> Part of the code concerning UART in TOP module:

```
////////////// UART Programmer Pinouts //////////////
    wire upg_clk, upg_clk_o;
    wire upg_wen_o; //Uart write out enable
    wire upg_done_o; //Uart iFpgaUartFromPc data have done
    //data to which memory unit of program_rom/dmemory32
    wire [14:0] upg_adr_o;
    //data to program_rom or dmemory32
    wire [31:0] upg_dat_o;
    wire spg_bufg;
    BUFG U1(.I(iStartReceiveCoe), .O(spg_bufg)); // de-twitter
    // Generate UART Programmer reset signal
    reg upg_rst;
    always @ (posedge iFpgaClk) begin
    if (spg_bufg) upg_rst = 0;
    if (iFpgaRst) upg_rst = 1;
    end
    //used for other modules which don't relate to UART
    wire rst;
    assign rst = iFpgaRst | !upg_rst;

    /* if kickOff is 1 means CPU work on normal mode, otherwise CPU work on Uart
communication mode */
    wire kickOff = upg_rst | (~upg_rst & upg_done_o );

    uart_bmpg_0 uart_instance(
        .upg_clk_i(upg_clk),
        .upg_rst_i(upg_rst),
        .upg_rx_i(iFpgaUartFromPc),
        .upg_clk_o(upg_clk_o),
        .upg_wen_o(upg_wen_o),
        .upg_adr_o(upg_adr_o),
        .upg_dat_o(upg_dat_o),
        .upg_done_o(upg_done_o)
    );
```

> I don't know what the hell it is.

```
`timescale 1ns / 1ps


module programrom (
// Program ROM Pinouts
input rom_clk_i, // ROM clock
input [13:0] rom_adr_i, // From IFetch
output [31:0] Instruction_o, // To IFetch
// UART Programmer Pinouts
input upg_rst_i, // UPG reset (Active High)
```

```
input upg_clk_i, // UPG clock (10MHz)
input upg_wen_i, // UPG write enable
input[13:0] upg_adr_i, // UPG write address
input[31:0] upg_dat_i, // UPG write data
input upg_done_i // 1 if program finished
);

wire kickOff = upg_rst_i | (~upg_rst_i & upg_done_i );
prgrom instmem (
.clka (kickOff ? rom_clk_i : upg_clk_i ),
.wea (kickOff ? 1'b0 : upg_wen_i ),
.addra (kickOff ? rom_adr_i : upg_adr_i ),
.dina (kickOff ? 32'h00000000 : upg_dat_i ),
.douta (Instruction_o) );
endmodule
```

> Here are codes related to `UART` in `dmemory` and `instruction memory`:

```
module DataMemory (
    input ram_clk_i, // from CPU top
    input ram_wen_i, // from Controller
    input [13:0] ram_adr_i, // from alu_result of ALU
    input [31:0] ram_dat_i, // from read_data_2 of Decoder
    output [31:0] ram_dat_o, // the data read from data-ram
    // UART Programmer Pinouts
    input upg_rst_i, // UPG reset (Active High)
    input upg_clk_i, // UPG ram_clk_i (10MHz)
    input upg_wen_i, // UPG write enable
    input [13:0] upg_adr_i, // UPG write address
    input [31:0] upg_dat_i, // UPG write data
    input upg_done_i // 1 if programming is finished
);
    wire ram_clk = !ram_clk_i;
    /* CPU work on normal mode when kickOff is 1. CPU work on Uart communicate
mode when kickOff is 0.*/
    wire kickOff = upg_rst_i | (~upg_rst_i & upg_done_i);
    RAM ram (
        .clka (kickOff ? ram_clk : upg_clk_i),
        .wea (kickOff ? ram_wen_i : upg_wen_i),
        .addra (kickOff ? ram_adr_i : upg_adr_i),
        .dina (kickOff ? ram_dat_i : upg_dat_i),
        .douta (ram_dat_o)
    );

//assign  upg_wen_i=upg_wen_o & upg_adr_o[14];
endmodule
```

```
module InstructionMemory (
    // Program ROM Pinouts
```

```
    input iRomClock, // ROM clock
    input[13:0] iAddressRequested, // From IFetch
    output [31:0] oInstructionFetched, // To IFetch
    // UART Programmer Pinouts
    input iUpgReset, // UPG reset (Active High)
    input iUpgClock, // UPG clock (10MHz)
    input iDoUpgWrites, // UPG write enable
    input[13:0] iUpgWriteAddress, // UPG write address
    input[31:0] iUpgWriteData, // UPG write data
    input iIsUpgDone // 1 if program finished
);
    /* if kickOff is 1 means CPU work on normal mode, otherwise CPU work on Uart
communication mode */
    wire kickOff = iUpgReset | (~iUpgReset & iIsUpgDone );
    prgrom instmem (
        .clka (kickOff ? iRomClock : iUpgClock ),
        .wea (kickOff ? 1'b0 : iDoUpgWrites ),
        .addra (kickOff ? iAddressRequested : iUpgWriteAddress ),
        .dina (kickOff ? 32'h00000000 : iUpgWriteData ),
        .douta (oInstructionFetched)
    );
endmodule
```