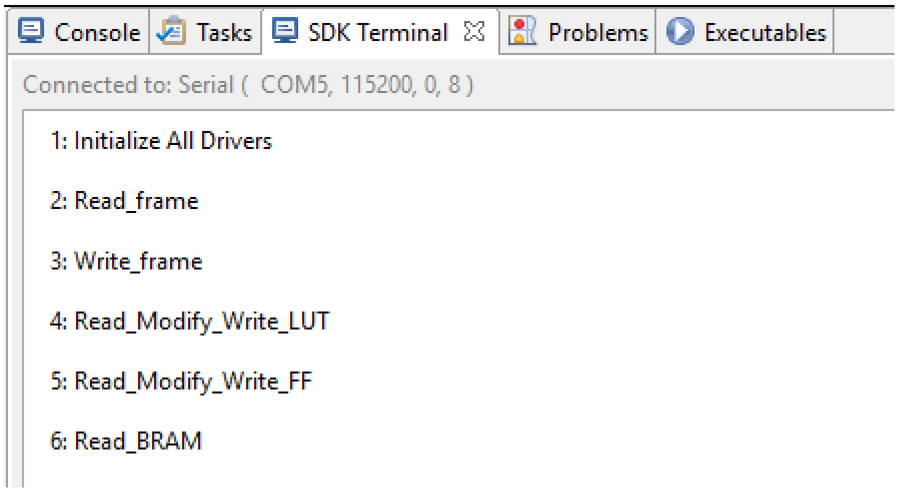
User Guide

VR-ZyCAP is a Zynq based reconfiguration controller, that allows resource-level (LUT & FF) run time reconfiguration. It performs five operations; Read frame, Write frame, DPR_LUT, DPR_FF and Read Memory. Input to these operations are transferred from Zynq processing System (PS) and the output can be shown on Integrated Logic Analyzer (ILA) or Zedboard.

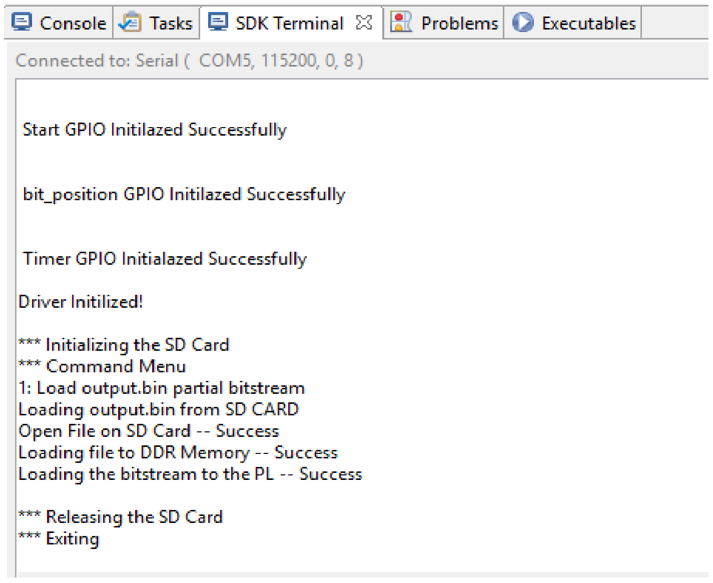
We will start with running RMW.c code in Xilinx SDK, which performs any of the above mentioned operation based on the OptionNext value. SDK terminal in Xilinx SDK is used to pass the input parameters.

First, after running RMW.c code, it will print menu to select OptionNext for different operations to be performed as shown in figure below;



```
Connected to: Serial ( COM5, 115200, 0, 8 )

1: Initialize All Drivers

2: Read_frame

3: Write_frame

4: Read_Modify_Write_LUT

5: Read_Modify_Write_FF

6: Read_BRAM
```

Drivers will be initialized first, so when program will ask to enter the OptionNext, we will enter 1, which will perform driver initialization and RAR configuration. Following messages will appear in SDK terminal after successful driver initialization and RAR_configuration.



```
Connected to: Serial ( COM5, 115200, 0, 8 )


 Start GPIO Initilazed Successfully


 bit_position GPIO Initilazed Successfully


 Timer GPIO Initialazed Successfully

Driver Initilized!

*** Initializing the SD Card
*** Command Menu
1: Load output.bin partial bitstream
Loading output.bin from SD CARD
Open File on SD Card -- Success
Loading file to DDR Memory -- Success
Loading the bitstream to the PL -- Success

*** Releasing the SD Card
*** Exiting
```

Then we can perform different operations;

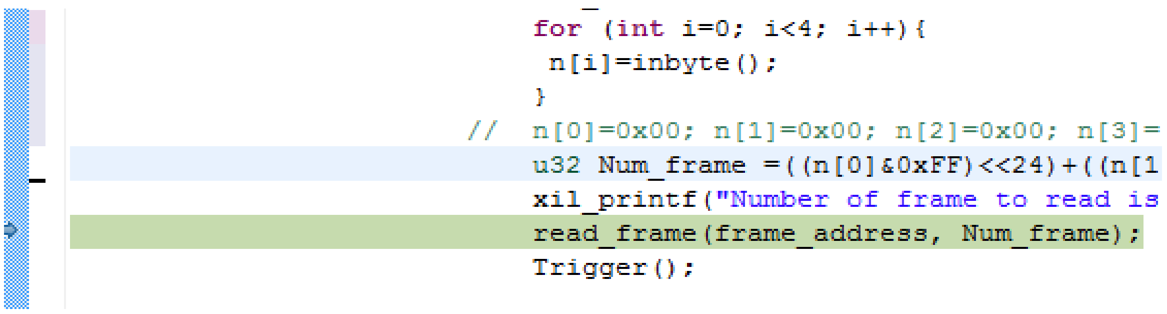
1. First we will start with the demonstration of Read frame operation, OptionNext = 2

Read frame operation require three inputs; Op_Sel, frame address and Num_of frames to read.

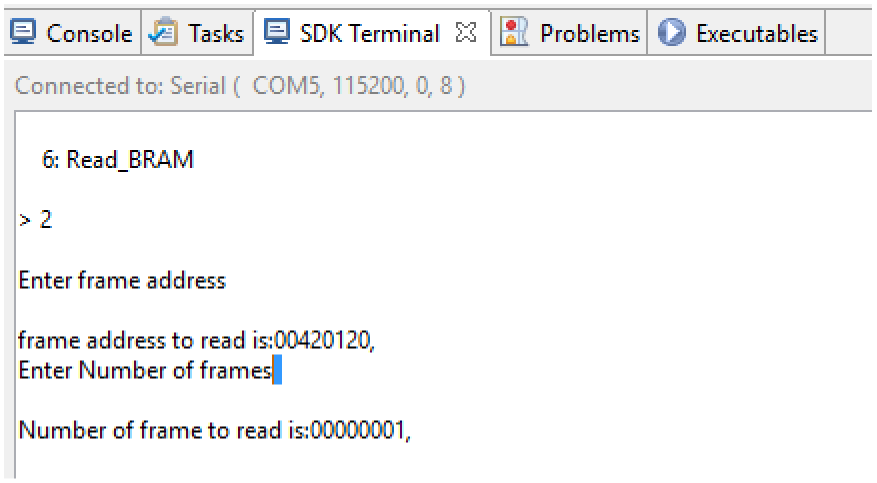
Read_frame(frame_address, Num_frame) function performs parameter transfer to VR-ZyCAP.

As Op_Sel is fixed for all operations so there is no need to input Op_Sel value. We hardcore this option in Read frame function. Other parameters are required to read through SDK terminal depending on which part of the device require reading.

So as an example, we placed LUT in X0Y0 Slice location in constraint file. So corresponding frame address (0x00420120) is generated and input to Read_frame function. Similarly, Num_frame = 4 is entered for reading LUT. For flip flop, Num_frame will be 1. So following input parameters are transferred through SDK terminal using inbyte() function.



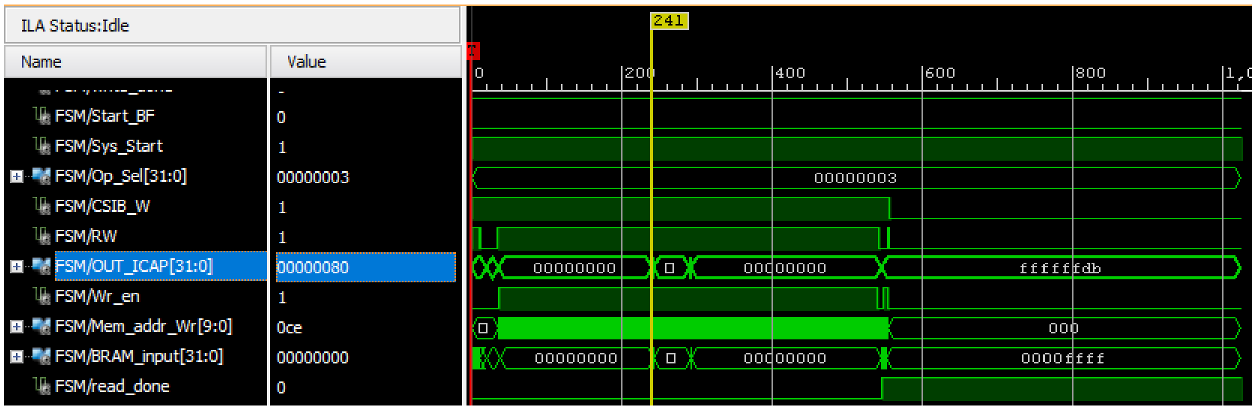
```
            for (int i=0; i<4; i++){
             n[i]=inbyte();
            }
        //  n[0]=0x00; n[1]=0x00; n[2]=0x00; n[3]=
            u32 Num_frame =((n[0]&0xFF)<<24)+((n[1
            xil_printf("Number of frame to read is
            read_frame(frame_address, Num_frame);
            Trigger();
```



```
Connected to: Serial ( COM5, 115200, 0, 8 )

  6: Read_BRAM

> 2

Enter frame address

frame address to read is:00420120,
Enter Number of frames

Number of frame to read is:00000001,
```

Trigger() function is important before start reading frames, because if there are any garbage values in BRAM of ICAP controller , that should be erased and every signal should be at its initialized state.

As our VR-ZyCAP is using ICAP for reading/writing to configuration memory, and controller is hardware based implementation in Programmable Logic (PL) section. So we can analyze Read frame at OUT_ICAP signal using ILA. ICAP primitive signals are important to mention, CSIB is the enable signal, RWB is Read or Write signal for ICAP, I is the input data to ICAP primitive and OUT_ICAP is the data output from ICAP. So All these signals can be seen on ILA in following figure which determine that frame reading is appropriately being performed.


ILA Status:Idle

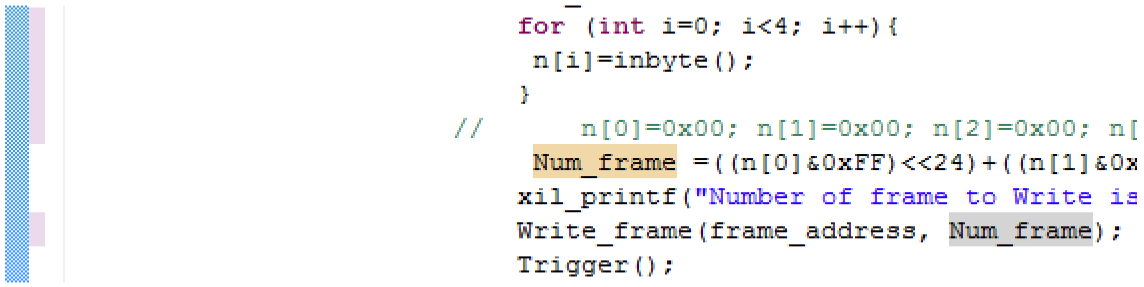
Name	Value
FSM/Start_BF	0
FSM/Sys_Start	1
FSM/Op_Sel[31:0]	00000003
FSM/CSIB_W	1
FSM/RW	1
FSM/OUT_ICAP[31:0]	00000080
FSM/Wr_en	1
FSM/Mem_addr_Wr[9:0]	0ce
FSM/BRAM_input[31:0]	00000000
FSM/read_done	0

Figure 1: Read frame Signals

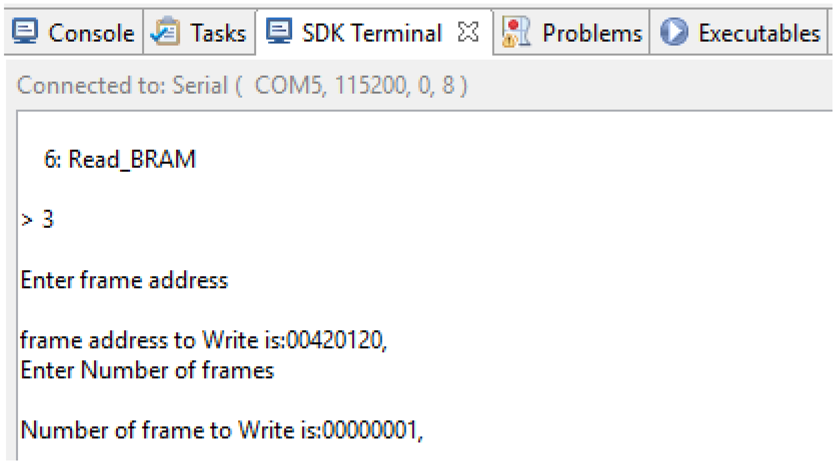
As we have implemented LUT as AND logic, so when will read data, initialization value of AND logic (0x8000000000000000) will be read in byte swap manner on OUT_ICAP signal. Which is being written to BRAM at Mem_addr_Wr signals. ICAP enable signal (CSIB) can also be seen which will be enabled until read process finishes. Also RW signal will be 1 during reading frame data from configuration memory otherwise it will be 1 while sending commands to configuration memory.

2. To perform Write frame Operation, OptionNext will be selected as OptionNext = 3.

Write frame operation require three inputs; Op_Sel is 2, which is hardcore. Other inputs are frame address to write and number of frames. Which will be entered using SDK terminal.

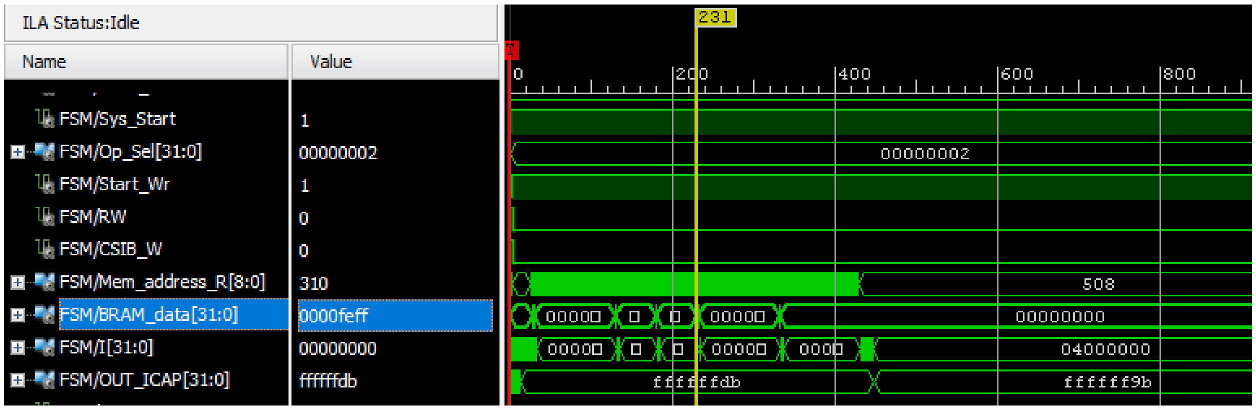

```
           for (int i=0; i<4; i++){
            n[i]=inbyte();
           }
       //       n[0]=0x00; n[1]=0x00; n[2]=0x00; n[
            Num_frame =((n[0]&0xFF)<<24)+((n[1]&0x
           xil_printf("Number of frame to Write is
           Write_frame(frame_address, Num_frame);
           Trigger();
```

As an example; LUT is placed in X0Y0 location, so corresponding frame_address (0x00420120) and number of frames are entered.



```
Connected to: Serial ( COM5, 115200, 0, 8 )

   6: Read_BRAM

> 3

Enter frame address

frame address to Write is:00420120,
Enter Number of frames

Number of frame to Write is:00000001,
```

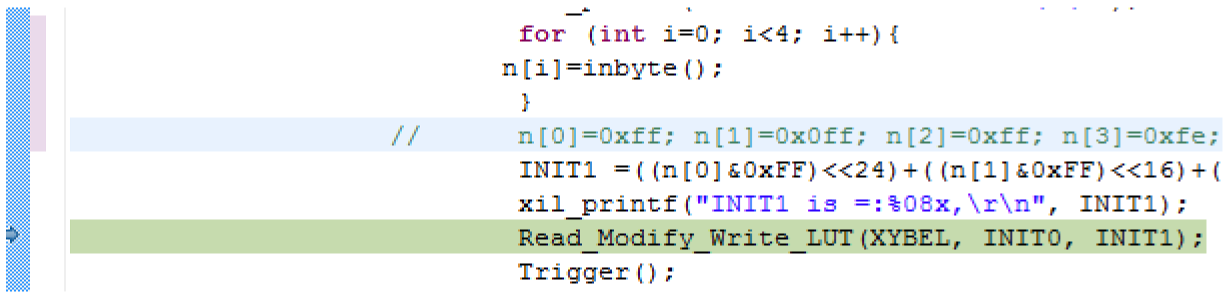
The data stored in BRAM will be written to configuration memory using this operation. And change in logic can also be observed on Zedboard. Signal I in following picture shows complete data frame writing. As if we have to change logic of LUT read in read frame operation. OR logic will be performed by writing initialization value (0xfffffffffffffffe) into BRAM. From which data will be read using Mem_address_R signal and sent to configuration memory using I port of ICAP. OUT_ICAP should be shown as constant 0xffffffdb signal. RW will remain 0 during whole reading process.



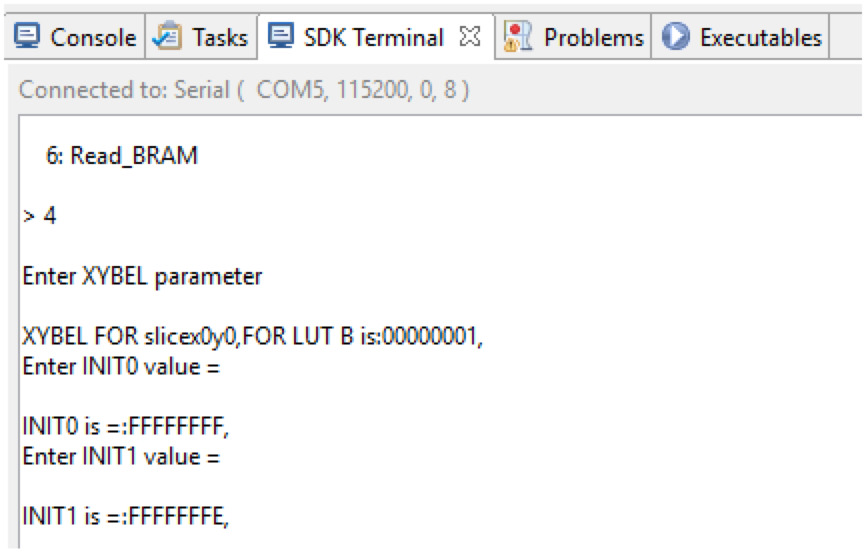
```
ILA Status:Idle

Name                      Value
FSM/Sys_Start             1
FSM/Op_Sel[31:0]          00000002
FSM/Start_Wr              1
FSM/RW                    0
FSM/CSIB_W                0
FSM/Mem_address_R[8:0]    310
FSM/BRAM_data[31:0]       0000feff
FSM/I[31:0]               00000000
FSM/OUT_ICAP[31:0]        ffffffdb
```

3. To perform Read Modify Write LUT operation, OptionNext is selected as 4.

We require three inputs; Op_Sel = 3, XYBEL parameter and Initialization (INIT) value. For LUT B in slice X0Y0, XYBEL parameter is generated as 0x00000001, and Num_frames must be 4 which is hardcore in this operation. Because 4 frames are required to modify one LUT. If we are going to change the LUT logic from AND to OR, then 64-bit initialization value is generated as (0xfffffffffffffffe). As one GPIO can be configured as maximum 32 bits for input or output. So we used two GPIOs to transfer 64-bit initialization value.

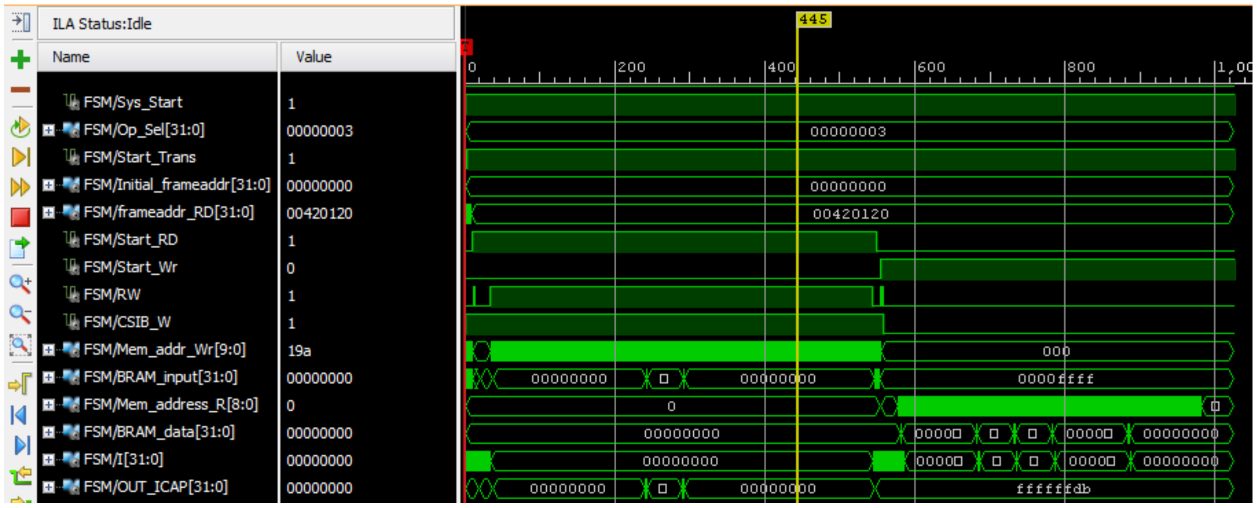


```
         for (int i=0; i<4; i++){
        n[i]=inbyte();
         }
//       n[0]=0xff; n[1]=0x0ff; n[2]=0xff; n[3]=0xfe;
         INIT1 =((n[0]&0xFF)<<24)+((n[1]&0xFF)<<16)+(
         xil_printf("INIT1 is =:%08x,\r\n", INIT1);
         Read_Modify_Write_LUT(XYBEL, INIT0, INIT1);
         Trigger();
```



```
Connected to: Serial ( COM5, 115200, 0, 8 )

  6: Read_BRAM

> 4

Enter XYBEL parameter

XYBEL FOR slicex0y0,FOR LUT B is:00000001,
Enter INIT0 value =

INIT0 is =:FFFFFFFF,
Enter INIT1 value =

INIT1 is =:FFFFFFFE,
```

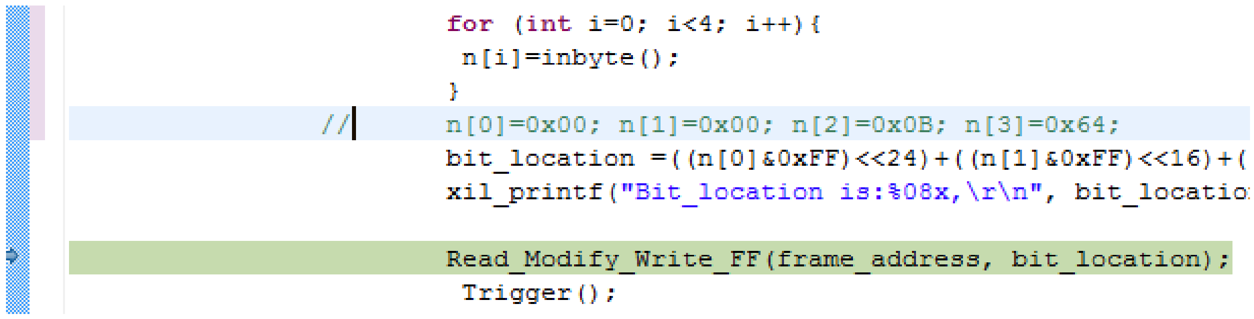
The modification in LUT logic can be clearly observed on Zedboard using leds and Dip Switches. Also the complete read modify and write operation can be seen in following logic analyzer. After Sys_Start signal, translation module (SLICE2FAR) will start that generates frameaddr_RD signal. So reading will start, and read data (0x8000000000000000) can be seen in OUT_ICAP signal. So after modification, frame writing will start that will write the modified data (0xfffffffffffffffe) in configuration memory at four frame addresses.



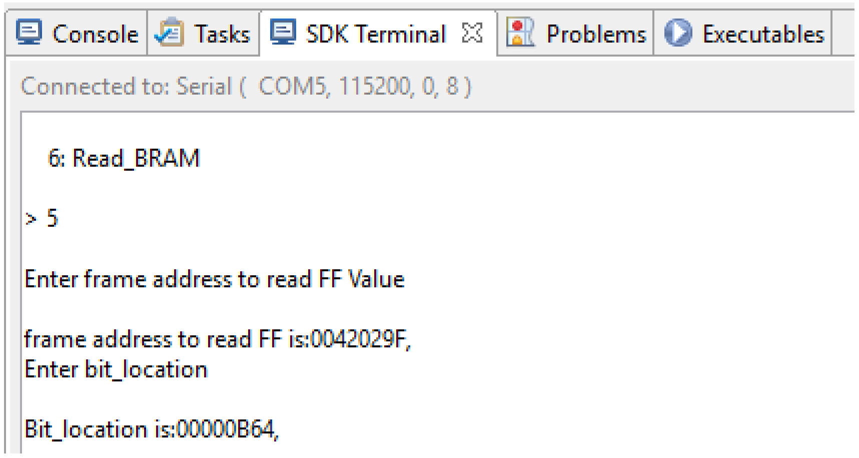
Name	Value
FSM/Sys_Start	1
FSM/Op_Sel[31:0]	00000003
FSM/Start_Trans	1
FSM/Initial_frameaddr[31:0]	00000000
FSM/frameaddr_RD[31:0]	00420120
FSM/Start_RD	1
FSM/Start_Wr	0
FSM/RW	1
FSM/CSIB_W	1
FSM/Mem_addr_Wr[9:0]	19a
FSM/BRAM_input[31:0]	00000000
FSM/Mem_address_R[8:0]	0
FSM/BRAM_data[31:0]	00000000
FSM/I[31:0]	00000000
FSM/OUT_ICAP[31:0]	00000000

4. To perform Read Modify Write flip flop operation, OptionNext is selected as 5.

We require three input parameters, Op_Sel, frame address and bit location within the frame. Op_Sel is 4 for this operation. Corresponding frame address for flip-flop placed in P-block region is 0x0042029f. Nume of frame is 1 for flip-flop bits, so it has kept fixed in architecture. Bit location is determined by the logic location file (ll) for partial reconfigurable region. Which is 2916. So in hex format, bit location is entered as 0x00000B64.



```
                        for (int i=0; i<4; i++){
                         n[i]=inbyte();
                        }
                //      n[0]=0x00; n[1]=0x00; n[2]=0x0B; n[3]=0x64;
                        bit_location =((n[0]&0xFF)<<24)+((n[1]&0xFF)<<16)+(
                        xil_printf("Bit_location is:%08x,\r\n", bit_locatio

                        Read_Modify_Write_FF(frame_address, bit_location);
                         Trigger();
```

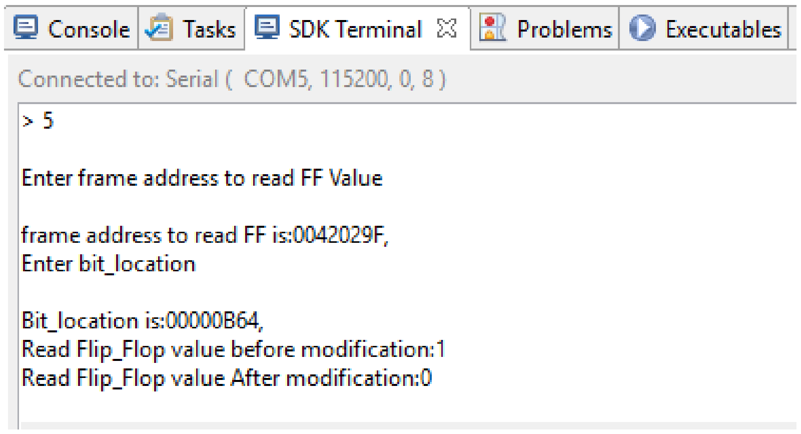


```
Connected to: Serial ( COM5, 115200, 0, 8 )

  6: Read_BRAM

> 5

Enter frame address to read FF Value

frame address to read FF is:0042029F,
Enter bit_location

Bit_location is:00000B64,
```

We can see the read and write from for flip flop in logic Analyzer in following picture. Word position is calculated as 94 so frame word 195(dummy frame + Word_position) is read which is then flipped and modified and written back to configuration memory.

Flip-Flop output can't be seen on board because of the high frequency. Also StartUp primitive doesn't allow ILA to display the results.

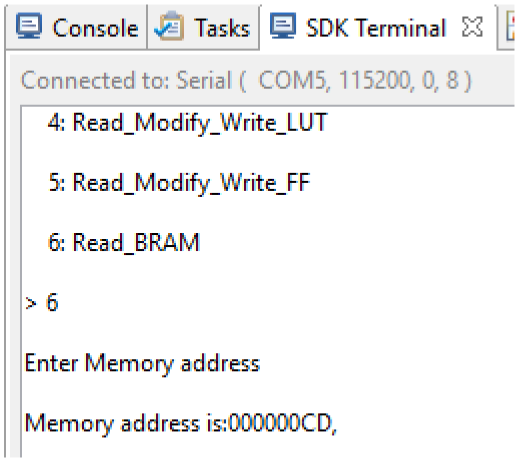
So we connected GPIO to Flip-Flop output which gives the modified value of flip flop.



```
Connected to: Serial ( COM5, 115200, 0, 8 )
> 5

Enter frame address to read FF Value

frame address to read FF is:0042029F,
Enter bit_location

Bit_location is:00000B64,
Read Flip_Flop value before modification:1
Read Flip_Flop value After modification:0
```

5. To perform Read BRAM operation, OptionNext is selected as 6.

Read BRAM operation just require two input parameter, Op_Sel and Mem_addr. Op_Sel for this operation is 5. This operation is only required for debugging. So we can enter any memory address to check the content of frames read through configuration memory. In following picture, Mem_addr is selected as 205 (0x000000CD). Which we can see, contains frame data for Ist frame. But this operation should be used after read frame operation.



```
Connected to: Serial ( COM5, 115200, 0, 8 )
   4: Read_Modify_Write_LUT

   5: Read_Modify_Write_FF

   6: Read_BRAM

> 6

Enter Memory address

Memory address is:000000CD,
```

For any Query: Please contact **bushrasultana21@live.com**