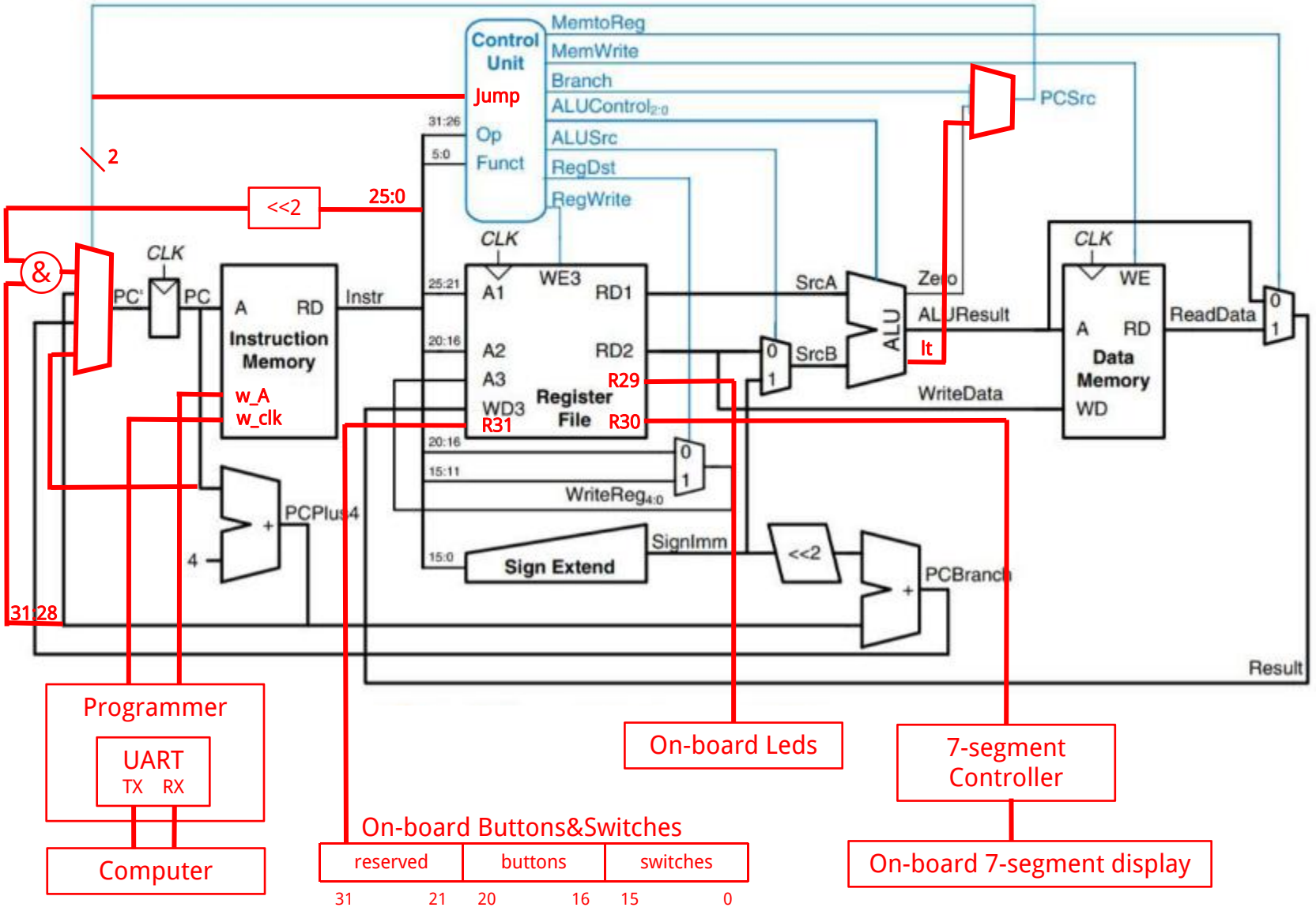
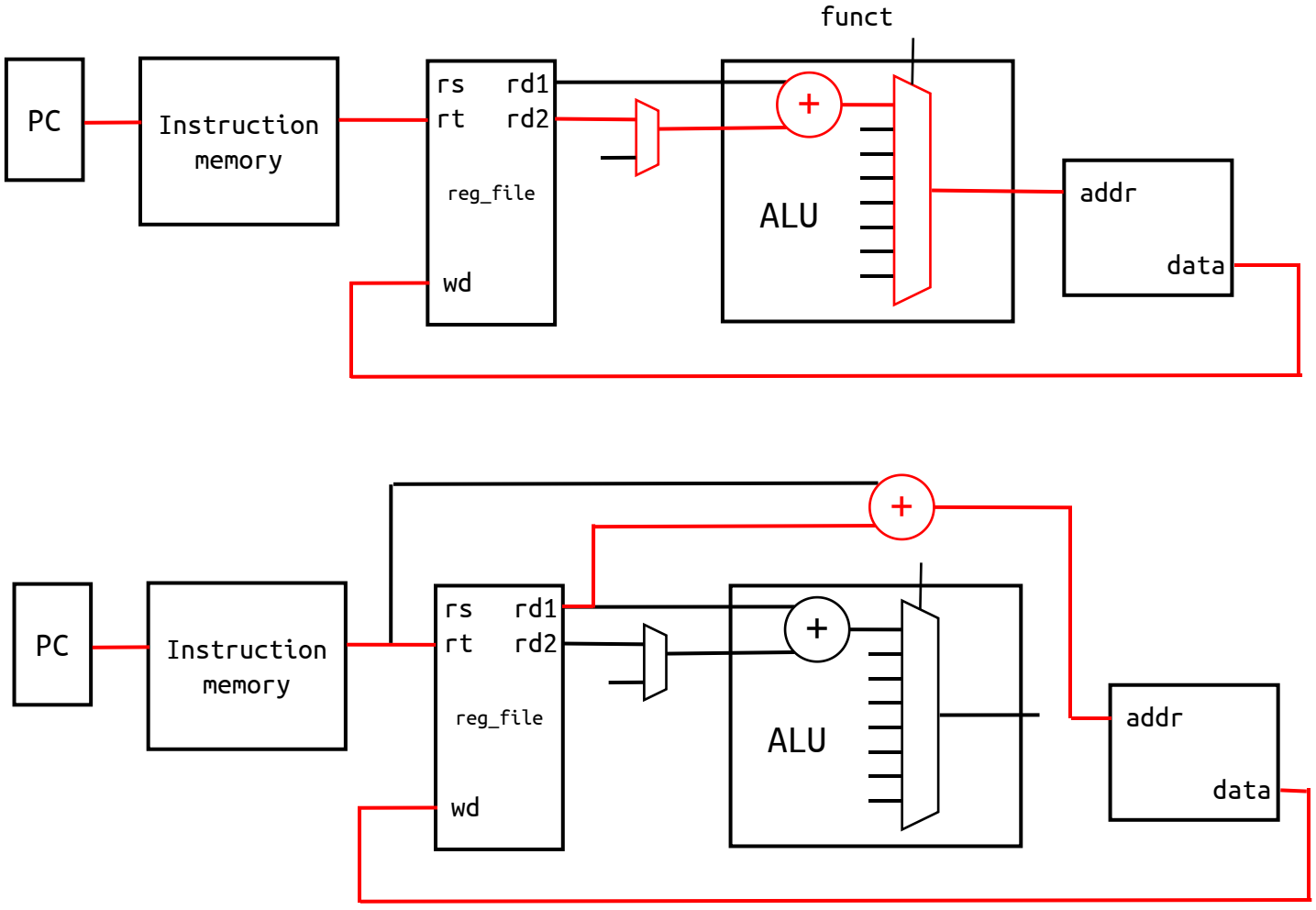
# Specifications

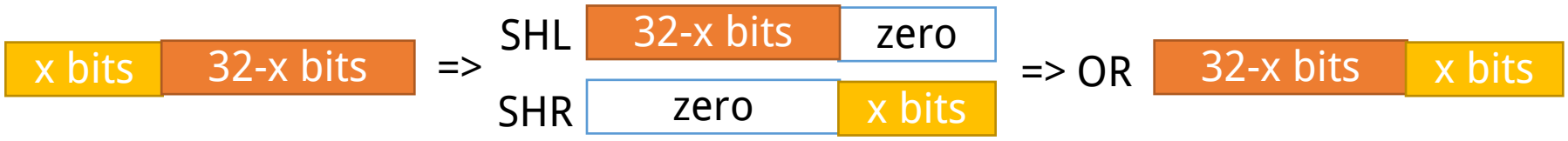
| Metric | value |
| --- | --- |
| cycles for encrytion/decryption | 513 |
| cycles for round key generation | 2574 |
| clock frequency | 135MHz (on-board)<br>76.8MHz (timing analysis) |
| utilization (LUTs) | 3.27% |
| test cases for functional | 1000*3 + 10*100 |
| test cases for timing | 5*3 + 2*5 |


Control Unit
Jump
Op
Funct
MemtoReg
MemWrite
Branch
ALUControl2:0
ALUSrc
RegDst
RegWrite
PCSrc
31:26
5:0
2
<<2
25:0
CLK
PC'
PC
A
RD
Instruction Memory
Instr
w_A
w_clk
25:21
20:16
A1
A2
A3
WD3
WE3
RD1
RD2
Register File
R29
R31
R30
SrcA
SrcB
ALU
Zero
ALUResult
It
WriteData
WE
Data Memory
WD
ReadData
0
1
20:16
15:11
WriteReg4:0
PCPlus4
4
15:0
Sign Extend
SignImm
<<2
PCBranch
31:28
Result
Programmer
UART
TX RX
Computer
On-board Leds
7-segment Controller
On-board 7-segment display
On-board Buttons&Switches
reserved
buttons
switches
31
21
20
16
15
0

# Critical path delay



# Rotation optimization




```
ORI  r6  r0  0      # rotation counter
AND  r16 r26 r8     # A: 5 LSB rotation bits
BEQ  r0  r0  2
SHL  r11 r11 1      # r11 <- r11 << 1
ADDI r6  r6  1      # r6 += 1
BLT  r16 r6  -3     # loop if r6 < rotation

ORI  r16 r0  32     # total rotation bits =
BEQ  r0  r0  2
SHR  r10 r10 1      # r10 <- r10 >> 1
ADDI r6  r6  1      # r6 += 1
BLT  r16 r6  -3     # loop if r6 < 32
```

3*32 cycles!

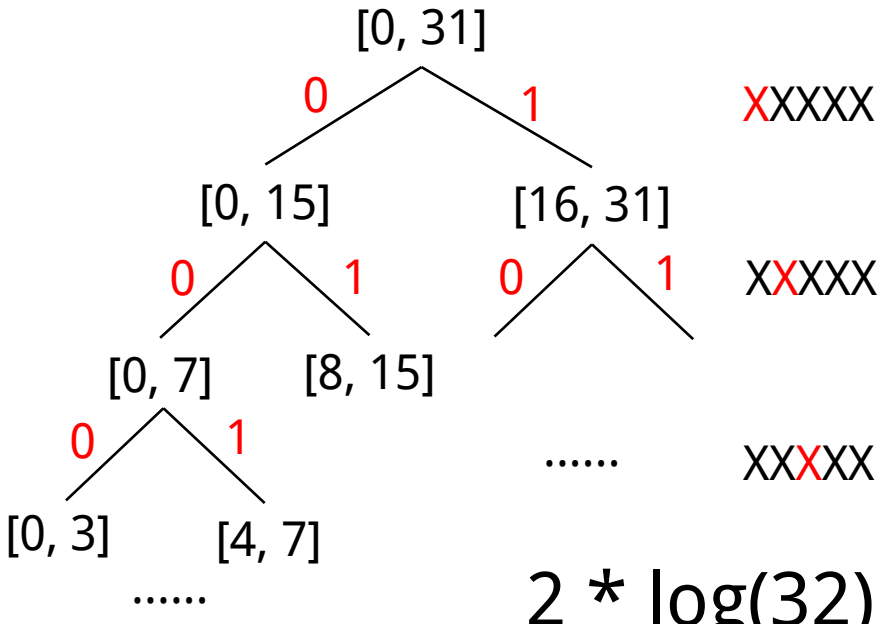
shift by 1 iteratively

```
AND  r6  r6  r26    # r26=0x1f get lower 5 bits
ORI  r7  r7  31     # r7 <- 31
BNE  r6  r7  4      # if r6 != 31, check next value
SHL  r10 r8  31     # if r6 == 31, do shift and comb
SHR  r11 r8  1
ORI  r8  r10 r11
JMP  END            # end of rotation
ORI  r7  r7  30     # r7 <- 30
BNE  r6  r7  4      # if r6 != 30, check next value
SHL  r10 r8  30     # if r6 == 30, do shift and comb
SHR  r11 r8  2
ORI  r8  r10 r11
JMP  END
......
```

2*16+4 cycles in average

shift only once using sequential comparison

# Rotation optimization




XXXXX

XXXXX

XXXXX

```
AND r12 r24 r6
BEQ r12 r24 ROT16_KEY   # 1xxxx, goto [16, 31]
ROT0_KEY:               # 0xxxx [0, 15]
AND r12 r23 r6
BEQ r12 r23 ROT8_KEY    # 01xxx, goto [8, 15]
AND r12 r22 r6
BEQ r12 r22 ROT4_KEY    # 001xx, goto [4, 7]
AND r12 r21 r6
BEQ r12 r21 ROT2_KEY    # 0001x, goto [2, 3]
AND r12 r20 r6
BEQ r12 r20 ROT1_KEY    # 00001, goto [1]
```

2 * log(32) + 4 = 14 cycles/rotation !

| Method | Average cycles | lines of code |
|---|---|---|
| Iteration | 96 | 10 |
| Sequential Comp. | 36 | 192 |
| Binary Search | 14 | 157 |

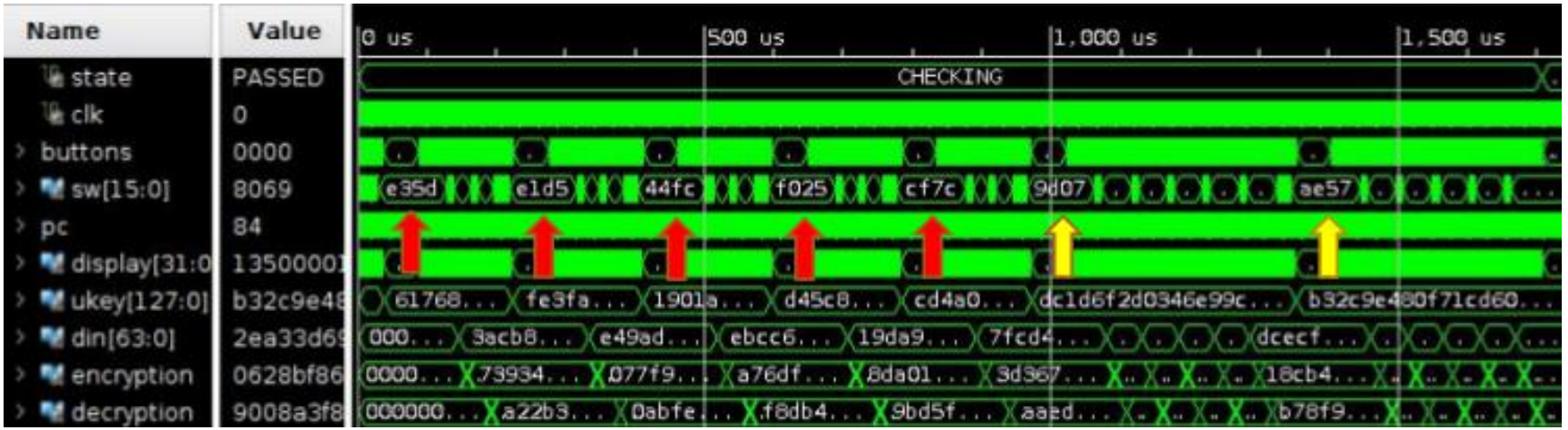
# Cycles for skey enc dec

# Cycles for skey enc dec

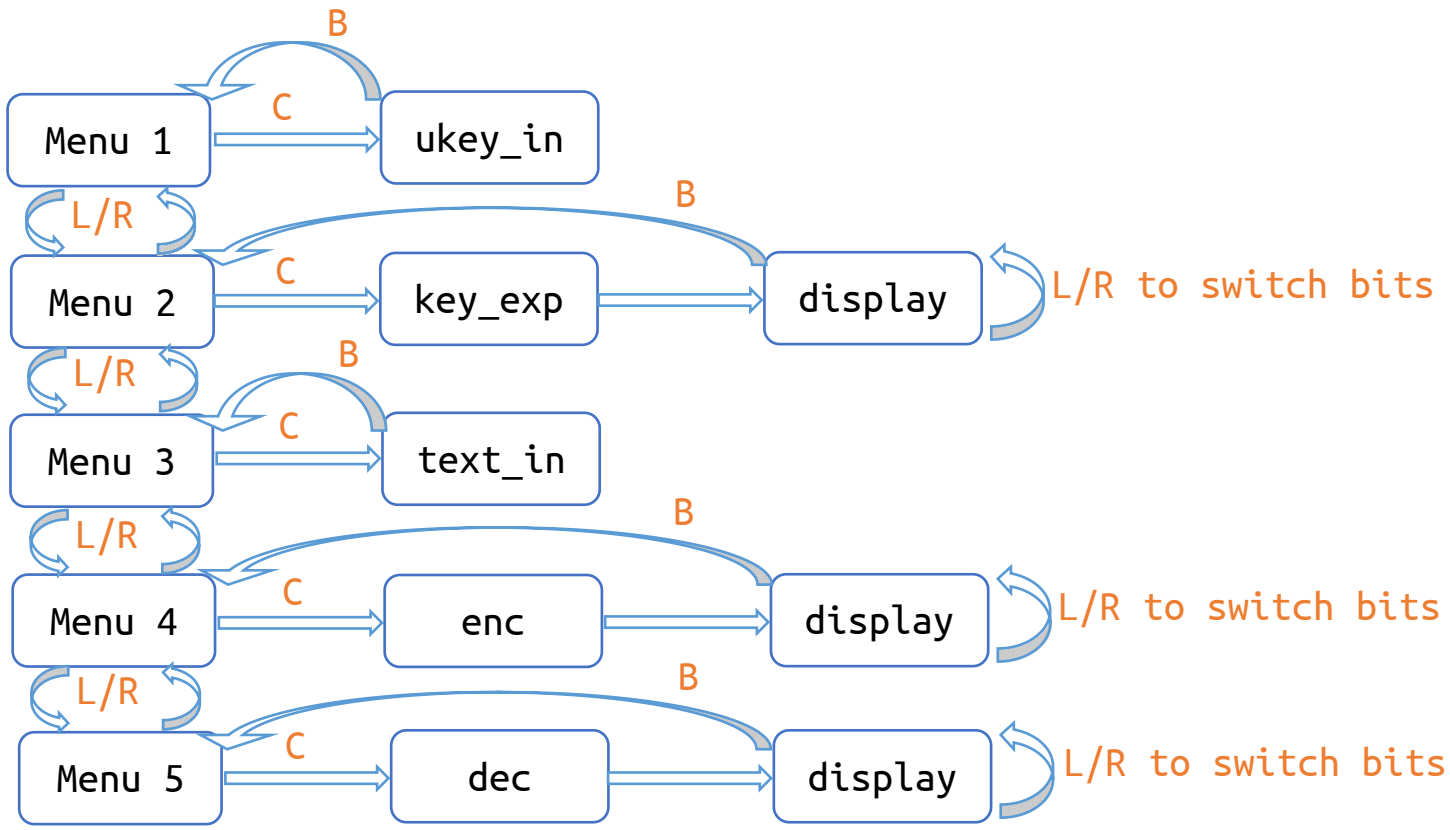
| function | Start at | End at | Cycles |
|---|---|---|---|
| key expansion | 358475 | 384215 | 2574 |
| encryption | 408515 | 413645 | 513 |
| decryption | 424125 | 429255 | 513 |

# Fuctional Simulation

# Timing simulation



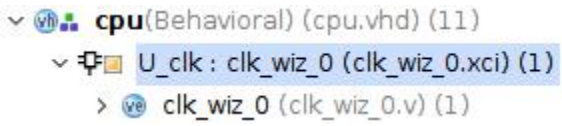
# RC5 program FSM



# Clock configuration

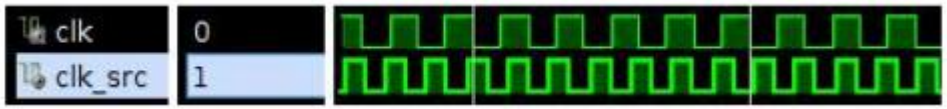


| Output Clock | Port Name | Output Freq (MHz) | |
|---|---|---|---|
| | | Requested | Actual |
| ☑ clk_out1 | clk_out1 | 135 | 135.000 |




```
component clk_wiz_0 is
    port (
        clk_out1: out std_logic;
        resetn: in std_logic;
        clk_in1: in std_logic);
end component;
```

```
U_clk : clk_wiz_0                    -- MMCM module
    port map (
        clk_out1 => clk_src,         -- out: 135 MHz
        resetn => cpu_rst,           -- active-low
        clk_in1 => clk);             -- in:  100 MHz
```

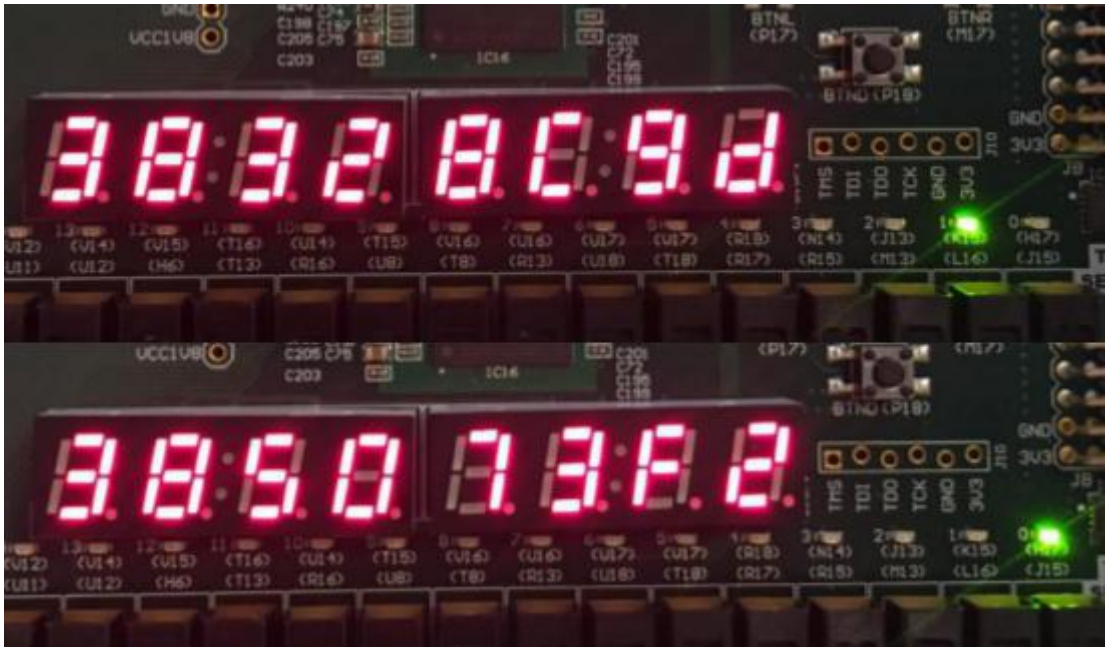



100MHz => 135MHz

ukey:
**12341111**
**0000abcd**
**11001111**
**33001111**

Encryption:
**9ebcf818**
**e2d82833**

din:
**cc0000cc**
**0000ee88**



Decryption:
**38328c9d**
**385073f2**