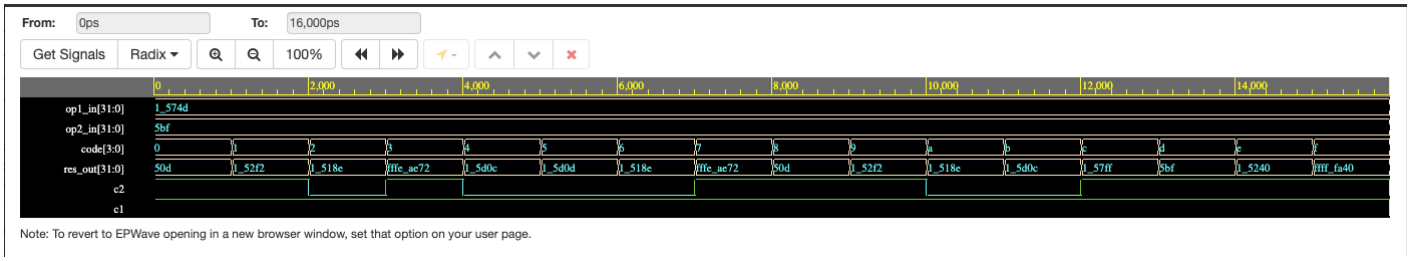
Stage 1
Divyanshu Pabia

The epwaves shown are the result of the testbenches that I have submitted along with my code in this zip file. In the memery parts, I have not used a 32-bit memory, instead, I have taken just what was required in the question(for example, 6 bits for 64). I will change it later in the subsequent parts.

Here are the 4 parts of stage 1:

1. ALU
   This program performs actions according to one of the 16 opcodes. In operations where carry is needed, I first append op1 and op2 with a '0', then save the result in a temporary 33 bit vector and the truncate the extra bit, which is returned in the form of a carry. In operations where carry bit was not required(like and or) I have returned carry_out = carry_in.

   Here is the epwave of my testbench:



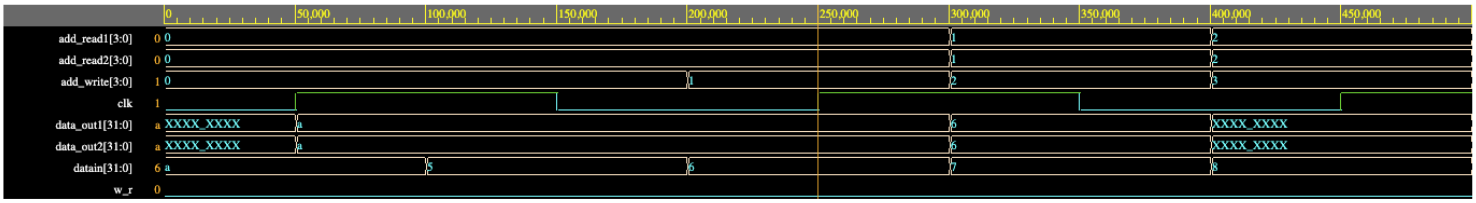
   Synthesis report –

```
Resource                          Used    Avail   Utilization
# Info: ------------------------------------------------------------
# Info: IOs                       102     210      48.57%
# Info: Global Buffers            0       32        0.00%
# Info: LUTs                      167     63400     0.26%
# Info: CLB Slices                40      15850     0.25%
# Info: Dffs or Latches           0       126800    0.00%
# Info: Block RAMs                0       135       0.00%
# Info: DSP48E1s                  0       240       0.00%


# Info: Library: work    Cell: alu    View: rtl
# Info: ***********************************************
# Info:  Number of ports :                          102
# Info:  Number of nets :                           408
# Info:  Number of instances :                      307
# Info:  Number of references to this view :          0
# Info: Total accumulated area :
```

```
# Info:  Number of LUTs :                          167
# Info:  Number of Primitive LUTs :                170
# Info:  Number of LUTs with LUTNM/HLUTNM :          6
# Info:  Number of MUX CARRYs :                     64
# Info:  Number of accumulated instances :         404
```

2. Register file

   In this, I have 2 read ports and 1 write port. Through data_out1 and 2, the contents of memory array at the 2 read locations are continuously being displayed. On the rising edge of the clock, the data taken from datain is written into the memory at the index specified by the write index. Here is the epwave of my code:



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

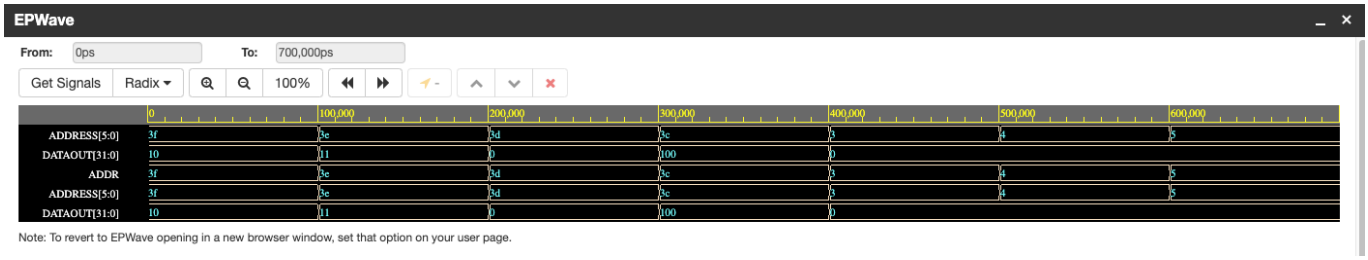
Here is the result of my synthesis:

```
# Info: Library: work    Cell: RF    View: BEV
# Info: ***********************************************
# Info:  Number of ports :                          110
# Info:  Number of nets :                           221
# Info:  Number of instances :                      112
# Info:  Number of references to this view :          0
# Info: Total accumulated area :
# Info:  Number of LUTs :                            49
# Info:  Number of Primitive LUTs :                  49
# Info:    Number of LUTs as Distributed RAM :       48
# Info:  Number of accumulated instances :          124


# Info: Device Utilization for 7A100TCSG324
# Info: ***********************************************************
# Info: Resource                          Used    Avail   Utilization
# Info: -----------------------------------------------------------
# Info: IOs                               110     210      52.38%
# Info: Global Buffers                    1       32        3.12%
# Info: LUTs                              49      63400     0.08%
# Info: CLB Slices                        13      15850     0.08%
# Info: Dffs or Latches                   0       126800    0.00%
```

```
# Info: Block RAMs                          0       135       0.00%
# Info: Distributed RAMs
# Info:    RAM32M                           10
# Info:    RAM64M                           2
# Info: DSP48E1s                            0       240       0.00%
```

3. Program Memory

In this, the memory is already filled with instructions, we can just read through it by inputting an address. This is a ROM. Here is the epwave for my testbench:



Here is the output of the synthesis:

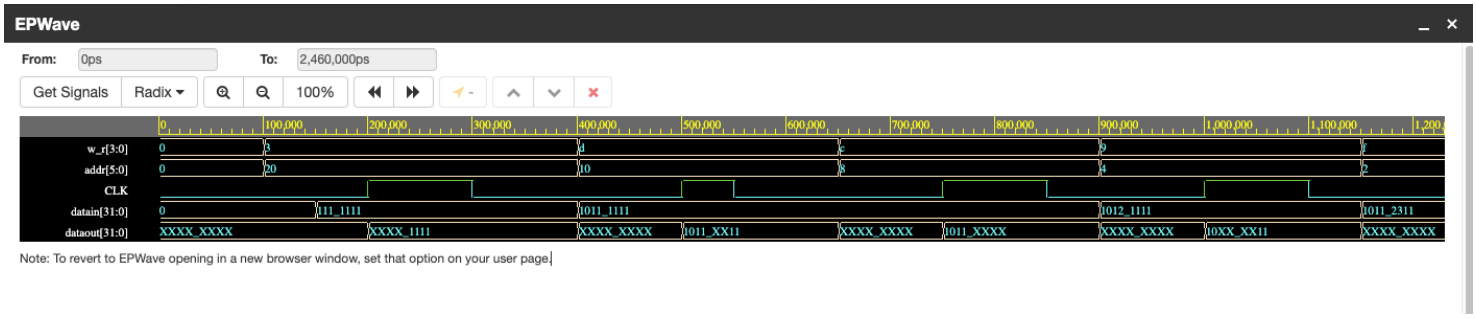
```
Device Utilization for 7A100TCSG324
# Info: ***********************************************************
# Info: Resource                          Used    Avail   Utilization
# Info: -----------------------------------------------------------
# Info: IOs                               38      210      18.10%
# Info: Global Buffers                    0       32        0.00%
# Info: LUTs                              3       63400     0.00%
# Info: CLB Slices                        1       15850     0.01%
# Info: Dffs or Latches                   0       126800    0.00%
# Info: Block RAMs                        0       135       0.00%
# Info: DSP48E1s                          0       240       0.00%
# Info: -----------------------------------------------------------
# Info: ************************************************


Info: Library: work    Cell: PRGM    View: BEV
# Info: ************************************************
# Info:  Number of ports :                          38
# Info:  Number of nets :                           48
# Info:  Number of instances :                      42
# Info:  Number of references to this view :         0
# Info: Total accumulated area :
# Info:  Number of LUTs :                            3
# Info:  Number of Primitive LUTs :                  3
```

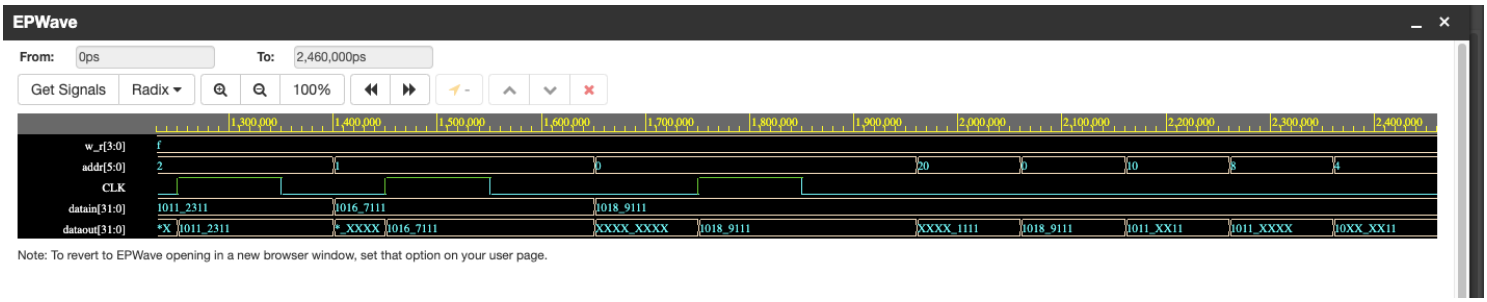
```
# Info:  Number of accumulated instances :          42
# Info: ****************************
```

4. Data Memory

We are given one address where we are continuously reading, and at the rising clock edge, we write to the memory. The 'write' here is a bit vector, in which the position of the 1 corresponds to the byte we want to overwrite. For example, a write value of 1010 implies that we want to change the 1st and 3rd byte value of the memory contents at the specified address. The non-1 bytes remain unchanged.

Here is the output of my epwave corresponding to my testbench:





Here is the result of my synthesis:

```
Device Utilization for 7A100TCSG324
# Info: ***********************************************************
# Info: Resource                          Used    Avail   Utilization
# Info: -----------------------------------------------------------
# Info: IOs                               75      210      35.71%
# Info: Global Buffers                    1       32        3.12%
# Info: LUTs                              32      63400     0.05%
# Info: CLB Slices                        8       15850     0.05%
# Info: Dffs or Latches                   0       126800    0.00%
# Info: Block RAMs                        0       135       0.00%
# Info: Distributed RAMs
```

```
# Info:    RAM64X1S                       32
# Info: DSP48E1s                          0       240       0.00%
# Info: ------------------------------------------------------------
# Info: ************************************************


# Info: Library: work    Cell: DATA    View: BEV
# Info: ************************************************
# Info:  Number of ports :                            75
# Info:  Number of nets :                            150
# Info:  Number of instances :                        76
# Info:  Number of references to this view :           0
# Info: Total accumulated area :
# Info:  Number of LUTs :                             32
# Info:  Number of Primitive LUTs :                   32
# Info:    Number of LUTs as Distributed RAM :        32
# Info:  Number of accumulated instances :           107
# Info: ****************************
```