به نام خدا

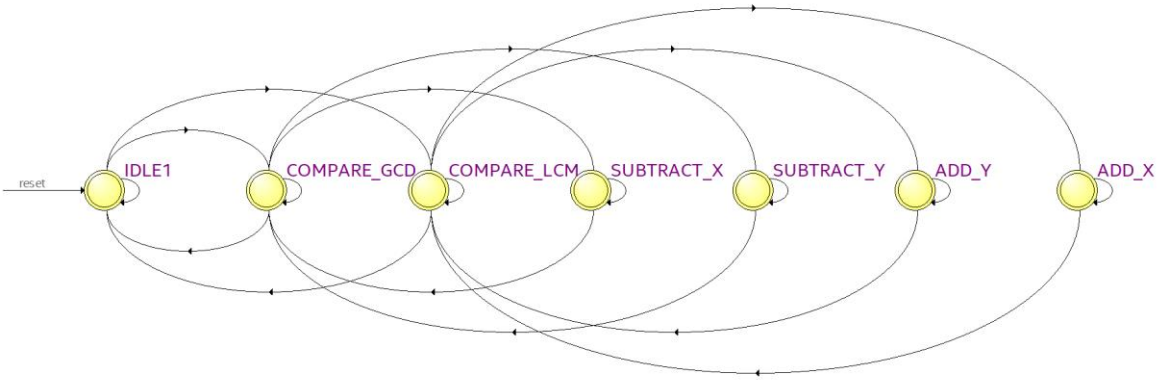
کد کمک پردازنده:

به دو بخش کنترلر و دیتاپث تقسیم می شود:

```
module Coprocessor(
    input logic clk,
    input logic reset,
    input logic start,
    input logic [7:0] x0,
    input logic [7:0] y0,
    input logic Op, // 0 for GCD, 1 for LCM
    output logic Done,
    output logic [7:0] result
);
    logic [7:0] x, y;
    logic load_x, load_y, subtract_x, subtract_y, add_x, add_y;

    Co_datapath datapath_inst (
        .clk(clk),
        .reset(reset),
        .x0(x0),
        .y0(y0),
        .load_x(load_x),
        .load_y(load_y),
        .subtract_x(subtract_x),
        .subtract_y(subtract_y),
        .add_x(add_x),
        .add_y(add_y),
        .x(x),
        .y(y)
    );

    Co_controller controller_inst (
        .clk(clk),
        .reset(reset),
        .start(start),
        .x(x),
        .y(y),
        .Op(Op),
        .load_x(load_x),
        .load_y(load_y),
        .subtract_x(subtract_x),
        .subtract_y(subtract_y),
        .add_x(add_x),
        .add_y(add_y),
        .Done(Done),
        .result(result)
    );
endmodule
```

کنترلر: یک استیت ماشین 7 حالته می باشد

```
module Co_controller(
    input logic clk,
    input logic reset,
    input logic start,
    input logic [7:0] x,
    input logic [7:0] y,
    input logic Op, // 0 for GCD, 1 for LCM
    output logic load_x,
    output logic load_y,
    output logic subtract_x,
    output logic subtract_y,
    output logic add_x,
    output logic add_y,
    output logic Done,
    output logic [7:0] result
);
    typedef enum logic [2:0] {
        IDLE1, COMPARE_LCM, COMPARE_GCD, SUBTRACT_X, SUBTRACT_Y, ADD_X, ADD_Y
    } state_t;

    state_t state, next_state;

    always_ff @(posedge clk or posedge reset) begin
        if (reset) begin
            state <= IDLE1;
        end else begin
            state <= next_state;
        end
    end

    always_comb begin
        load_x = 0;
        load_y = 0;
        subtract_x = 0;
        subtract_y = 0;
        add_x = 0;
        add_y = 0;
        Done = 0;
        result = 8'b0;
        next_state = state;

        if (start) begin
            case (state)
                IDLE1: begin
                    load_x = 1;
                    load_y = 1;
                    if (Op)
                        next_state = COMPARE_LCM; // LCM mode
                    else
                        next_state = COMPARE_GCD; // GCD mode
                end
                COMPARE_GCD: begin
                    if (x == y) begin
                        result = x; // GCD computation
                        next_state = IDLE1;
                        Done = 1;
                    end else if (x > y) begin
                        subtract_x = 1;
                        next_state = SUBTRACT_X;
                    end else begin
                        subtract_y = 1;
                        next_state = SUBTRACT_Y;
                    end
                end
                COMPARE_LCM: begin
                    if (x == y) begin
                        result = x; // LCM computation
                        next_state = IDLE1;
                        Done = 1;
                    end else if (x < y) begin
                        add_x = 1;
                        next_state = ADD_X;
                    end else begin
                        add_y = 1;
                        next_state = ADD_Y;
                    end
                end
                SUBTRACT_X: begin
                    next_state = COMPARE_GCD;
                end
                SUBTRACT_Y: begin
                    next_state = COMPARE_GCD;
                end
                ADD_X: begin
                    next_state = COMPARE_LCM;
                end
                ADD_Y: begin
                    next_state = COMPARE_LCM;
                end
                default: next_state = IDLE1;
            endcase
        end
    end
endmodule
```



دیتاپث: وظیفه محاسبه GCD و LCM را دارد

```
module Co_datapath(
    input logic clk,
    input logic reset,
    input logic [7:0] x0,
    input logic [7:0] y0,
    input logic load_x,
    input logic load_y,
    input logic subtract_x,
    input logic subtract_y,
    input logic add_x,
    input logic add_y,
    output logic [7:0] x,
    output logic [7:0] y
);
    always_ff @(posedge clk or posedge reset) begin
        if (reset) begin
            x <= 8'b0;
            y <= 8'b0;
        end else begin
            if (load_x) x <= x0;
            if (load_y) y <= y0;
            if (subtract_x) x <= x - y;
            if (subtract_y) y <= y - x;
            if (add_x) x <= x + x0;
            if (add_y) y <= y + y0;
        end
    end
endmodule
```

تست بنچ کمک پردازنده: محاسبه ب م م 12 و 18

کوپروسسور به گونه ای کار می کند که ابتدا باید با سیگنال ریست، تمام خروجی ها صفر شود و به استیت Idel برود.

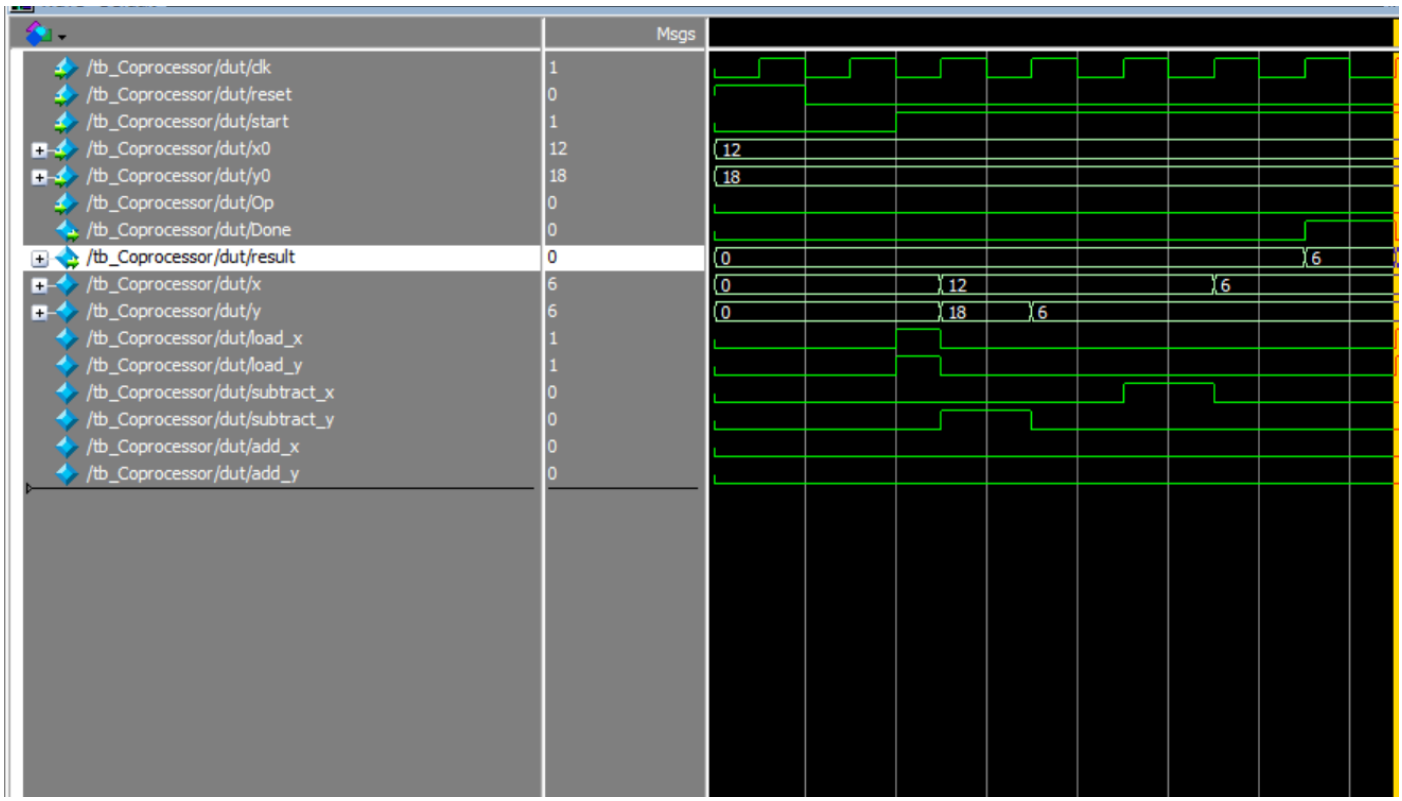
سپس ورودی های x0 و y0 مقداردهی می شوند.

مقدار op نمایانگر حالت محاسبه gcd(0) یا lcm(1) می باشد.

به محض دریافت سیگنال start پردازنده شروع به محاسبه می کند.

```
module tb_Coprocessor;

    // Inputs
    logic clk;
    logic reset;
    logic start;
    logic [7:0] x0;
    logic [7:0] y0;
    logic Op;

    // Outputs
    logic Done;
    logic [7:0] result;

    // Instantiate the Coprocessor module
    Coprocessor dut (
        .clk(clk),
        .reset(reset),
        .start(start),
        .x0(x0),
        .y0(y0),
        .Op(Op),
        .Done(Done),
        .result(result)
    );

    // Clock generation
    always #5 clk = ~clk;

    // Initialize inputs
    initial begin
        clk = 0;
        reset = 1;
        start = 0;
        x0 = 12; // Example input values
        y0 = 18;
        Op = 0; // Compute GCD (change to 1 for LCM)
        #10 reset = 0;
        #10 start = 1;
        #100;
        $display("Result: %d", result);
        $display("Done: %b", Done);
        $finish;
    end

endmodule
```

سیمولیشن پردازنده: طبق ورودی ها result مورد نظر 6 می باشد.



سنتز کمک پردازنده-تایم-مساحت:

Path #1: Delay is 10.599

Path Summary | Statistics | Data Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	10.599	10.599					data path
1	0.000	0.000			1	PIN_AF26	start
2	0.000	0.000	FF	IC	1	IOIBUF_X89_Y4_N78	start~input|i
3	0.866	0.866	FF	CELL	38	IOIBUF_X89_Y4_N78	start~input|o
4	3.441	2.575	FF	IC	1	LABCELL_X83_Y4_N9	controller_inst|result[7]~8|datac
5	3.917	0.476	FF	CELL	1	LABCELL_X83_Y4_N9	controller_inst|result[7]~8|combout
6	7.446	3.529	FF	IC	1	IOOBUF_X80_Y0_N53	result[7]~output|i
7	10.599	3.153	FF	CELL	1	IOOBUF_X80_Y0_N53	result[7]~output|o
8	10.599	0.000	FF	CELL	0	PIN_AH24	result[7]

M10K blocks	0 / 553	0 %
Total MLAB memory bits	0	
Total block memory bits	0 / 5,662,720	0 %
Total block memory implementation bits	0 / 5,662,720	0 %
Total DSP Blocks	0 / 112	0 %
Fractional PLLs	0 / 6	0 %
Global signals	1	
-- Global clocks	1 / 16	6 %
-- Quadrant clocks	0 / 66	0 %
-- Horizontal periphery clocks	0 / 18	0 %
SERDES Transmitters	0 / 100	0 %
SERDES Receivers	0 / 100	0 %
JTAGs	0 / 1	0 %
ASMI blocks	0 / 1	0 %
CRC blocks	0 / 1	0 %
Remote update blocks	0 / 1	0 %
Oscillator blocks	0 / 1	0 %
Impedance control blocks	0 / 4	0 %
Hard Memory Controllers	0 / 1	0 %
Average interconnect usage (total/H/V)	0.0% / 0.0% / 0.0%	
Peak interconnect usage (total/H/V)	1.2% / 1.3% / 1.0%	
Maximum fan-out	38	
Highest non-global fan-out	38	
Total fan-out	428	
Average fan-out	3.10	

Fitter Resource Usage Summary

Resource	Usage	%
Logic utilization (ALMs needed / total ALMs on device)	32 / 41,910	< 1 %
ALMs needed [=A-B+C]	32	
[A] ALMs used in final placement [=a+b+c+d]	33 / 41,910	< 1 %
[a] ALMs used for LUT logic and registers	11	
[b] ALMs used for LUT logic	21	
[c] ALMs used for registers	1	
[d] ALMs used for memory (up to half of total ALMs)	0	
[B] Estimate of ALMs recoverable by dense packing	1 / 41,910	< 1 %
[C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 41,910	0 %
[a] Due to location constrained logic	0	
[b] Due to LAB-wide signal conflicts	0	
[c] Due to LAB input limits	0	
[d] Due to virtual I/Os	0	
Difficulty packing design	Low	
Total LABs: partially or completely used	5 / 4,191	< 1 %
-- Logic LABs	5	
-- Memory LABs (up to half of total LABs)	0	
Combinational ALUT usage for logic	56	
-- 7 input functions	0	
-- 6 input functions	6	
-- 5 input functions	30	
-- 4 input functions	12	
-- <=3 input functions	8	
Combinational ALUT usage for route-throughs	0	
Dedicated logic registers	23	
-- By type:		
-- Primary logic registers	23 / 83,820	< 1 %
-- Secondary logic registers	0 / 83,820	0 %
-- By function:		
-- Design implementation registers	23	
-- Routing optimization registers	0	
Virtual pins	0	
I/O pins	29 / 314	9 %
-- Clock pins	1 / 8	13 %
-- Dedicated input pins	0 / 21	0 %
Hard processor system peripheral utilization		
-- Boot from FPGA	0 / 1 (0 %)	
-- Clock resets	0 / 1 (0 %)	
-- Cross trigger	0 / 1 (0 %)	
-- S2F AXI	0 / 1 (0 %)	
-- F2S AXI	0 / 1 (0 %)	
-- AXI Lightweight	0 / 1 (0 %)	
-- SDRAM	0 / 1 (0 %)	
-- Interrupts	0 / 1 (0 %)	
-- JTAG	0 / 1 (0 %)	
-- Loan I/O	0 / 1 (0 %)	
-- MPU event standby	0 / 1 (0 %)	
-- MPU general purpose	0 / 1 (0 %)	
-- STM event	0 / 1 (0 %)	
-- TPIU trace	0 / 1 (0 %)	
-- DMA	0 / 1 (0 %)	
-- CAN	0 / 2 (0 %)	
-- EMAC	0 / 2 (0 %)	
-- I2C	0 / 4 (0 %)	
-- NAND Flash	0 / 1 (0 %)	
-- QSPI	0 / 1 (0 %)	
-- SDMMC	0 / 1 (0 %)	
-- SPI Master	0 / 2 (0 %)	
-- SPI Slave	0 / 2 (0 %)	
-- UART	0 / 2 (0 %)	
-- USB	0 / 2 (0 %)	

سیستم ریسک فایو در کنار کمک پردازنده:

ادرس دیکودر: مقداردهی enable

```
module addressD(
input logic [6:0] op,
output logic MemWrite
);

logic control;
assign MemWrite = control;
always_comb
case(op)

7'b0000011: control = 0; // lw
7'b0100011: control = 1; // sw
7'b0110011: control = 0; // R-type
7'b1100011: control = 0; // beq/bne
7'b0010011: control = 0; // I-type
7'b1101111: control = 0; // jal
7'b1100111: control = 0; // jalr

7'b0000000: control = 0; // GCD
7'b1111111: control = 0; // LCM

default: control = 0; // ???
endcase

endmodule
```

سیگنال های جدید:

```
logic cp_op;
logic IO;
```

Cp_op نوع دستور gcd را از lcm تمایز می دهد

IO سیگنالی است که نشاندهنده نیازمندی به استفاده از IO می باشد

از این سیگنال ها در دیکودر و دیتاپث استفاده می کنیم

```
module maindec(input logic [6:0] op,
               output logic [1:0] ResultSrc,
               output logic nop,
               output logic Branch, ALUSrc,
               output logic RegWrite, Jump,
               output logic [1:0] ImmSrc,
               output logic [1:0] ALUOp,
               output logic PCTargetSRC, cp_op, IO
               );
logic [13:0] controls;
assign {IO ,cp_op, RegWrite, ImmSrc, ALUSrc, nop,ResultSrc, Branch, ALUOp, Jump,PCTargetSRC} = controls;
always_comb
case(op)
// RegWrite_ImmSrc_ALUSrc_MemWrite_ResultSrc_Branch_ALUOp_Jump
7'b0000011: controls = 14'b0_x_1_00_1_0_01_0_00_0_x; // lw
7'b0100011: controls = 14'b0_x_0_01_1_1_xx_0_00_0_x; // sw
7'b0110011: controls = 14'b0_x_1_xx_0_0_00_0_10_0_x; // R-type
7'b1100011: controls = 14'b0_x_0_10_0_0_xx_1_01_0_0; // beq/bne
7'b0010011: controls = 14'b0_x_1_00_1_0_00_0_10_0_x; // I-type
7'b1101111: controls = 14'b0_x_1_11_x_0_10_0_xx_1_0; // jal
7'b1100111: controls = 14'b0_x_0_00_1_0_10_0_xx_1_1; // jalr

7'b0000000: controls = 14'b1_0_1_xx_x_0_11_0_xx_0_0; // GCD
7'b1111111: controls = 14'b1_1_1_xx_x_0_11_0_xx_0_0; // LCM

default: controls = 14'bx_x_xx_x_x_xx_x_xx_x_x; // ???
endcase
endmodule
```

سیگنال های جدید برای کنترل ورودی کمک پردازنده:

```
logic [31:0] cp_result;
logic cp_r=0;
logic cp_s=0;
logic cp_done;

logic enable=1;
```

از cp_r برای ریست کردن استفاده می کنیم

از cp_s برای استارت کردن استفاده می کنیم

Cp_done خروجی پردازنده هست و اعلام می کند که نتیجه یا result آماده می باشد

نحوه هندل کردن کمک پردازنده:

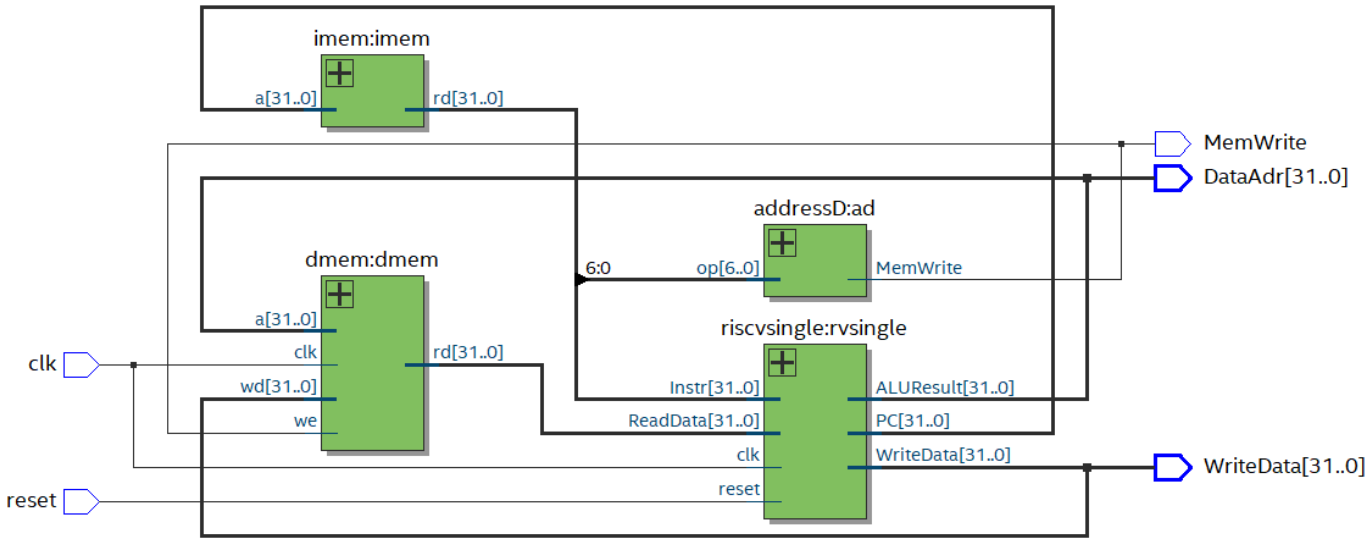
```
Coprocessor cop(clk, cp_r, cp_s, SrcA, WriteData, cp_op, cp_done, cp_result);

logic reset_done=0;
logic start_done=0;
always_ff @(posedge clk) begin
    if(IO) begin
    $display("IO is 1");
        if(!reset_done) begin
            cp_r = 1;
            reset_done = 1;
            $display("reset signal sent");
        end else if(!start_done) begin
            $display("src A: %b", SrcA);
            $display("src B: %b", SrcB);
            cp_r = 0;
            cp_s = 1;
            start_done = 1;
            $display("start signal sent");
            end

    end

end

always_comb begin
    if(IO) begin
        if(!cp_done)begin
            enable = 0;
            $display("pc is halted");
        end else begin
            enable = 1;
            $display("pc is started");
            end
    end else enable =1;

end


endmodule
```

برای هندل و کنترل کردن کمک پردازنده به دو بلاک always نیازمندیم.

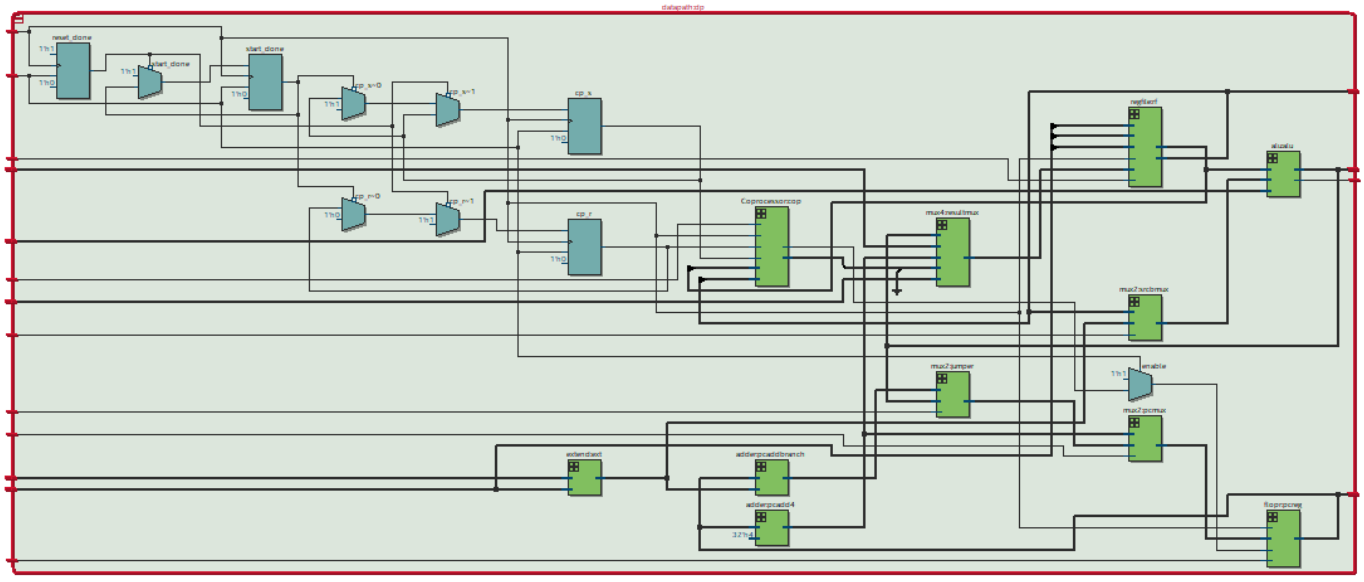
always_ff: این بلاک تشخیصی دهد که آیا به استفاده IO نیاز داریم یا خیر در این صورت در یک کلاک اقدام به ریست کردن کمک پردازنده و در کلاک بعدی اقدام به استارت آن می کند.

always_comb: این بلاک پس تشخیص نیازمندی به IO، رجیستر PC/PCnext را غیز فعال یا disable می کند و پس از پایان کار کمک پردازنده سیستم به روال قبل بر می گردد.

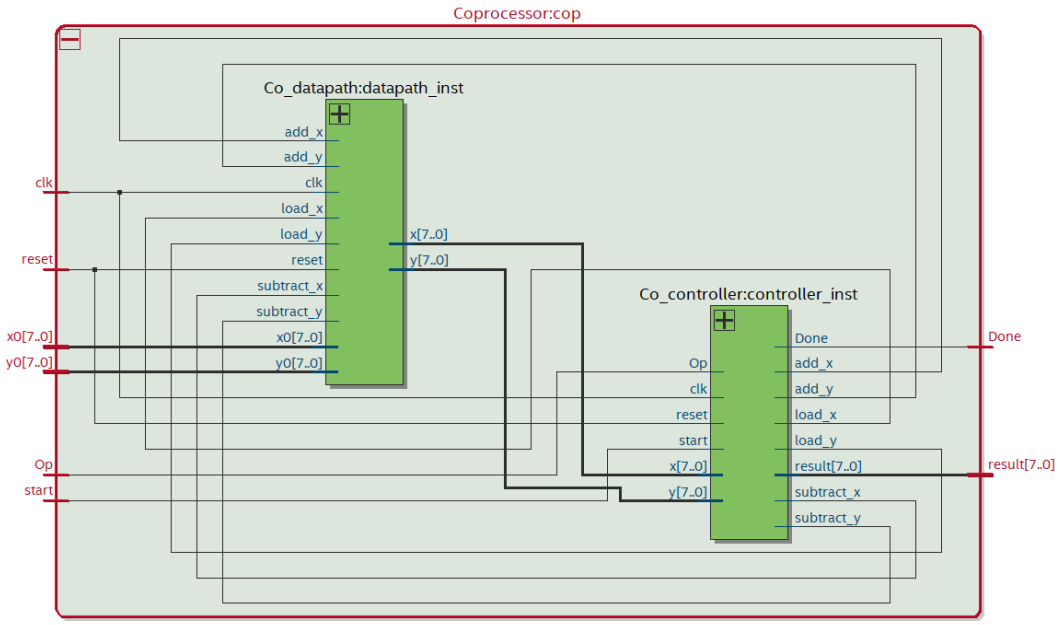
:top


imem:imem
a[31..0]
rd[31..0]
MemWrite
DataAdr[31..0]
addressD:ad
6:0
op[6..0]
MemWrite
dmem:dmem
a[31..0]
clk
wd[31..0]
we
rd[31..0]
riscvsingle:rvsingle
Instr[31..0]
ReadData[31..0]
clk
reset
ALUResult[31..0]
PC[31..0]
WriteData[31..0]
clk
reset
WriteData[31..0]

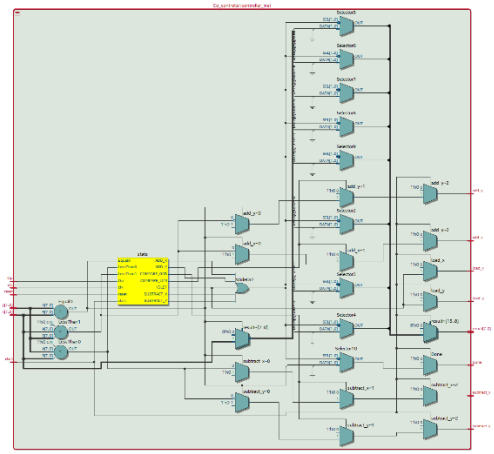
:CP



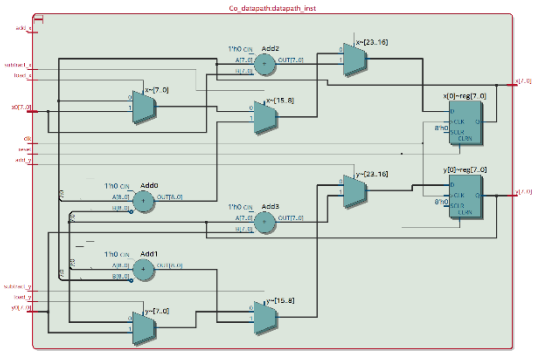
اجزای cp:



کنترلر cp:



دیتا cp :



سنتز کمک پردازنده-تایم-مساحت:

Path #1: Delay is 16.614

Path Summary | Statistics | Data Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	⌄ 16.614	16.614					data path
1	0.000	0.000			1	FF_X79_Y16_N32	riscvsingle:rvsingle|datapath:dp|flopr:pcreg|q[4]~DUPLICATE
2	0.000	0.000	RR	CELL	11	FF_X79_Y16_N32	rvsingle|dp|pcreg|q[4]~DUPLICATE|q
3	0.318	0.318	RR	IC	1	LABCELL_X79_Y16_N48	imem|RAM~14|datab
4	0.875	0.557	RF	CELL	4	LABCELL_X79_Y16_N48	imem|RAM~14|combout
5	1.099	0.224	FF	IC	1	LABCELL_X79_Y16_N51	imem|RAM~15|dataf
6	1.184	0.085	FF	CELL	33	LABCELL_X79_Y16_N51	imem|RAM~15|combout
7	2.546	1.362	FF	IC	1	LABCELL_X75_Y13_N42	rvsingle|dp|rf|rf~1198|datac
8	3.060	0.514	FF	CELL	3	LABCELL_X75_Y13_N42	rvsingle|dp|rf|rf~1198|combout
9	4.169	1.109	FF	IC	2	LABCELL_X77_Y16_N54	rvsingle|dp|alu|Add0~69|dataf
10	5.284	1.115	FF	CELL	1	LABCELL_X77_Y16_N54	rvsingle|dp|alu|Add0~69|cout
11	5.284	0.000	FF	IC	2	LABCELL_X77_Y16_N57	rvsingle|dp|alu|Add0~73|cin
12	5.284	0.000	FF	CELL	1	LABCELL_X77_Y16_N57	rvsingle|dp|alu|Add0~73|cout
13	5.284	0.000	FF	IC	2	LABCELL_X77_Y15_N0	rvsingle|dp|alu|Add0~77|cin
14	5.810	0.526	FF	CELL	1	LABCELL_X77_Y15_N0	rvsingle|dp|alu|Add0~77|sumout
15	6.754	0.944	FF	IC	1	LABCELL_X79_Y13_N15	rvsingle|dp|alu|Mux12~0|dataf
16	6.837	0.083	FF	CELL	2	LABCELL_X79_Y13_N15	rvsingle|dp|alu|Mux12~0|combout
17	13.471	6.634	FF	IC	1	IOOBUF_X38_Y81_N53	DataAdr[19]~output|i
18	16.614	3.143	FF	CELL	1	IOOBUF_X38_Y81_N53	DataAdr[19]~output|o
19	16.614	0.000	FF	CELL	0	PIN_D8	DataAdr[19]

Fitter Resource Usage Summary

Resource	Usage	%
Logic utilization (ALMs needed / total ALMs on device)	1,453 / 41,910	3 %
ALMs needed [=A-B+C]	1,453	
[A] ALMs used in final placement [=a+b+c+d]	1,865 / 41,910	4 %
[a] ALMs used for LUT logic and registers	355	
[b] ALMs used for LUT logic	680	
[c] ALMs used for registers	830	
[d] ALMs used for memory (up to half of total ALMs)	0	
[B] Estimate of ALMs recoverable by dense packing	418 / 41,910	< 1 %
[C] Estimate of ALMs unavailable [=a+b+c+d]	6 / 41,910	< 1 %
[a] Due to location constrained logic	0	
[b] Due to LAB-wide signal conflicts	4	
[c] Due to LAB input limits	2	
[d] Due to virtual I/Os	0	
Difficulty packing design	Low	
Total LABs: partially or completely used	321 / 4,191	8 %
-- Logic LABs	321	
-- Memory LABs (up to half of total LABs)	0	
Combinational ALUT usage for logic	1,366	
-- 7 input functions	15	
-- 6 input functions	935	
-- 5 input functions	79	
-- 4 input functions	76	
-- <=3 input functions	261	
Combinational ALUT usage for route-throughs	1,065	
Dedicated logic registers	2,708	
-- By type:		
-- Primary logic registers	2,369 / 83,820	3 %
-- Secondary logic registers	339 / 83,820	< 1 %
-- By function:		
-- Design implementation registers	2,683	
-- Routing optimization registers	25	
Virtual pins	0	
I/O pins	67 / 314	21 %

-- Clock pins	4 / 8	50 %
-- Dedicated input pins	0 / 21	0 %
Hard processor system peripheral utilization		
-- Boot from FPGA	0 / 1 (0 %)	
-- Clock resets	0 / 1 (0 %)	
-- Cross trigger	0 / 1 (0 %)	
-- S2F AXI	0 / 1 (0 %)	
-- F2S AXI	0 / 1 (0 %)	
-- AXI Lightweight	0 / 1 (0 %)	
-- SDRAM	0 / 1 (0 %)	
-- Interrupts	0 / 1 (0 %)	
-- JTAG	0 / 1 (0 %)	
-- Loan I/O	0 / 1 (0 %)	
-- MPU event standby	0 / 1 (0 %)	
-- MPU general purpose	0 / 1 (0 %)	
-- STM event	0 / 1 (0 %)	
-- TPIU trace	0 / 1 (0 %)	
-- DMA	0 / 1 (0 %)	
-- CAN	0 / 2 (0 %)	
-- EMAC	0 / 2 (0 %)	
-- I2C	0 / 4 (0 %)	
-- NAND Flash	0 / 1 (0 %)	
-- QSPI	0 / 1 (0 %)	
-- SDMMC	0 / 1 (0 %)	
-- SPI Master	0 / 2 (0 %)	
-- SPI Slave	0 / 2 (0 %)	
-- UART	0 / 2 (0 %)	
-- USB	0 / 2 (0 %)	
M10K blocks	0 / 553	0 %
Total MLAB memory bits	0	
Total block memory bits	0 / 5,662,720	0 %
Total block memory implementation bits	0 / 5,662,720	0 %
Total DSP Blocks	0 / 112	0 %
Fractional PLLs	0 / 6	0 %

Global signals	1	
-- Global clocks	1 / 16	6 %
-- Quadrant clocks	0 / 66	0 %
-- Horizontal periphery clocks	0 / 18	0 %
SERDES Transmitters	0 / 100	0 %
SERDES Receivers	0 / 100	0 %
JTAGs	0 / 1	0 %
ASMI blocks	0 / 1	0 %
CRC blocks	0 / 1	0 %
Remote update blocks	0 / 1	0 %
Oscillator blocks	0 / 1	0 %
Impedance control blocks	0 / 4	0 %
Hard Memory Controllers	0 / 1	0 %
Average interconnect usage (total/H/V)	2.0% / 2.0% / 2.0%	
Peak interconnect usage (total/H/V)	37.0% / 37.1% / 36.6%	
Maximum fan-out	2708	
Highest non-global fan-out	293	
Total fan-out	16533	
Average fan-out	3.13	

سیمولیشن سیستم: رایت کردن دو مقادر در حافظه و خواندن آن ها و سپس محاسبه GCD و LCM

risc.txt

```
addi s1, zero, 0
addi s2, zero, 6
addi s3, zero, 7
sw s2, 0(s1)
sw s3, 4(s1)
lw s4, 0(s1)
lw s5, 4(s1)
GCD s6, s5, s4
LMC s7, s5, s4
```

باینری کردن دو دستور جدید:

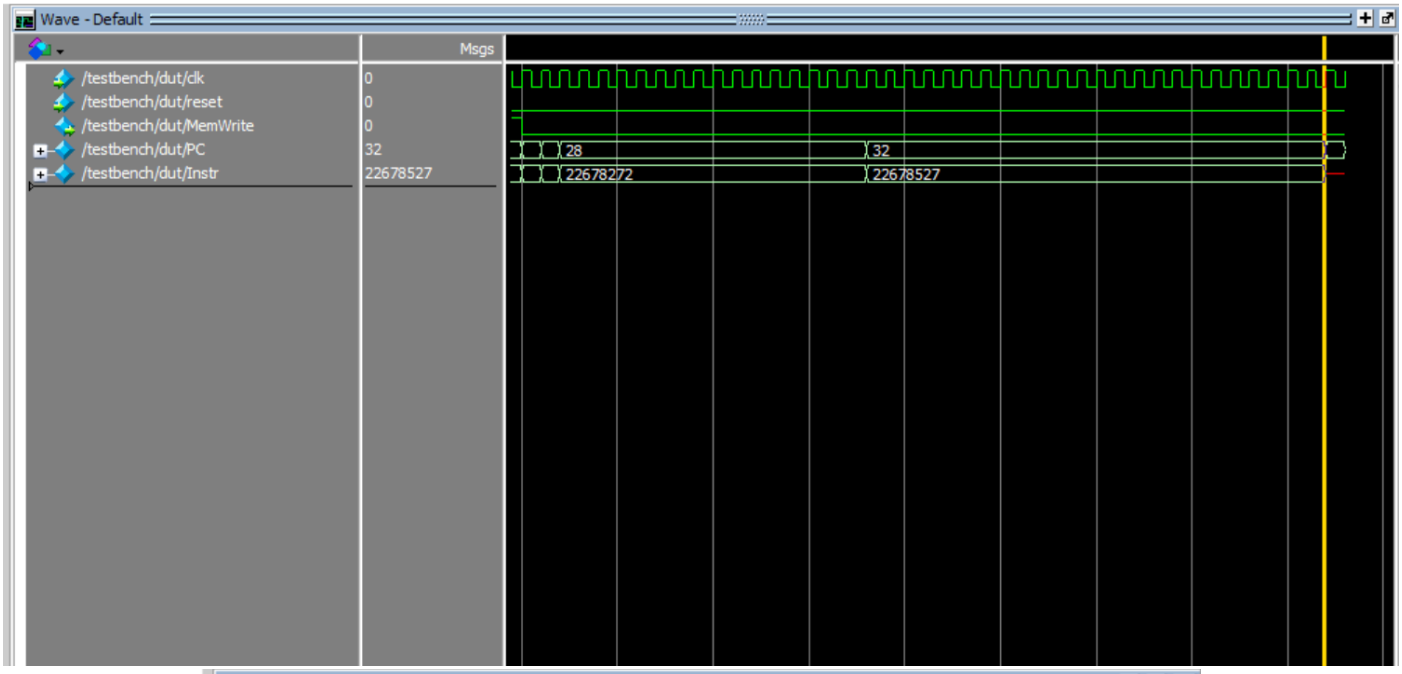
binary_test.txt

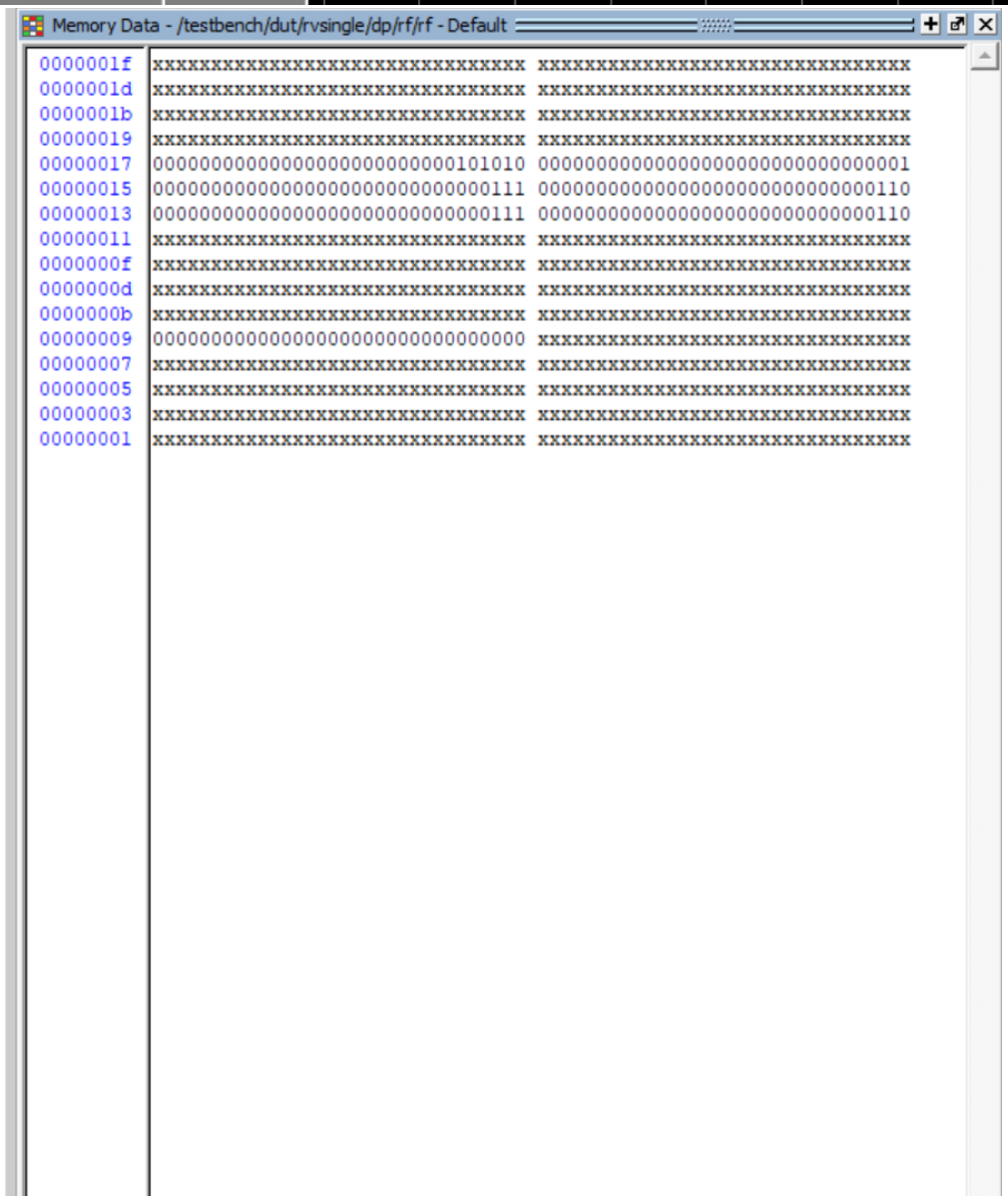
```
015a0b00
015a0bff


0000 000/1 0101 /1010 0/000 /1011 0/000 0000

0000 000/1 0101 /1010 0/000 /1011 1/111 1111
```

زبان سی:

```c
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>

// State enumeration
typedef enum {
    IDLE1, COMPARE_LCM, COMPARE_GCD, SUBTRACT_X, SUBTRACT_Y, ADD_X, ADD_Y
} state_t;

// Coprocessor struct to hold state and variables
typedef struct {
    uint8_t x, y;
    state_t state, next_state;
    bool Done;
    uint8_t result;
} Coprocessor;

void Coprocessor_reset(Coprocessor *cp) {
    printf("cp reseted\n");
    cp->state = IDLE1;
    cp->Done = false;
}

void Coprocessor_start(Coprocessor *cp, uint8_t x0, uint8_t y0, bool Op) {
    cp->x = x0;
    cp->y = y0;
    cp->Done = false;
    cp->state = IDLE1;

    while (!cp->Done) {
        cp->next_state = cp->state; // Default to hold state

        switch (cp->state) {
            case IDLE1:
                printf("entered idle1\n");
                if (Op)
                    cp->next_state = COMPARE_LCM; // LCM mode
                else
                    cp->next_state = COMPARE_GCD; // GCD mode
                break;

            case COMPARE_GCD:
                printf("entered GCD compare\n");
                if (cp->x == cp->y) {
                    cp->result = cp->x; // GCD computation
                    cp->next_state = IDLE1;
                    cp->Done = true;
                    printf("we are done\n");
                } else if (cp->x > cp->y) {
                    cp->next_state = SUBTRACT_X;
                    printf("x: %d\n", cp->x);
                    printf("y: %d\n", cp->y);
                    printf("not done yet\n");
                } else {
                    cp->next_state = SUBTRACT_Y;
                    printf("not done yet\n");
                }
                break;

            case COMPARE_LCM:
                printf("entered LCM compare\n");
                if (cp->x == cp->y) {
                    cp->result = cp->x; // LCM computation
                    cp->next_state = IDLE1;
                    cp->Done = true;
                    printf("we are done\n");
                } else if (cp->x < cp->y) {
                    cp->next_state = ADD_X;
                    printf("not done yet\n");
                } else {
                    cp->next_state = ADD_Y;
                    printf("not done yet\n");
                }
                break;

            case SUBTRACT_X:
                printf("subtracting\n");
                cp->x -= cp->y;
                cp->next_state = COMPARE_GCD;
                break;

            case SUBTRACT_Y:
                printf("subtracting\n");
                cp->y -= cp->x;
                cp->next_state = COMPARE_GCD;
                break;

            case ADD_X:
                printf("adding\n");
                cp->x += x0;
                cp->next_state = COMPARE_LCM;
                break;

            case ADD_Y:
                printf("adding\n");
                cp->y += y0;
                cp->next_state = COMPARE_LCM;
                break;

            default:
                cp->next_state = IDLE1;
                break;
        }

        cp->state = cp->next_state;
    }
}

int main() {
    Coprocessor cp;
    uint8_t x0 = 6, y0 = 7;
    bool Op = 0; // 0 for GCD, 1 for LCM

    Coprocessor_reset(&cp);
    Coprocessor_start(&cp, x0, y0, Op);

    printf("Result: %d\n", cp.result);

    bool = 1;

    printf("Result: %d\n", cp.result);


    return 0;
}
```

سیمولیشن:


Wave - Default
Msgs
/testbench/dut/clk
/testbench/dut/reset
/testbench/dut/MemWrite
/testbench/dut/PC
/testbench/dut/Instr
0
0
0
32
22678527
28
32
22678272
22678527


Memory Data - /testbench/dut/rvsingle/dp/rf/rf - Default
0000001f xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0000001d xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0000001b xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00000019 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00000017 00000000000000000000000000101010 00000000000000000000000000000001
00000015 00000000000000000000000000000111 00000000000000000000000000000110
00000013 00000000000000000000000000000111 00000000000000000000000000000110
00000011 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0000000f xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0000000d xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0000000b xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00000009 00000000000000000000000000000000 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00000007 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00000005 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00000003 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00000001 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx