((B-b\_cur >= 0 && B-b\_cur <= T) || (b\_cur-B >= 0 && b\_cur-B <= T)) begin*  *b\_cnt = b\_cnt + 1;*  *B\_code = 0;*  *end*  *else begin*  *B\_code = {b\_cnt, b\_cur[7:0]};*  *b\_cnt = 1;*  *b\_cur = B;*  *end*    *i = i + 1;*  *if (i > MAX\_I)*  *j = j + 1;*  *end*  *else begin*  *if (j > MAX\_I) begin*  *j = j + 1;*  *if (j >= MAX\_I + 8)*  *done = 1;*  *end*  *if (i == MAX\_I + 1) begin*  *if (r\_cur == MAX\_CUR)*  *R\_code = 0;*  *else*  *R\_code = {r\_cnt, r\_cur[7:0]};*  *r\_cnt = 0;*  *r\_cur = MAX\_CUR;*    *if (g\_cur == MAX\_CUR)*  *G\_code = 0;*  *else*  *G\_code = {g\_cnt, g\_cur[7:0]};*  *g\_cnt = 0;*  *g\_cur = MAX\_CUR;*  *if (b\_cur == MAX\_CUR)*  *B\_code = 0;*  *else*  *B\_code = {b\_cnt, b\_cur[7:0]};*  *b\_cnt = 0;*  *b\_cur = MAX\_CUR;*    *NR = 1;*  *i = 0;*  *end*  *else begin*  *R\_code = 0;*  *G\_code = 0;*  *B\_code = 0;*  *end*  *end*  *end*  *end*  *endmodule* |

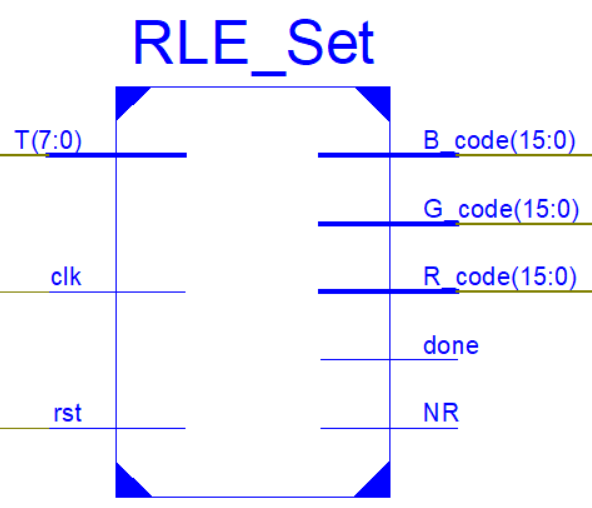
**نمای شماتیک RLE RAM Interface**

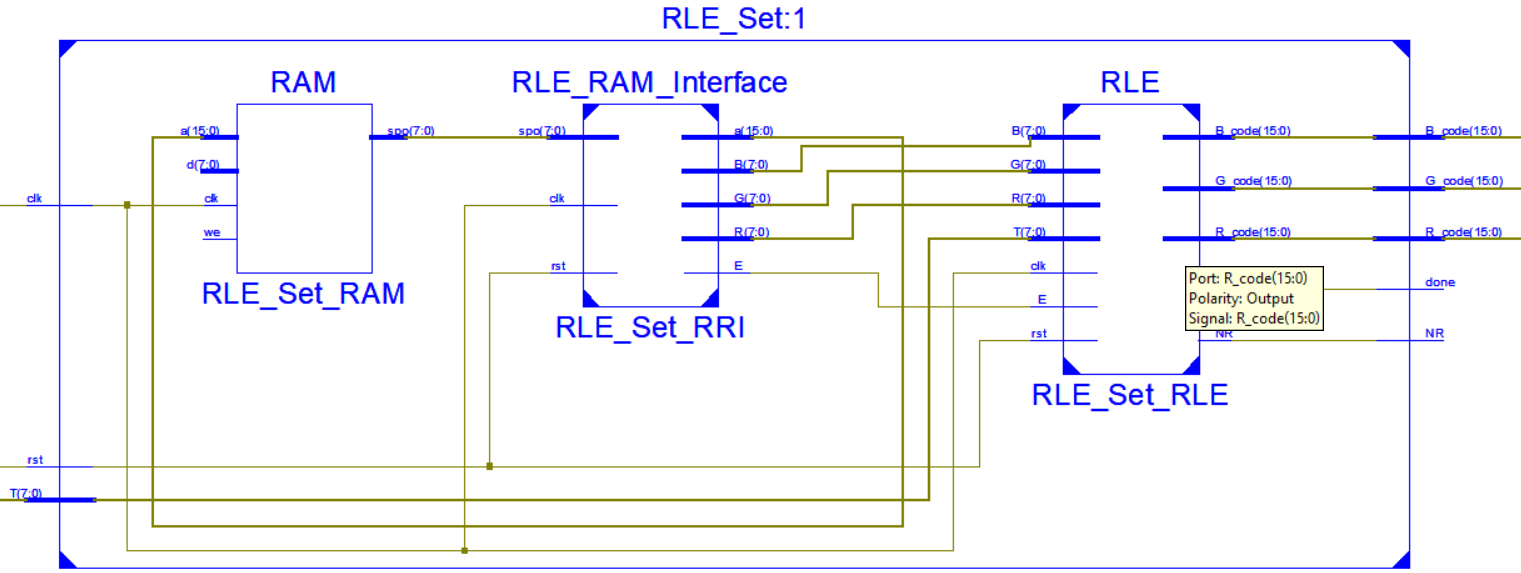
****

**کد RLE RAM Interface**

|  |
| --- |
| *module RLE\_RAM\_Interface(clk, rst, a, spo, E, R, G, B);*  *input clk, rst;*  *input [7:0] spo;*  *output reg E;*  *output reg [15:0] a;*  *output reg [7:0] R, G, B;*    *parameter HOLD\_CLK = 1;*  *reg [15:0] hold\_cnt;*    *parameter RBA = 0, GBA = 16384, BBA = 32768;*  *reg [14:0] offset;*    *reg [2:0] state;*  *parameter [2:0] r1 = 0, r2 = 1, g1 = 2, g2 = 3, b1 = 4, b2 = 5, hold = 6;*    *always @ (posedge clk or posedge rst) begin*  *if (rst) begin*  *E = 0;*  *R = 0;*  *G = 0;*  *B = 0;*  *offset = 0;*  *state = r1;*  *hold\_cnt = 0;*  *end*  *else begin*  *E = 0;*  *if (offset < GBA) begin*  *if (state == r1) begin*  *a = offset + RBA;*  *state = r2;*  *end*  *else if (state == r2) begin*  *R = spo;*  *state = g1;*  *end*  *else if (state == g1) begin*  *a = offset + GBA;*  *state = g2;*  *end*  *else if (state == g2) begin*  *G = spo;*  *state = b1;*  *end*    *else if (state == b1) begin*  *a = offset + BBA;*  *state = b2;*  *end*  *else if (state == b2) begin*  *B = spo;*  *E = 1;*  *offset = offset + 1;*  *state = hold;*  *end*  *else if (state == hold) begin*  *hold\_cnt = hold\_cnt + 1;*  *if (hold\_cnt >= HOLD\_CLK) begin*  *hold\_cnt = 0;*  *state = r1;*  *end*  *end*  *end*  *end*  *end*  *endmodule* |

**نمای شماتیک RLE Set**



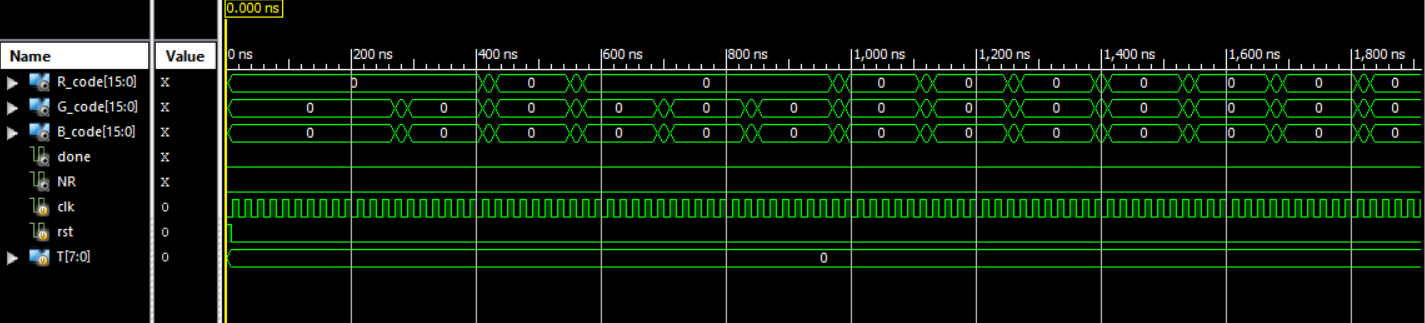


**کد RLE Set**

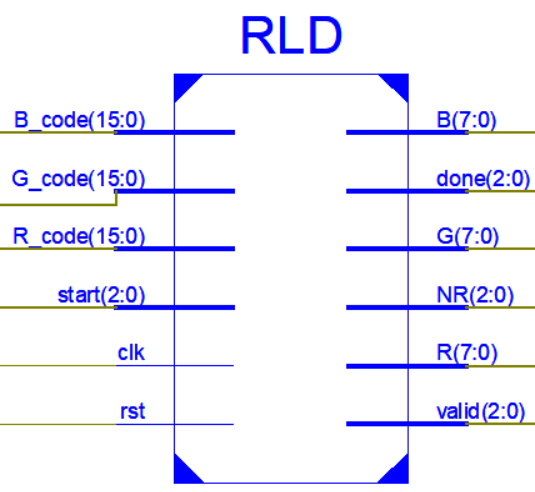
|  |
| --- |
| *module RLE\_Set(clk, rst, T, done, NR, R\_code, G\_code, B\_code);*  *input clk, rst;*  *input [7:0] T;*  *output done, NR;*  *output [15:0] R\_code, G\_code, B\_code;*    *parameter HOLD\_CLK = 1;*    *wire [15:0] a;*  *wire [7:0] spo;*  *wire E;*  *wire [7:0] R, G, B;*    *RAM RLE\_Set\_RAM (*  *.a(a),*  *.clk(clk),*  *.spo(spo)*  *);*    *RLE\_RAM\_Interface RLE\_Set\_RRI (*  *.rst(rst),*  *.clk(clk),*  *.a(a),*  *.spo(spo),*  *.E(E),*  *.R(R),*  *.G(G),*  *.B(B)*  *);*  *defparam RLE\_Set\_RRI.HOLD\_CLK = HOLD\_CLK;*    *RLE RLE\_Set\_RLE (*  *.rst(rst),*  *.clk(clk),*  *.E(E),*  *.T(T),*  *.R(R),*  *.G(G),*  *.B(B),*  *.R\_code(R\_code),*  *.G\_code(G\_code),*  *.B\_code(B\_code),*  *.done(done),*  *.NR(NR)*  *);*  *endmodule* |

**تست RLE Set**

|  |
| --- |
| *module RLE\_Set\_tb;*  *reg clk, rst;*  *reg [7:0] T;*  *wire [15:0] R\_code, G\_code, B\_code;*  *wire done, NR;*  *integer rf, gf, bf;*    *RLE\_Set uut (*  *.clk(clk),*  *.rst(rst),*  *.T(T),*  *.done(done),*  *.NR(NR),*  *.R\_code(R\_code),*  *.G\_code(G\_code),*  *.B\_code(B\_code)*  *);*  *initial begin*  *clk = 0;*  *rst = 0;*  *T = 0;*  *#2 rst = 1; #7 rst = 0;*  *end*    *initial forever #10 clk = ~clk;*    *initial begin*  *#20;*    *rf = $fopen("test/encoded/R.txt", "w");*  *gf = $fopen("test/encoded/G.txt", "w");*  *bf = $fopen("test/encoded/B.txt", "w");*    *while (~done) begin*  *#20;*    *if (R\_code > 0)*  *$fwrite(rf, "%0d,", R\_code);*  *//$fwrite(rf, "%0d,%0d ", R\_code[15:8], R\_code[7:0]);*  *if (NR)*  *$fwrite(rf, "\n");*    *if (G\_code > 0)*  *$fwrite(gf, "%0d,", G\_code);*  *//$fwrite(gf, "%0d,%0d ", G\_code[15:8], G\_code[7:0]);*  *if (NR)*  *$fwrite(gf, "\n");*  *if (B\_code > 0)*  *$fwrite(bf, "%0d,", B\_code);*  *//$fwrite(bf, "%0d,%0d ", B\_code[15:8], B\_code[7:0]);*  *if (NR)*  *$fwrite(bf, "\n");*  *end*  *$fclose(rf);*  *$fclose(gf);*  *$fclose(bf);*  *rf = 0;*  *gf = 0;*  *bf = 0;*  *end*  *endmodule* |



**نمای شماتیک RLD**

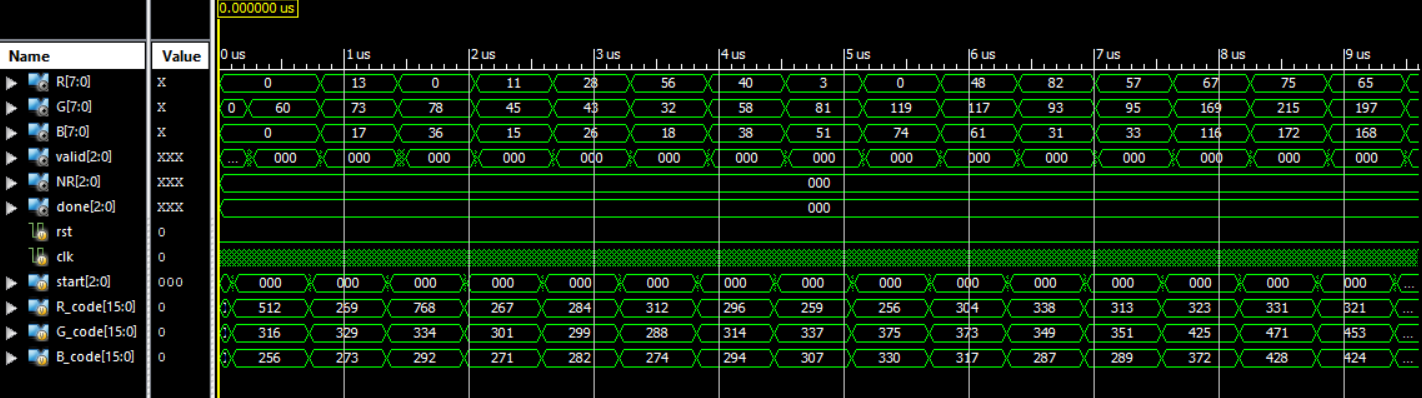
****

**کد RLD**

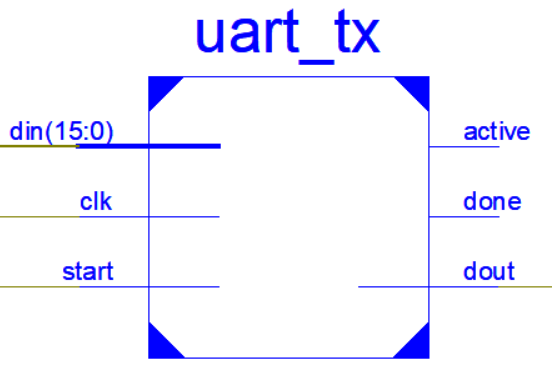
|  |
| --- |
| *module RLD(rst, clk, start, R\_code, G\_code, B\_code, R, G, B, done, valid, NR);*  *input rst, clk;*  *input [2:0] start;*  *input [15:0] R\_code, G\_code, B\_code;*  *output reg [7:0] R, G, B;*  *output reg [2:0] valid, NR, done;*    *parameter [6:0] MAX\_I = 127;*    *reg [15:0] r\_c, r\_din;*  *reg [7:0] r\_cnt, r\_cur;*  *reg r\_wr, r\_rd;*  *reg [7:0] r\_i, r\_j;*  *reg [1:0] r\_k;*  *wire r\_full, r\_empty, r\_valid;*  *wire [15:0] r\_dout;*    *reg [15:0] g\_c, g\_din;*  *reg [7:0] g\_cnt, g\_cur;*  *reg g\_wr, g\_rd;*  *reg [7:0] g\_i, g\_j;*  *reg [1:0] g\_k;*  *wire g\_full, g\_empty, g\_valid;*  *wire [15:0] g\_dout;*  *reg [15:0] b\_c, b\_din;*  *reg [7:0] b\_cnt, b\_cur;*  *reg b\_wr, b\_rd;*  *reg [7:0] b\_i, b\_j;*  *reg [1:0] b\_k;*  *wire b\_full, b\_empty, b\_valid;*  *wire [15:0] b\_dout;*  *Buff R\_BUF (*  *.clk(clk),*  *.rst(rst),*  *.din(r\_din),*  *.wr\_en(r\_wr),*  *.rd\_en(r\_rd),*  *.dout(r\_dout),*  *.full(r\_full),*  *.empty(r\_empty),*  *.valid(r\_valid)*  *);*    *Buff G\_BUF (*  *.clk(clk),*  *.rst(rst),*  *.din(g\_din),*  *.wr\_en(g\_wr),*  *.rd\_en(g\_rd),*  *.dout(g\_dout),*  *.full(g\_full),*  *.empty(g\_empty),*  *.valid(g\_valid)*  *);*    *Buff B\_BUF (*  *.clk(clk),*  *.rst(rst),*  *.din(b\_din),*  *.wr\_en(b\_wr),*  *.rd\_en(b\_rd),*  *.dout(b\_dout),*  *.full(b\_full),*  *.empty(b\_empty),*  *.valid(b\_valid)*  *);*    *always @ (posedge clk or posedge rst) begin*  *if (rst) begin*  *r\_din <= 0;*  *r\_wr <= 0;*    *g\_din <= 0;*  *g\_wr <= 0;*  *b\_din <= 0;*  *b\_wr <= 0;*  *end*  *else begin*  *if (R\_code > 0 && ~r\_full && start[0]) begin*  *r\_din <= R\_code;*  *r\_wr <= 1;*  *end*  *else begin*  *r\_wr <= 0;*  *end*  *if (G\_code > 0 && ~g\_full && start[1]) begin*  *g\_din <= G\_code;*  *g\_wr <= 1;*  *end*  *else begin*  *g\_wr <= 0;*  *end*  *if (B\_code > 0 && ~b\_full && start[2]) begin*  *b\_din <= B\_code;*  *b\_wr <= 1;*  *end*  *else begin*  *b\_wr <= 0;*  *end*  *end*  *end*    *always @ (posedge clk or posedge rst) begin*  *if (rst) begin*  *done = 0;*  *valid = 0;*  *NR = 0;*  *R = 0;*  *G = 0;*  *B = 0;*    *r\_rd = 0;*  *r\_c = 0;*  *r\_cur = 0;*  *r\_cnt = 0;*  *r\_i = 0;*  *r\_j = 0;*  *r\_k = 0;*  *g\_rd = 0;*  *g\_c = 0;*  *g\_cur = 0;*  *g\_cnt = 0;*  *g\_i = 0;*  *g\_j = 0;*  *g\_k = 0;*  *b\_rd = 0;*  *b\_c = 0;*  *b\_cur = 0;*  *b\_cnt = 0;*  *b\_i = 0;*  *b\_j = 0;*  *b\_k = 0;*  *end*  *else begin*    *// R Section*  *if (r\_cnt == 0 && r\_c == r\_dout && ~r\_empty && ~r\_wr) begin*  *r\_rd = 1;*  *if (r\_i == 0)*  *r\_c = 0;*  *end*  *else begin*  *r\_rd = 0;*  *end*    *if (r\_i == 0)*  *if (r\_k == 3)*  *r\_k = 0;*  *else*  *r\_k = r\_k + 1;*    *if (r\_cnt == 0 && r\_dout > 0 && r\_c != r\_dout && r\_valid && r\_k == 0) begin*  *r\_c = r\_dout;*  *r\_cnt = r\_dout[15:8];*  *r\_cur = r\_dout[7:0];*  *end*    *if (r\_cnt > 0) begin*  *R = r\_cur;*  *valid[0] = 1;*  *r\_cnt = r\_cnt - 1;*  *r\_i = r\_i + 1;*  *end*  *else begin*  *valid[0] = 0;*  *end*  *if (r\_i == MAX\_I + 1 && r\_cnt == 0) begin*  *r\_j = r\_j + 1;*  *NR[0] = 1;*  *r\_i = 0;*  *end*  *else begin*  *NR[0] = 0;*  *end*    *if (r\_j == MAX\_I + 1)*  *done[0] = 1;*    *// G Section*  *if (g\_cnt == 0 && g\_c == g\_dout && ~g\_empty && ~g\_wr) begin*  *g\_rd = 1;*  *if (g\_i == 0)*  *g\_c = 0;*  *end*  *else begin*  *g\_rd = 0;*  *end*    *if (g\_i == 0)*  *if (g\_k == 3)*  *g\_k = 0;*  *else*  *g\_k = g\_k + 1;*    *if (g\_cnt == 0 && g\_dout > 0 && g\_c != g\_dout && g\_valid && g\_k == 0) begin*  *g\_c = g\_dout;*  *g\_cnt = g\_dout[15:8];*  *g\_cur = g\_dout[7:0];*  *end*    *if (g\_cnt > 0) begin*  *G = g\_cur;*  *valid[1] = 1;*  *g\_cnt = g\_cnt - 1;*  *g\_i = g\_i + 1;*  *end*  *else begin*  *valid[1] = 0;*  *end*  *if (g\_i == MAX\_I + 1 && g\_cnt == 0) begin*  *g\_j = g\_j + 1;*  *NR[1] = 1;*  *g\_i = 0;*  *end*  *else begin*  *NR[1] = 0;*  *end*    *if (g\_j == MAX\_I + 1)*  *done[1] = 1;*  *// B Section*  *if (b\_cnt == 0 && b\_c == b\_dout && ~b\_empty && ~b\_wr) begin*  *b\_rd = 1;*  *if (b\_i == 0)*  *b\_c = 0;*  *end*  *else begin*  *b\_rd = 0;*  *end*    *if (b\_i == 0)*  *if (b\_k == 3)*  *b\_k = 0;*  *else*  *b\_k = b\_k + 1;*    *if (b\_cnt == 0 && b\_dout > 0 && b\_c != b\_dout && b\_valid && b\_k == 0) begin*  *b\_c = b\_dout;*  *b\_cnt = b\_dout[15:8];*  *b\_cur = b\_dout[7:0];*  *end*    *if (b\_cnt > 0) begin*  *B = b\_cur;*  *valid[2] = 1;*  *b\_cnt = b\_cnt - 1;*  *b\_i = b\_i + 1;*  *end*  *else begin*  *valid[2] = 0;*  *end*  *if (b\_i == MAX\_I + 1 && b\_cnt == 0) begin*  *b\_j = b\_j + 1;*  *NR[2] = 1;*  *b\_i = 0;*  *end*  *else begin*  *NR[2] = 0;*  *end*    *if (b\_j == MAX\_I + 1)*  *done[2] = 1;*    *end*  *end*  *endmodule* |

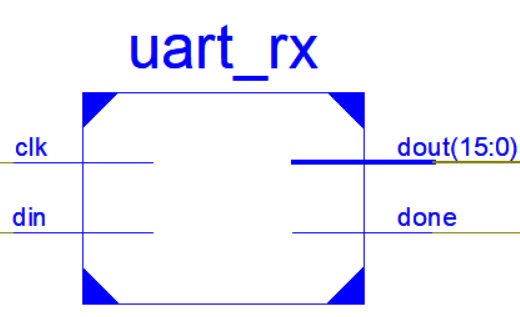
**تست RLD**

|  |
| --- |
| *module RLD\_tb;*  *reg rst, clk;*  *reg [2:0] start;*  *reg [15:0] R\_code, G\_code, B\_code;*  *wire [7:0] R, G, B;*  *wire [2:0] valid, NR, done;*    *integer rrf, rgf, rbf;*  *integer wrf, wgf, wbf;*  *RLD uut (*  *.rst(rst),*  *.clk(clk),*  *.start(start),*  *.R\_code(R\_code),*  *.G\_code(G\_code),*  *.B\_code(B\_code),*  *.R(R),*  *.G(G),*  *.B(B),*  *.done(done),*  *.valid(valid),*  *.NR(NR)*  *);*  *initial begin*  *rst = 0;*  *clk = 0;*  *start = 0;*  *R\_code = 0;*  *G\_code = 0;*  *B\_code = 0;*  *#2 rst = 1; #7 rst = 0;*  *end*    *initial forever #10 clk = ~clk;*    *initial begin*  *#100;*    *rrf = $fopen("test/encoded/R.txt", "r");*  *$fscanf(rrf, "%d,", R\_code);*  *start[0] = 1;*  *#20;*  *start[0] = 0;*    *while (~R\_code && ~done[0]) begin*  *#600;*  *$fscanf(rrf, "%d,", R\_code);*  *start[0] = 1;*  *#20;*  *start[0] = 0;*  *end*    *$fclose(rrf);*  *rrf = 0;*  *end*    *initial begin*  *#100;*    *rgf = $fopen("test/encoded/G.txt", "r");*  *$fscanf(rgf, "%d,", G\_code);*  *start[1] = 1;*  *#20;*  *start[1] = 0;*    *while (~G\_code && ~done[1]) begin*  *#600;*  *$fscanf(rgf, "%d,", G\_code);*  *start[1] = 1;*  *#20;*  *start[1] = 0;*  *end*    *$fclose(rgf);*  *rgf = 0;*  *end*  *initial begin*  *#100;*    *rbf = $fopen("test/encoded/B.txt", "r");*  *$fscanf(rbf, "%d,", B\_code);*  *start[2] = 1;*  *#20;*  *start[2] = 0;*    *while (~B\_code && ~done[2]) begin*  *#600;*  *$fscanf(rbf, "%d,", B\_code);*  *start[2] = 1;*  *#20;*  *start[2] = 0;*  *end*    *$fclose(rbf);*  *rbf = 0;*  *end*  *initial begin*  *#100;*    *wrf = $fopen("test/decoded/R.txt", "w");*    *while (~done[0]) begin*  *#20;*  *if (valid[0])*  *$fwrite(wrf, "%0d,", R);*  *if (NR[0])*  *$fwrite(wrf, "\n");*  *end*    *$fclose(wrf);*  *wrf = 0;*  *end*  *initial begin*  *#100;*    *wgf = $fopen("test/decoded/G.txt", "w");*    *while (~done[1]) begin*  *#20;*  *if (valid[1])*  *$fwrite(wgf, "%0d,", G);*  *if (NR[1])*  *$fwrite(wgf, "\n");*  *end*    *$fclose(wgf);*  *wgf = 0;*  *end*  *initial begin*  *#100;*    *wbf = $fopen("test/decoded/B.txt", "w");*    *while (~done[2]) begin*  *#20;*  *if (valid[2])*  *$fwrite(wbf, "%0d,", B);*  *if (NR[2])*  *$fwrite(wbf, "\n");*  *end*    *$fclose(wbf);*  *wbf = 0;*  *end*  *endmodule* |



**نمای شماتیک Uart**

****

****

**کد Uart Transmitter**

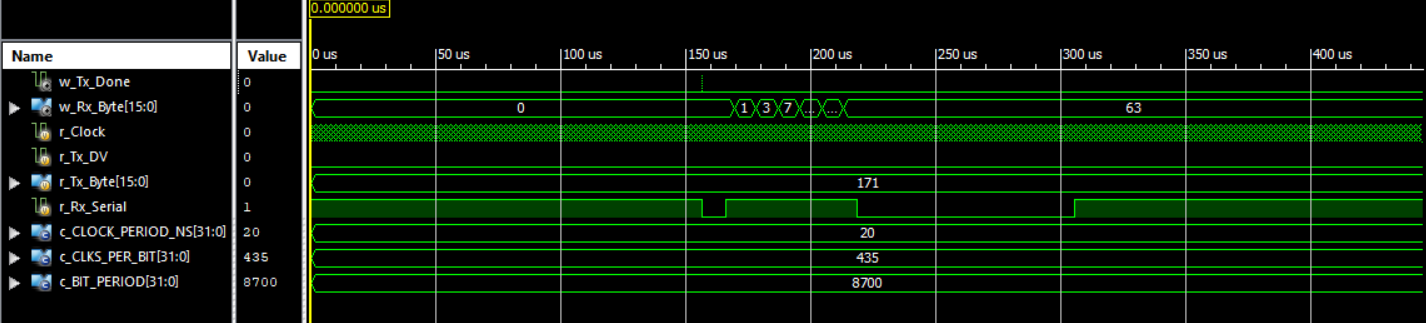
|  |
| --- |
| *module uart\_tx*  *(*  *input        clk,*  *input        start,*  *input [15:0] din,*  *output       active,*  *output reg   dout,*  *output       done*  *);*  *// 50000000 / 115200 = 435*  *parameter CLKS\_PER\_BIT = 435;*    *parameter s\_IDLE         = 3'b000;*  *parameter s\_TX\_START\_BIT = 3'b001;*  *parameter s\_TX\_DATA\_BITS = 3'b010;*  *parameter s\_TX\_STOP\_BIT  = 3'b011;*  *parameter s\_CLEANUP      = 3'b100;*    *reg [2:0]     r\_SM\_Main     = 0;*  *reg [15:0]    r\_Clock\_Count = 0;*  *reg [3:0]     r\_Bit\_Index   = 0;*  *reg [15:0]    r\_Tx\_Data     = 0;*  *reg           r\_Tx\_Done     = 0;*  *reg           r\_Tx\_Active   = 0;*    *always @(posedge clk)*  *begin*    *case (r\_SM\_Main)*  *s\_IDLE :*  *begin*  *dout   <= 1'b1;         // Drive Line High for Idle*  *r\_Tx\_Done     <= 1'b0;*  *r\_Clock\_Count <= 0;*  *r\_Bit\_Index   <= 0;*    *if (start == 1'b1)*  *begin*  *r\_Tx\_Active <= 1'b1;*  *r\_Tx\_Data   <= din;*  *r\_SM\_Main   <= s\_TX\_START\_BIT;*  *end*  *else*  *r\_SM\_Main <= s\_IDLE;*  *end // case: s\_IDLE*      *// Send out Start Bit. Start bit = 0*  *s\_TX\_START\_BIT :*  *begin*  *dout <= 1'b0;*    *// Wait CLKS\_PER\_BIT-1 clock cycles for start bit to finish*  *if (r\_Clock\_Count < CLKS\_PER\_BIT-1)*  *begin*  *r\_Clock\_Count <= r\_Clock\_Count + 1;*  *r\_SM\_Main     <= s\_TX\_START\_BIT;*  *end*  *else*  *begin*  *r\_Clock\_Count <= 0;*  *r\_SM\_Main     <= s\_TX\_DATA\_BITS;*  *end*  *end // case: s\_TX\_START\_BIT*      *// Wait CLKS\_PER\_BIT-1 clock cycles for data bits to finish*  *s\_TX\_DATA\_BITS :*  *begin*  *dout <= r\_Tx\_Data[r\_Bit\_Index];*    *if (r\_Clock\_Count < CLKS\_PER\_BIT-1)*  *begin*  *r\_Clock\_Count <= r\_Clock\_Count + 1;*  *r\_SM\_Main     <= s\_TX\_DATA\_BITS;*  *end*  *else*  *begin*  *r\_Clock\_Count <= 0;*    *// Check if we have sent out all bits*  *if (r\_Bit\_Index < 15)*  *begin*  *r\_Bit\_Index <= r\_Bit\_Index + 1;*  *r\_SM\_Main   <= s\_TX\_DATA\_BITS;*  *end*  *else*  *begin*  *r\_Bit\_Index <= 0;*  *r\_SM\_Main   <= s\_TX\_STOP\_BIT;*  *end*  *end*  *end // case: s\_TX\_DATA\_BITS*      *// Send out Stop bit.  Stop bit = 1*  *s\_TX\_STOP\_BIT :*  *begin*  *dout <= 1'b1;*    *// Wait CLKS\_PER\_BIT-1 clock cycles for Stop bit to finish*  *if (r\_Clock\_Count < CLKS\_PER\_BIT-1)*  *begin*  *r\_Clock\_Count <= r\_Clock\_Count + 1;*  *r\_SM\_Main     <= s\_TX\_STOP\_BIT;*  *end*  *else*  *begin*  *r\_Tx\_Done     <= 1'b1;*  *r\_Clock\_Count <= 0;*  *r\_SM\_Main     <= s\_CLEANUP;*  *r\_Tx\_Active   <= 1'b0;*  *end*  *end // case: s\_Tx\_STOP\_BIT*      *// Stay here 1 clock*  *s\_CLEANUP :*  *begin*  *r\_Tx\_Done <= 1'b1;*  *r\_SM\_Main <= s\_IDLE;*  *end*      *default :*  *r\_SM\_Main <= s\_IDLE;*    *endcase*  *end*    *assign active = r\_Tx\_Active;*  *assign done   = r\_Tx\_Done;*    *endmodule* |

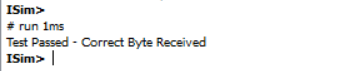
**کد Uart Receiver**

|  |
| --- |
| *module uart\_rx*  *(*  *input         clk,*  *input         din,*  *output        done,*  *output [15:0] dout*  *);*    *// 50000000 / 115200 = 435*  *parameter CLKS\_PER\_BIT = 435;*    *parameter s\_IDLE         = 3'b000;*  *parameter s\_RX\_START\_BIT = 3'b001;*  *parameter s\_RX\_DATA\_BITS = 3'b010;*  *parameter s\_RX\_STOP\_BIT  = 3'b011;*  *parameter s\_CLEANUP      = 3'b100;*    *reg           r\_Rx\_Data\_R = 1'b1;*  *reg           r\_Rx\_Data   = 1'b1;*    *reg [15:0]     r\_Clock\_Count = 0;*  *reg [3:0]       r\_Bit\_Index   = 0;*  *reg [15:0]     r\_Rx\_Byte     = 0;*  *reg             r\_Rx\_DV       = 0;*  *reg [2:0]      r\_SM\_Main     = 0;*    *// Purpose: Double-register the incoming data.*  *// This allows it to be used in the UART RX Clock Domain.*  *// (It removes problems caused by metastability)*  *always @(posedge clk)*  *begin*  *r\_Rx\_Data\_R <= din;*  *r\_Rx\_Data   <= r\_Rx\_Data\_R;*  *end*      *// Purpose: Control RX state machine*  *always @(posedge clk)*  *begin*    *case (r\_SM\_Main)*  *s\_IDLE :*  *begin*  *r\_Rx\_DV       <= 1'b0;*  *r\_Clock\_Count <= 0;*  *r\_Bit\_Index   <= 0;*    *if (r\_Rx\_Data == 1'b0)          // Start bit detected*  *r\_SM\_Main <= s\_RX\_START\_BIT;*  *else*  *r\_SM\_Main <= s\_IDLE;*  *end*    *// Check middle of start bit to make sure it's still low*  *s\_RX\_START\_BIT :*  *begin*  *if (r\_Clock\_Count == (CLKS\_PER\_BIT-1)/2)*  *begin*  *if (r\_Rx\_Data == 1'b0)*  *begin*  *r\_Clock\_Count <= 0;  // reset counter, found the middle*  *r\_SM\_Main     <= s\_RX\_DATA\_BITS;*  *end*  *else*  *r\_SM\_Main <= s\_IDLE;*  *end*  *else*  *begin*  *r\_Clock\_Count <= r\_Clock\_Count + 1;*  *r\_SM\_Main     <= s\_RX\_START\_BIT;*  *end*  *end // case: s\_RX\_START\_BIT*      *// Wait CLKS\_PER\_BIT-1 clock cycles to sample serial data*  *s\_RX\_DATA\_BITS :*  *begin*  *if (r\_Clock\_Count < CLKS\_PER\_BIT-1)*  *begin*  *r\_Clock\_Count <= r\_Clock\_Count + 1;*  *r\_SM\_Main     <= s\_RX\_DATA\_BITS;*  *end*  *else*  *begin*  *r\_Clock\_Count          <= 0;*  *r\_Rx\_Byte[r\_Bit\_Index] <= r\_Rx\_Data;*    *// Check if we have received all bits*  *if (r\_Bit\_Index < 15)*  *begin*  *r\_Bit\_Index <= r\_Bit\_Index + 1;*  *r\_SM\_Main   <= s\_RX\_DATA\_BITS;*  *end*  *else*  *begin*  *r\_Bit\_Index <= 0;*  *r\_SM\_Main   <= s\_RX\_STOP\_BIT;*  *end*  *end*  *end // case: s\_RX\_DATA\_BITS*      *// Receive Stop bit.  Stop bit = 1*  *s\_RX\_STOP\_BIT :*  *begin*  *// Wait CLKS\_PER\_BIT-1 clock cycles for Stop bit to finish*  *if (r\_Clock\_Count < CLKS\_PER\_BIT-1)*  *begin*  *r\_Clock\_Count <= r\_Clock\_Count + 1;*  *r\_SM\_Main     <= s\_RX\_STOP\_BIT;*  *end*  *else*  *begin*  *r\_Rx\_DV       <= 1'b1;*  *r\_Clock\_Count <= 0;*  *r\_SM\_Main     <= s\_CLEANUP;*  *end*  *end // case: s\_RX\_STOP\_BIT*      *// Stay here 1 clock*  *s\_CLEANUP :*  *begin*  *r\_SM\_Main <= s\_IDLE;*  *r\_Rx\_DV   <= 1'b0;*  *end*      *default :*  *r\_SM\_Main <= s\_IDLE;*    *endcase*  *end*    *assign done   = r\_Rx\_DV;*  *assign dout = r\_Rx\_Byte;*    *endmodule // uart\_rx* |

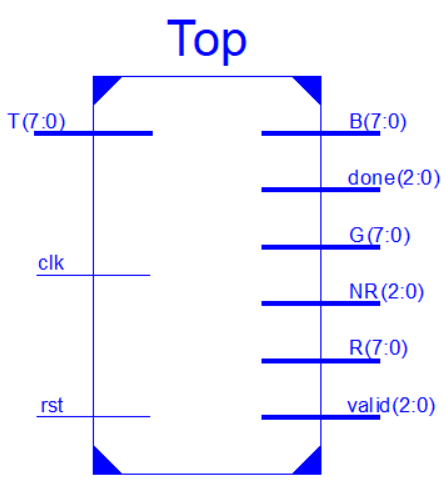
**تست Uart**

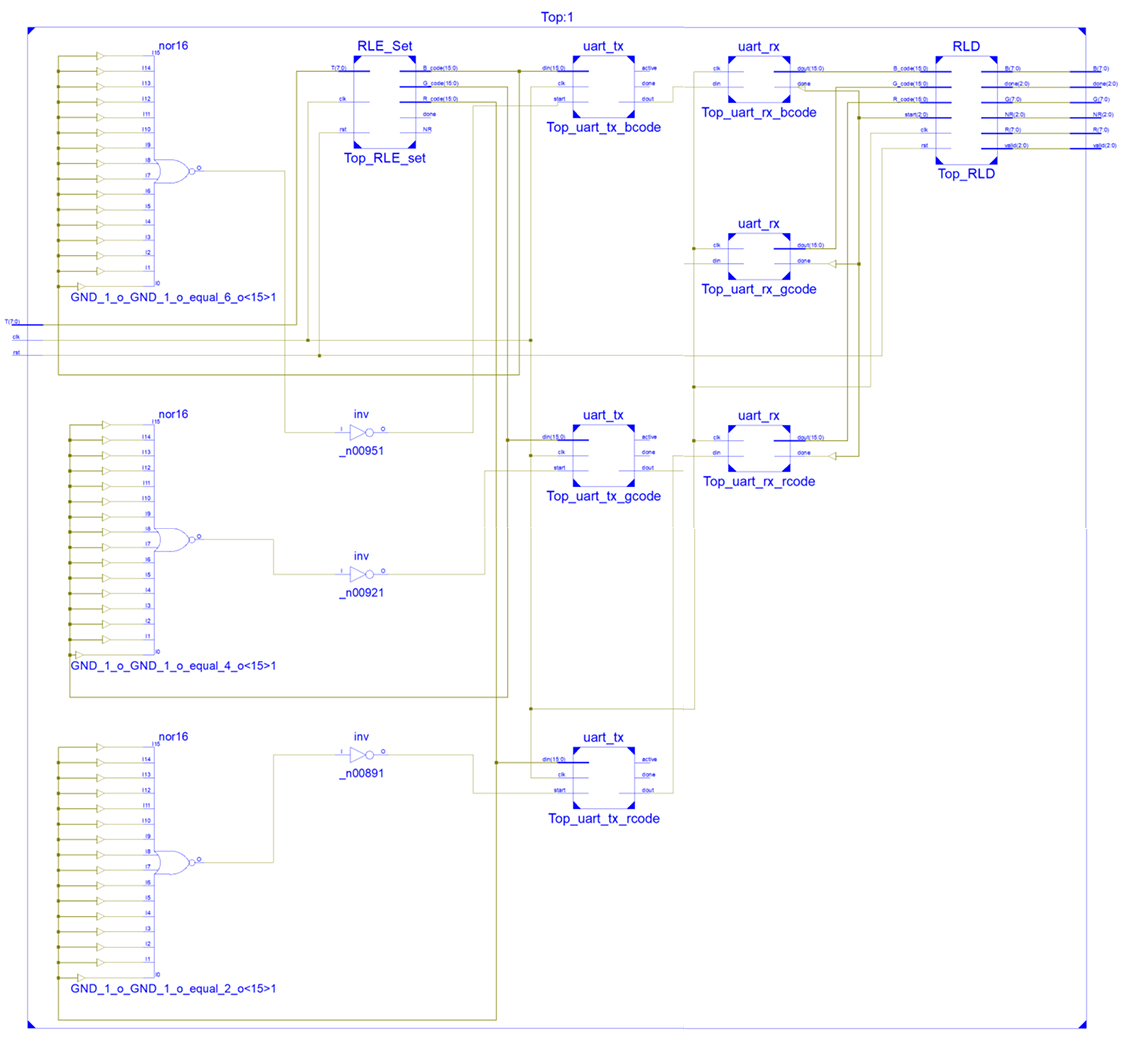
|  |
| --- |
| *module uart\_tb ();*    *// Testbench uses a 50 MHz clock*  *// Want to interface to 115200 baud UART*  *// 50000000 / 115200 = 435 Clocks Per Bit.*  *parameter c\_CLOCK\_PERIOD\_NS = 20;*  *parameter c\_CLKS\_PER\_BIT    = 435;*  *parameter c\_BIT\_PERIOD      = 8700;*    *reg  r\_Clock = 0;*  *reg  r\_Tx\_DV = 0;*  *wire w\_Tx\_Done;*  *reg  [15:0] r\_Tx\_Byte = 0;*  *reg  r\_Rx\_Serial = 1;*  *wire [15:0] w\_Rx\_Byte;*      *// Takes in input byte and serializes it*  *task UART\_WRITE\_BYTE;*  *input   [15:0] i\_Data;*  *integer        ii;*  *begin*    *// Send Start Bit*  *r\_Rx\_Serial <= 1'b0;*  *#(c\_BIT\_PERIOD);*  *#1000;*      *// Send Data Byte*  *for (ii=0; ii<16; ii=ii+1)*  *begin*  *r\_Rx\_Serial <= i\_Data[ii];*  *#(c\_BIT\_PERIOD);*  *end*    *// Send Stop Bit*  *r\_Rx\_Serial <= 1'b1;*  *#(c\_BIT\_PERIOD);*  *end*  *endtask // UART\_WRITE\_BYTE*      *uart\_rx #(.CLKS\_PER\_BIT(c\_CLKS\_PER\_BIT)) UART\_RX\_INST*  *(.clk(r\_Clock),*  *.din(r\_Rx\_Serial),*  *.done(),*  *.dout(w\_Rx\_Byte)*  *);*    *uart\_tx #(.CLKS\_PER\_BIT(c\_CLKS\_PER\_BIT)) UART\_TX\_INST*  *(.clk(r\_Clock),*  *.start(r\_Tx\_DV),*  *.din(r\_Tx\_Byte),*  *.active(),*  *.dout(),*  *.done(w\_Tx\_Done)*  *);*      *always*  *#(c\_CLOCK\_PERIOD\_NS/2) r\_Clock <= !r\_Clock;*      *// Main Testing:*  *initial*  *begin*    *// Tell UART to send a command (exercise Tx)*  *@(posedge r\_Clock);*  *@(posedge r\_Clock);*  *r\_Tx\_DV <= 1'b1;*  *r\_Tx\_Byte <= 16'hAB;*  *@(posedge r\_Clock);*  *r\_Tx\_DV <= 1'b0;*  *@(posedge w\_Tx\_Done);*    *// Send a command to the UART (exercise Rx)*  *@(posedge r\_Clock);*  *UART\_WRITE\_BYTE(16'h3F);*  *@(posedge r\_Clock);*    *// Check that the correct command was received*  *if (w\_Rx\_Byte == 16'h3F)*  *$display("Test Passed - Correct Byte Received");*  *else*  *$display("Test Failed - Incorrect Byte Received");*    *end*    *endmodule* |

****



**نمای شماتیک Top**

****

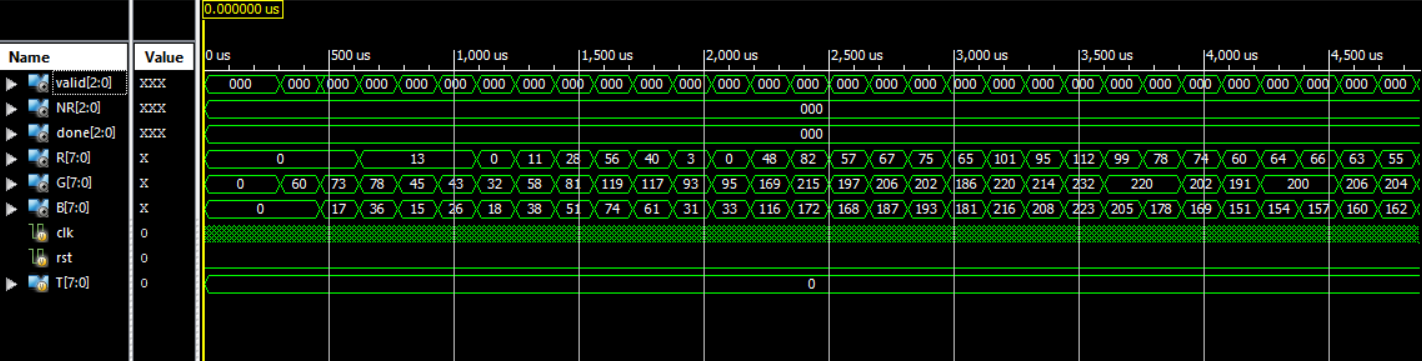


**کد Top**

|  |
| --- |
| *module Top(clk, rst, T, valid, NR, done, R, G, B);*  *input clk, rst;*  *input [7:0] T;*  *output [2:0] valid, NR, done;*  *output [7:0] R, G, B;*    *wire [15:0] rle\_rcode, rle\_gcode, rle\_bcode;*    *wire tx\_rcode\_start, tx\_gcode\_start, tx\_bcode\_start;*  *wire tx\_rcode\_dout, tx\_gcode\_dout, tx\_bcode\_dout;*  *wire rx\_rcode\_done, rx\_gcode\_done, rx\_bcode\_done;*  *wire [15:0] rx\_rcode, rx\_gcode, rx\_bcode;*    *wire [2:0] rld\_start;*    *RLE\_Set Top\_RLE\_set (*  *.clk(clk),*  *.rst(rst),*  *.T(T),*  *.done(),*  *.R\_code(rle\_rcode),*  *.G\_code(rle\_gcode),*  *.B\_code(rle\_bcode)*  *);*  *// 435 clks per bit*  *// each code is 16 bit*  *// 1 bit start 1 bit stop*  *// 435 \* 18 = 7830 clk per code*  *defparam Top\_RLE\_set.HOLD\_CLK = 7830;*      *uart\_tx Top\_uart\_tx\_rcode (*  *.clk(clk),*  *.din(rle\_rcode),*  *.start(tx\_rcode\_start),*  *.dout(tx\_rcode\_dout),*  *.active(),*  *.done()*  *);*  *uart\_tx Top\_uart\_tx\_gcode (*  *.clk(clk),*  *.din(rle\_gcode),*  *.start(tx\_gcode\_start),*  *.dout(tx\_gcode\_dout),*  *.active(),*  *.done()*  *);*  *uart\_tx Top\_uart\_tx\_bcode (*  *.clk(clk),*  *.din(rle\_bcode),*  *.start(tx\_bcode\_start),*  *.dout(tx\_bcode\_dout),*  *.active(),*  *.done()*  *);*      *uart\_rx Top\_uart\_rx\_rcode (*  *.clk(clk),*  *.din(tx\_rcode\_dout),*  *.dout(rx\_rcode),*  *.done(rx\_rcode\_done)*  *);*  *uart\_rx Top\_uart\_rx\_gcode (*  *.clk(clk),*  *.din(tx\_gcode\_dout),*  *.dout(rx\_gcode),*  *.done(rx\_gcode\_done)*  *);*    *uart\_rx Top\_uart\_rx\_bcode (*  *.clk(clk),*  *.din(tx\_bcode\_dout),*  *.dout(rx\_bcode),*  *.done(rx\_bcode\_done)*  *);*      *RLD Top\_RLD (*  *.clk(clk),*  *.rst(rst),*  *.NR(NR),*  *.valid(valid),*  *.done(done),*  *.start(rld\_start),*  *.R\_code(rx\_rcode),*  *.G\_code(rx\_gcode),*  *.B\_code(rx\_bcode),*  *.R(R),*  *.G(G),*  *.B(B)*  *);*    *assign rld\_start = {rx\_bcode\_done, rx\_gcode\_done, rx\_rcode\_done};*    *assign tx\_rcode\_start = (rle\_rcode == 0) ? 0 : 1;*  *assign tx\_gcode\_start = (rle\_gcode == 0) ? 0 : 1;*  *assign tx\_bcode\_start = (rle\_bcode == 0) ? 0 : 1;*  *endmodule* |

**تست Top**

|  |
| --- |
| *module Top\_tb;*  *reg clk, rst;*  *reg [7:0] T;*  *wire [2:0] valid, NR, done;*  *wire [7:0] R, G, B;*    *integer wrf, wgf, wbf;*  *Top uut (*  *.clk(clk),*  *.rst(rst),*  *.T(T),*  *.valid(valid),*  *.NR(NR),*  *.done(done),*  *.R(R),*  *.G(G),*  *.B(B)*  *);*  *initial begin*  *clk = 0;*  *rst = 0;*  *T = 0;*  *#2 rst = 1; #7 rst = 0;*  *end*    *initial forever #10 clk = ~clk;*    *initial begin*  *#100;*    *wrf = $fopen("test/decoded/R.txt", "w");*    *while (~done[0]) begin*  *#20;*  *if (valid[0])*  *$fwrite(wrf, "%0d,", R);*  *if (NR[0])*  *$fwrite(wrf, "\n");*  *end*    *$fclose(wrf);*  *wrf = 0;*  *end*  *initial begin*  *#100;*    *wgf = $fopen("test/decoded/G.txt", "w");*    *while (~done[1]) begin*  *#20;*  *if (valid[1])*  *$fwrite(wgf, "%0d,", G);*  *if (NR[1])*  *$fwrite(wgf, "\n");*  *end*    *$fclose(wgf);*  *wgf = 0;*  *end*  *initial begin*  *#100;*    *wbf = $fopen("test/decoded/B.txt", "w");*    *while (~done[2]) begin*  *#20;*  *if (valid[2])*  *$fwrite(wbf, "%0d,", B);*  *if (NR[2])*  *$fwrite(wbf, "\n");*  *end*    *$fclose(wbf);*  *wbf = 0;*  *end*  *endmodule* |

****

**کد متلب جهت تست خروجی T=0**

|  |
| --- |
| *close all*  *clear all*  *clc*  *root = '..\test\';*  *for i = 1:3*  *rop = strcat(root, 'original\img', num2str(i), '\R.txt');*  *gop = strcat(root, 'original\img', num2str(i), '\G.txt');*  *bop = strcat(root, 'original\img', num2str(i), '\B.txt');*    *rp = strcat(root, 'decoded\img', num2str(i), '-T0\R.txt');*  *gp = strcat(root, 'decoded\img', num2str(i), '-T0\G.txt');*  *bp = strcat(root, 'decoded\img', num2str(i), '-T0\B.txt');*    *ro = dlmread(rop);*  *ro = uint8(ro(1:128, 1:128));*  *go = dlmread(gop);*  *go = uint8(go(1:128, 1:128));*  *bo = dlmread(bop);*  *bo = uint8(bo(1:128, 1:128));*  *r = dlmread(rp);*  *r = uint8(r(1:128, 1:128));*  *g = dlmread(gp);*  *g = uint8(g(1:128, 1:128));*  *b = dlmread(bp);*  *b = uint8(b(1:128, 1:128));*  *r\_equality = isequal(r, ro);*  *g\_equality = isequal(g, go);*  *b\_equality = isequal(b, bo);*  *if (r\_equality && g\_equality && b\_equality)*  *fprintf('Test %d Passed!\n', i);*  *else*  *fprintf('Test %d Faild!\n', i);*  *end*  *end* |

**کد متلب جهت تجزیه RGB تصویر و ساخت فایل ورودی RAM**

|  |
| --- |
| *close all*  *clear all*  *clc*  *root = '..\test\original\';*  *for i = 1:3*  *imgp = strcat(root, 'img', num2str(i), '\img.jpg');*  *rp = strcat(root, 'img', num2str(i), '\R.txt');*  *gp = strcat(root, 'img', num2str(i), '\G.txt');*  *bp = strcat(root, 'img', num2str(i), '\B.txt');*  *coep = strcat(root, 'img', num2str(i), '\RGB.coe');*    *x = imread(imgp);*  *r = x(:,:,1);*  *g = x(:,:,2);*  *b = x(:,:,3);*  *dlmwrite(rp, r);*  *dlmwrite(gp, g);*  *dlmwrite(bp, b);*    *coef = fopen(coep, 'w');*  *fprintf(coef, 'memory\_initialization\_radix = 10;\nmemory\_initialization\_vector =\n');*  *fclose(coef);*    *dlmwrite(coep, r, '-append', 'delimiter', ' ', 'roffset', 2);*  *dlmwrite(coep, g, '-append', 'delimiter', ' ', 'roffset', 2);*  *dlmwrite(coep, b, '-append', 'delimiter', ' ', 'roffset', 2);*    *coef = fopen(coep, 'a');*  *fprintf(coef, ';');*  *fclose(coef);*  *end* |

**کد متلب جهت ترکیب RGB و ساخت تصویر**

|  |
| --- |
| *close all*  *clear all*  *clc*  *root = '..\test\decoded\img';*  *for i = 1:3*  *pt = strcat(root, num2str(i), '-T0', '\');*  *r = dlmread(strcat(pt, 'R.txt'));*  *r = uint8(r(1:128, 1:128));*  *g = dlmread(strcat(pt, 'G.txt'));*  *g = uint8(g(1:128, 1:128));*  *b = dlmread(strcat(pt, 'B.txt'));*  *b = uint8(b(1:128, 1:128));*  *x = cat(3, r, g, b);*  *imwrite(x, strcat(pt, 'img.jpg'));*    *pt = strcat(root, num2str(i), '-T30', '\');*    *r = dlmread(strcat(pt, 'R.txt'));*  *r = uint8(r(1:128, 1:128));*  *g = dlmread(strcat(pt, 'G.txt'));*  *g = uint8(g(1:128, 1:128));*  *b = dlmread(strcat(pt, 'B.txt'));*  *b = uint8(b(1:128, 1:128));*  *x = cat(3, r, g, b);*  *imwrite(x, strcat(pt, 'img.jpg'));*  *end* |