SOC Design
Lab4-1 Execute Code in User Memory

112061611 陳伯丞, 112061524 葉又菘, 110063553 張傑閔

Description:

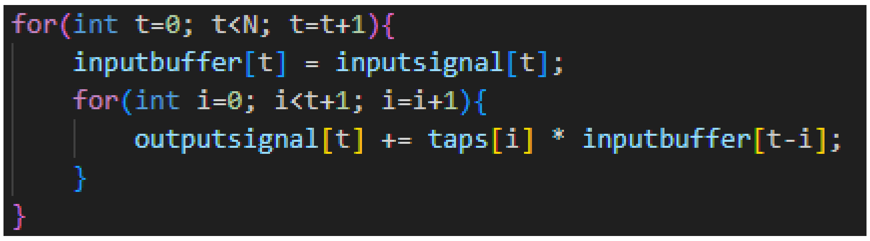
In this lab4-1, we design the firmware code in user_proj.v to manage the interface between user BRAM and wishbone. And design with FIR engine with c language, which is in fir.c. The entire flow of this lab4-1 is (1) the FIR engine compile into RISCV code (2) the user project get the FIR engine output through wishbone bus (3) the user project store the result into BRAM.

The following sections is the implementation detail and waveform.

1. Explanation of your firmware code

1.1. How does it execute a multiplication in assembly code.

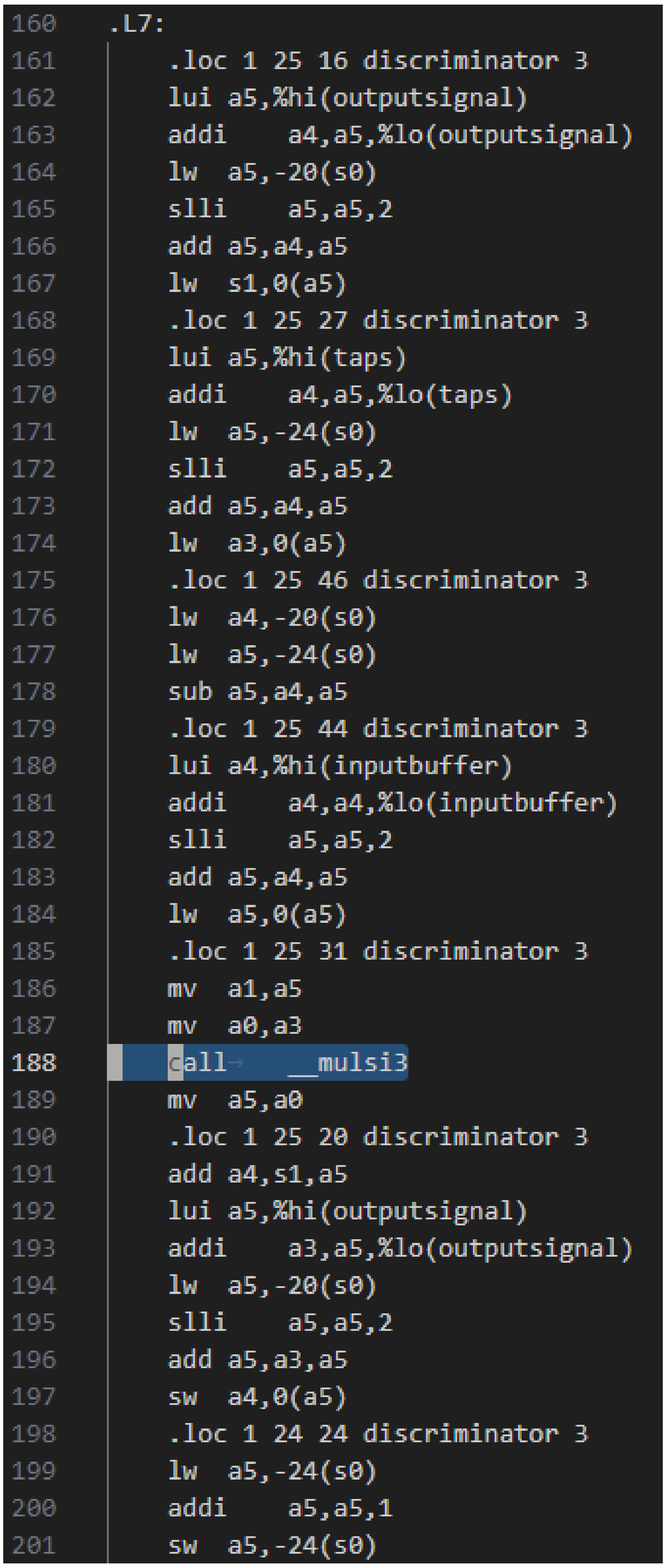
The following code is the FIR engine in fir.c. It computes the FIR result with respect to the convolution function.



```
for(int t=0; t<N; t=t+1){
    inputbuffer[t] = inputsignal[t];
    for(int i=0; i<t+1; i=i+1){
        outputsignal[t] += taps[i] * inputbuffer[t-i];
    }
}
```

Figure. Code for fir.c

And the following figure shows the multiplication execution in RISCV code. The __mulsi3 is the multiplication function call in assembly code. And the whole L7 part is the FIR engine function. This screenshot is from elf-fir.s.

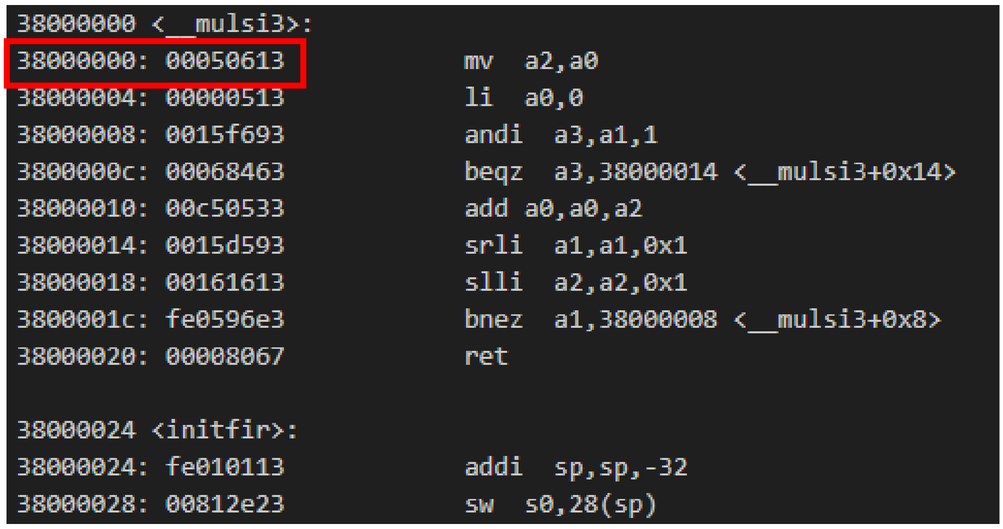


```
.L7:
	.loc 1 25 16 discriminator 3
	lui a5,%hi(outputsignal)
	addi	a4,a5,%lo(outputsignal)
	lw  a5,-20(s0)
	slli	a5,a5,2
	add a5,a4,a5
	lw  s1,0(a5)
	.loc 1 25 27 discriminator 3
	lui a5,%hi(taps)
	addi	a4,a5,%lo(taps)
	lw  a5,-24(s0)
	slli	a5,a5,2
	add a5,a4,a5
	lw  a3,0(a5)
	.loc 1 25 46 discriminator 3
	lw  a4,-20(s0)
	lw  a5,-24(s0)
	sub a5,a4,a5
	.loc 1 25 44 discriminator 3
	lui a4,%hi(inputbuffer)
	addi	a4,a4,%lo(inputbuffer)
	slli	a5,a5,2
	add a5,a4,a5
	lw  a5,0(a5)
	.loc 1 25 31 discriminator 3
	mv  a1,a5
	mv  a0,a3
	call	__mulsi3
	mv  a5,a0
	.loc 1 25 20 discriminator 3
	add a4,s1,a5
	lui a5,%hi(outputsignal)
	addi	a3,a5,%lo(outputsignal)
	lw  a5,-20(s0)
	slli	a5,a5,2
	add a5,a3,a5
	sw  a4,0(a5)
	.loc 1 24 24 discriminator 3
	lw  a5,-24(s0)
	addi	a5,a5,1
	sw  a5,-24(s0)
```

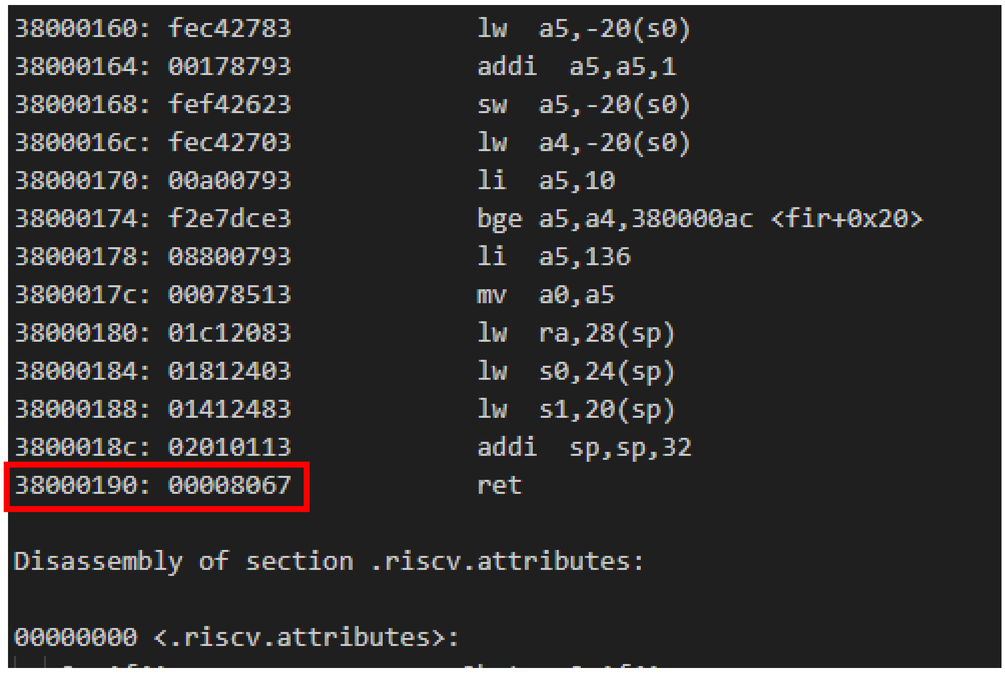
Fig. Assembly code

1.2. What address allocate for user project and how many space is required to allocate to firmware.

According to the screenshot, user project is allocated in 0x38000000~0x38000190. The space requirement of firmware code is 2448 Bytes.



```
38000000 <__mulsi3>:
38000000: 00050613          mv   a2,a0
38000004: 00000513          li   a0,0
38000008: 0015f693          andi  a3,a1,1
3800000c: 00068463          beqz  a3,38000014 <__mulsi3+0x14>
38000010: 00c50533          add a0,a0,a2
38000014: 0015d593          srli  a1,a1,0x1
38000018: 00161613          slli  a2,a2,0x1
3800001c: fe0596e3          bnez  a1,38000008 <__mulsi3+0x8>
38000020: 00008067          ret

38000024 <initfir>:
38000024: fe010113          addi  sp,sp,-32
38000028: 00812e23          sw   s0,28(sp)
```

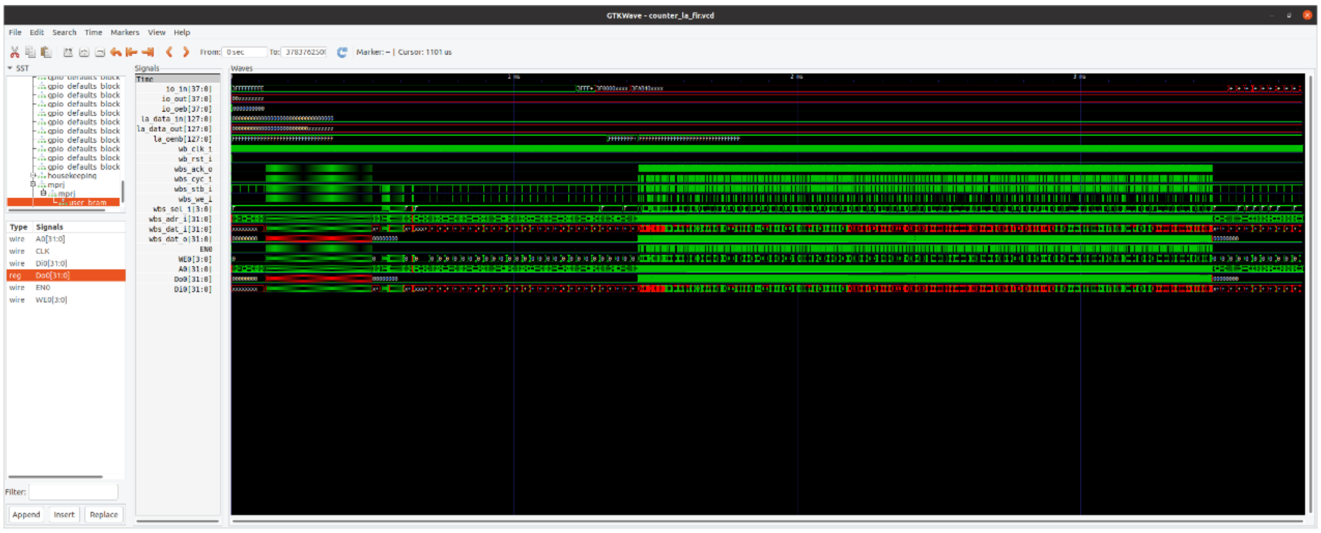


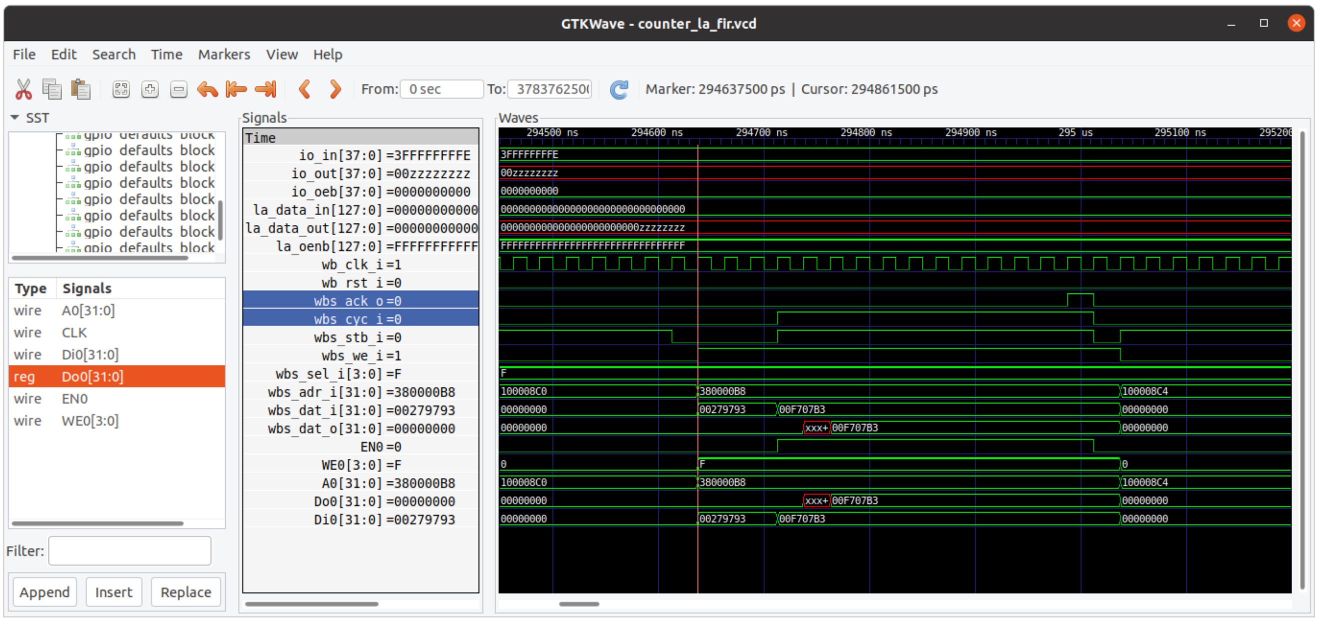
```
38000160: fec42783          lw   a5,-20(s0)
38000164: 00178793          addi  a5,a5,1
38000168: fef42623          sw   a5,-20(s0)
3800016c: fec42703          lw   a4,-20(s0)
38000170: 00a00793          li   a5,10
38000174: f2e7dce3          bge a5,a4,380000ac <fir+0x20>
38000178: 08800793          li   a5,136
3800017c: 00078513          mv   a0,a5
38000180: 01c12083          lw   ra,28(sp)
38000184: 01812403          lw   s0,24(sp)
38000188: 01412483          lw   s1,20(sp)
3800018c: 02010113          addi  sp,sp,32
38000190: 00008067          ret

Disassembly of section .riscv.attributes:

00000000 <.riscv.attributes>:
```

Fig. user project allocation

2. Interface between BRAM and wishbone

2.1 Waveform from xsim





GTKWave - counter_la_fir.vcd

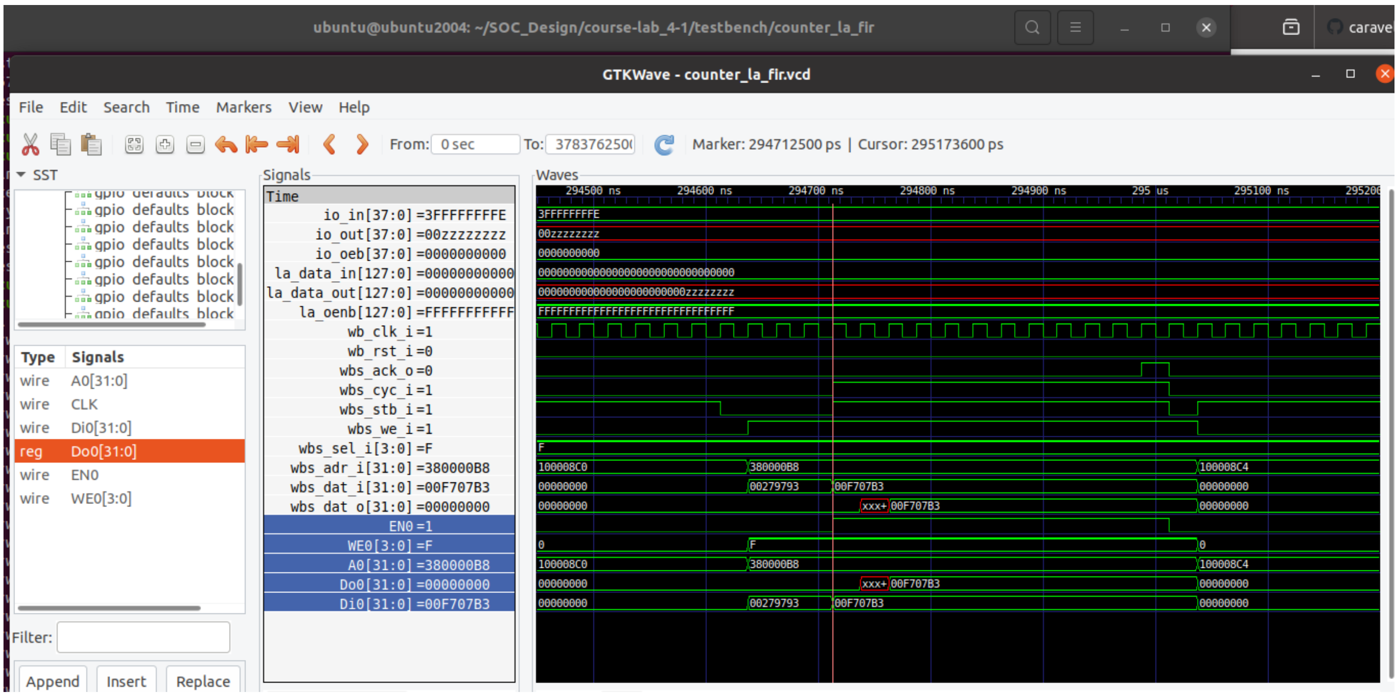
Marker: 294637500 ps | Cursor: 294861500 ps

```
Time
   io_in[37:0] =3FFFFFFFFE
  io_out[37:0] =00zzzzzzzz
  io_oeb[37:0] =0000000000
la_data_in[127:0] =00000000000
la_data_out[127:0] =00000000000
 la_oenb[127:0] =FFFFFFFFFFF
     wb_clk_i =1
     wb_rst_i =0
    wbs_ack_o =0
    wbs_cyc_i =0
    wbs_stb_i =0
     wbs_we_i =1
  wbs_sel_i[3:0] =F
 wbs_adr_i[31:0] =380000B8
 wbs_dat_i[31:0] =00279793
 wbs_dat_o[31:0] =00000000
          EN0 =0
     WE0[3:0] =F
     A0[31:0] =380000B8
    Do0[31:0] =00000000
    Di0[31:0] =00279793
```

Type	Signals
wire	A0[31:0]
wire	CLK
wire	Di0[31:0]
reg	Do0[31:0]
wire	EN0
wire	WE0[3:0]

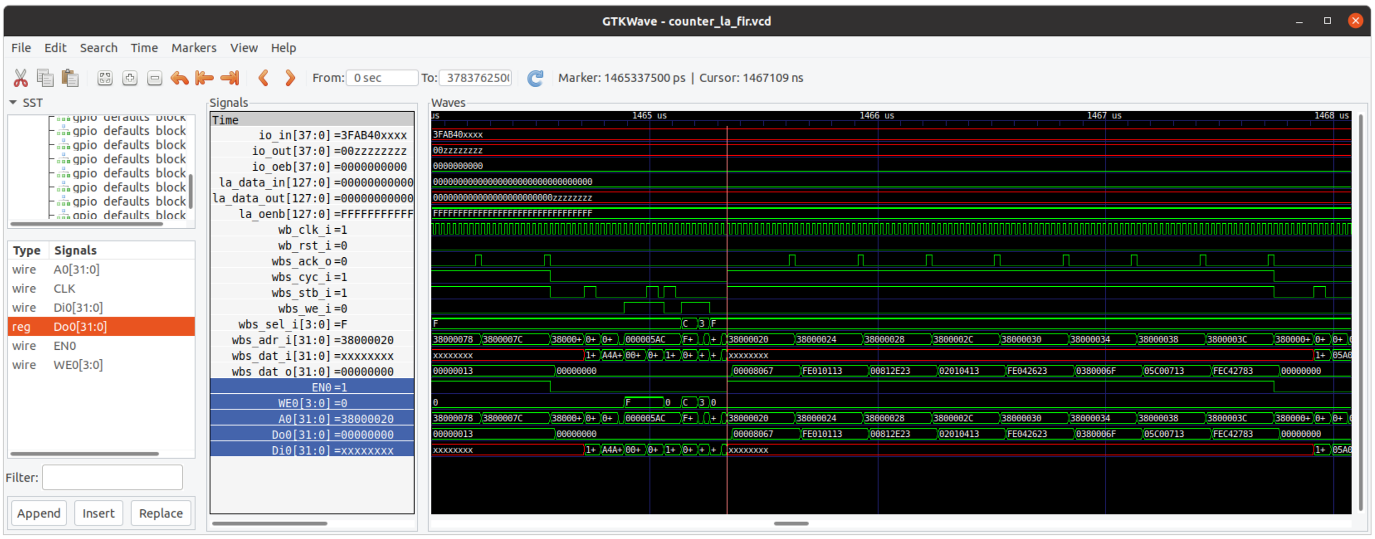
Delay between 'wbs_ack_o' and 'posedge of wbs_cyc_i' is 10 clock cycle.

BRAM write:



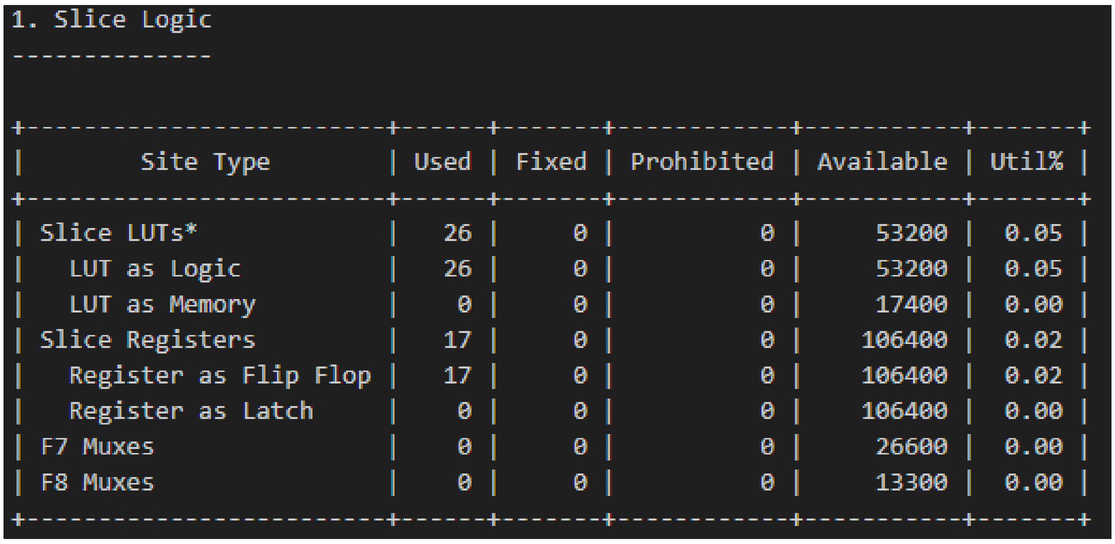
```
ubuntu@ubuntu2004: ~/SOC_Design/course-lab_4-1/testbench/counter_la_fir
GTKWave - counter_la_fir.vcd
From: 0 sec  To: 3783762500  Marker: 294712500 ps | Cursor: 295173600 ps

Type  Signals
wire  A0[31:0]
wire  CLK
wire  Di0[31:0]
reg   Do0[31:0]
wire  EN0
wire  WE0[3:0]

Time
      io_in[37:0] =3FFFFFFFFE
     io_out[37:0] =00zzzzzzzz
     io_oeb[37:0] =0000000000
 la_data_in[127:0] =00000000000
la_data_out[127:0] =00000000000
    la_oenb[127:0] =FFFFFFFFFFF
          wb_clk_i =1
          wb_rst_i =0
         wbs_ack_o =0
         wbs_cyc_i =1
         wbs_stb_i =1
          wbs_we_i =1
    wbs_sel_i[3:0] =F
   wbs_adr_i[31:0] =380000B8
   wbs_dat_i[31:0] =00F707B3
   wbs_dat_o[31:0] =00000000
               EN0 =1
          WE0[3:0] =F
          A0[31:0] =380000B8
         Do0[31:0] =00000000
         Di0[31:0] =00F707B3
```

BRAM read:

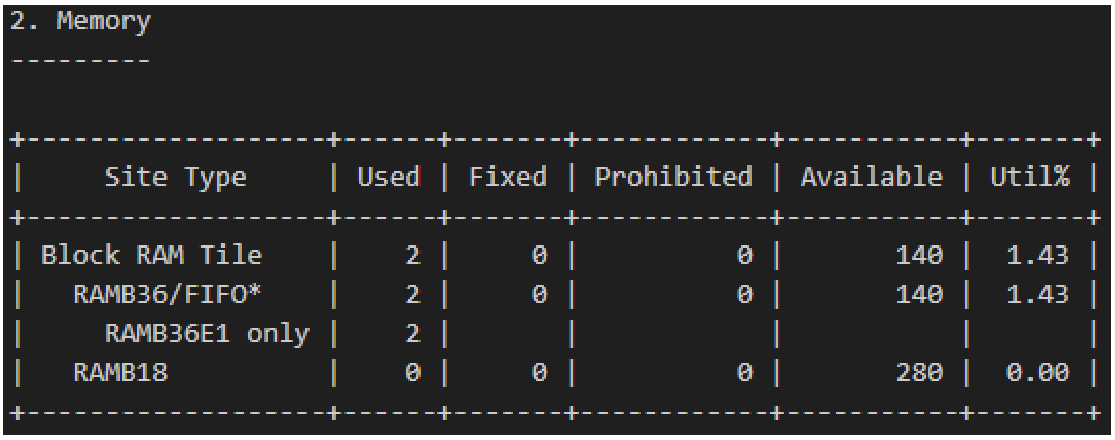


```
GTKWave - counter_la_fir.vcd
From: 0 sec  To: 3783762500  Marker: 1465337500 ps | Cursor: 1467109 ns

Type  Signals
wire  A0[31:0]
wire  CLK
wire  Di0[31:0]
reg   Do0[31:0]
wire  EN0
wire  WE0[3:0]

Time
      io_in[37:0] =3FAB40xxxx
     io_out[37:0] =00zzzzzzzz
     io_oeb[37:0] =0000000000
 la_data_in[127:0] =00000000000
la_data_out[127:0] =00000000000
    la_oenb[127:0] =FFFFFFFFFFF
          wb_clk_i =1
          wb_rst_i =0
         wbs_ack_o =0
         wbs_cyc_i =1
         wbs_stb_i =1
          wbs_we_i =0
    wbs_sel_i[3:0] =F
   wbs_adr_i[31:0] =38000020
   wbs_dat_i[31:0] =xxxxxxxx
   wbs_dat_o[31:0] =00000000
               EN0 =1
          WE0[3:0] =0
          A0[31:0] =38000020
         Do0[31:0] =00000000
         Di0[31:0] =xxxxxxxx
```

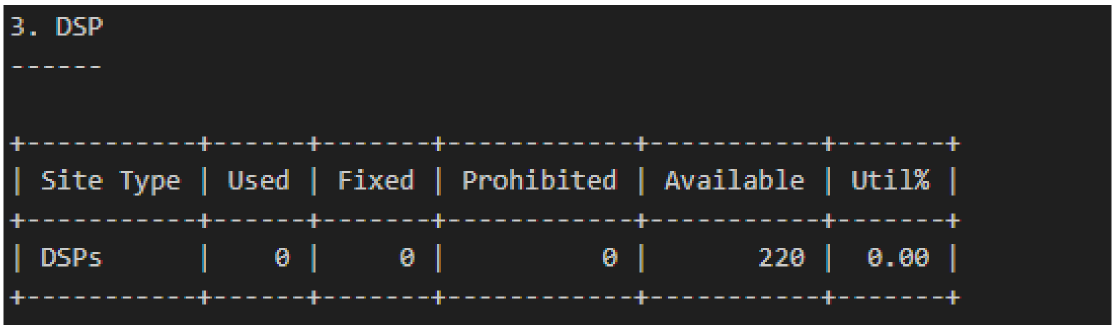
3. Synthesis report



```
1. Slice Logic
--------------

+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs*             |   26 |     0 |          0 |     53200 |  0.05 |
|   LUT as Logic          |   26 |     0 |          0 |     53200 |  0.05 |
|   LUT as Memory         |    0 |     0 |          0 |     17400 |  0.00 |
| Slice Registers         |   17 |     0 |          0 |    106400 |  0.02 |
|   Register as Flip Flop |   17 |     0 |          0 |    106400 |  0.02 |
|   Register as Latch     |    0 |     0 |          0 |    106400 |  0.00 |
| F7 Muxes                |    0 |     0 |          0 |     26600 |  0.00 |
| F8 Muxes                |    0 |     0 |          0 |     13300 |  0.00 |
+-------------------------+------+-------+------------+-----------+-------+
```



```
2. Memory
---------

+-------------------+------+-------+------------+-----------+-------+
|     Site Type     | Used | Fixed | Prohibited | Available | Util% |
+-------------------+------+-------+------------+-----------+-------+
| Block RAM Tile    |    2 |     0 |          0 |       140 |  1.43 |
|   RAMB36/FIFO*    |    2 |     0 |          0 |       140 |  1.43 |
|     RAMB36E1 only |    2 |       |            |           |       |
|   RAMB18          |    0 |     0 |          0 |       280 |  0.00 |
+-------------------+------+-------+------------+-----------+-------+
```



```
3. DSP
------

+-----------+------+-------+------------+-----------+-------+
| Site Type | Used | Fixed | Prohibited | Available | Util% |
+-----------+------+-------+------------+-----------+-------+
| DSPs      |    0 |     0 |          0 |       220 |  0.00 |
+-----------+------+-------+------------+-----------+-------+
```

4. GitHub Link:

https://github.com/ken01235/SOC_Design/tree/master/course-lab_4-1%20report