EE 789 : Assignment 1

Rohan Rajesh Kalbag
20D170033

September 2023

1 Shift and Subtract Algorithm for Division

2(a) Describe the algorithm using Aa, and write a testbench for the design.

1.1 Pseudo Code For the Algorithm

```
shift_and_subtract_divison(a : unsigned 8-bit, b : unsigned 8-bit)
returns -> quotient : unsigned 8-bit
{
    curr_quotient: unsigned 8-bit
    curr_remainder: unsigned 8-bit
    count: unsigned 8-bit
    curr_b : 8-bit

    count = 0
    curr_quotient = 0
    curr_remainder = 0
    curr_b = b

    while count is < 8 {
        if curr_remainder is >= b{
            //add next bit from dividend to remainder and also subtract divisor
             curr_remainder = (((curr_remainder << 1) | (a >> (7 - count) & 1)) - b

            //mask the corresponding bit in quotient to 1
            curr_quotient = (curr_quotient | (1 << (7 - count))
        } else{
            // add the next bit from dividend to remainder
            curr_remainder = ((curr_remainder << 1) | (a >> (7 - count) & 1)
        }
        count = count + 1
        curr_b = curr_b >> 1
    }

    quotient = curr_quotient
}
```

1.2 Test Vector Generation

The strategy used to generate the test vectors, was to include at least one number from each of the 25 interval of ten numbers [0-10], [10-20]...[240-250], and the final interval of [250-255], a python script was used to generate 52 values of a and b, each having two representatives of each interval and both were shuffled randomly following this python script. In addition to this, we make sure b does not have zero (to prevent **Division By Zero Exception**)

```python
import random

# for a

l = [i + random.randint(0, 9) for i in range(0, 250, 10)]
l.extend([i + random.randint(0, 9) for i in range(0, 250, 10)])
l.append(250 + random.randint(0, 5))
l.append(250 + random.randint(0, 5))
random.shuffle(l)
print(l)

# for b
l = [i + random.randint(0, 9) for i in range(10, 250, 10)]
l.extend([i + random.randint(0, 9) for i in range(0, 250, 10)])
l.append(250 + random.randint(0, 5))
l.append(250 + random.randint(0, 5))
# to make sure there is no zero for b (divide by zero exception prevention)
l.append(0 + random.randint(1, 9))
l.append(0 + random.randint(1, 9))
random.shuffle(l)
print(l)
```

```
rohankalbag@Rohan:~/ee789-iitb/assignment-1/divider$ python3 test_generator.py
[48, 181, 113, 215, 102, 42, 156, 253, 55, 236, 234, 133, 27, 126, 174, 226, 62, 107, 250, 60, 166, 13, 95, 177, 29, 73, 91, 112, 201, 17, 36, 84, 186, 141, 78, 128, 86,
4, 191, 143, 229, 248, 9, 217, 130, 194, 162, 246, 33, 203, 153, 59]
[117, 29, 111, 3, 233, 57, 227, 120, 49, 218, 184, 148, 130, 16, 38, 146, 1, 153, 19, 58, 250, 209, 138, 101, 210, 240, 90, 94, 4, 239, 120, 28, 100, 205, 72, 77, 192, 22
8, 190, 162, 253, 68, 80, 30, 163, 185, 173, 46, 86, 157, 179, 69, 247]
```

1.3 Description of Algorithm using the AA language

```
$module [shift_and_subtract_div] $in (a b: $uint<8>) $out (quotient: $uint<8>) $is
{
    $branchblock[loop] {
        $merge $entry loopback
            $phi curr_b := b $on $entry next_b $on loopback
            $phi curr_quotient := ($bitcast ($uint<8>) 0) $on $entry next_quotient $on loopback
            $phi curr_remainder := ($bitcast ($uint<8>) 0) $on $entry next_remainder $on loopback
            $phi count := ($bitcast ($uint<8>) 0) $on $entry next_count $on loopback
        $endmerge

        $volatile continue_flag := (count < 8)
        $volatile next_bit_from_dividend := ((a >> (7 - count)) & 1)
        $volatile new_rem := ((curr_remainder << 1) | next_bit_from_dividend)
```

```
        $volatile sub_shifted := (new_rem >= b)
        $volatile new_quot := (curr_quotient | (1 << (7 - count)))

        next_count := (count + 1)
        next_remainder := ($mux sub_shifted (new_rem - b) new_rem)
        next_quotient := ($mux sub_shifted new_quot curr_quotient)
        next_b := (curr_b >> 1)


        $if continue_flag $then
            $place [loopback]
        $else
            quotient := curr_quotient
        $endif
    }
}
```

1.4 C Testbench Used

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <pthread.h>
#include <pthreadUtils.h>
#include <Pipes.h>
#include <pipeHandler.h>
#ifndef SW
#include "vhdlCStubs.h"
#endif

int main(int argc, char *argv[])
{
        uint8_t a[52] = {48, 181, 113, 215, 102, 42, 156, 253, 55, 236, 234, 133, 27, 126,
        ↪  174, 226, 62, 107, 250, 60, 166, 13, 95, 177, 29, 73, 91, 112, 201, 17, 36, 84,
        ↪  186, 141, 78, 128, 86, 4, 191, 143, 229, 248, 9, 217, 130, 194, 162, 246, 33,
        ↪  203, 153, 59};
        uint8_t b[52] = {117, 29, 111, 3, 233, 57, 227, 120, 49, 218, 184, 148, 130, 16,
        ↪  38, 146, 1, 153, 19, 58, 250, 209, 138, 101, 210, 240, 90, 94, 4, 239, 120, 28,
        ↪  100, 205, 72, 77, 192, 228, 190, 162, 253, 68, 80, 30, 163, 185, 173, 46, 86,
        ↪  157, 179, 69, 247};

        uint16_t c;
        int pass_tests = 0;
        int curr_testcase = 0;
        while (curr_testcase < 52)
        {
                c = shift_and_subtract_div(a[curr_testcase], b[curr_testcase]);
                if (c == a[curr_testcase] / b[curr_testcase])
                {
                        pass_tests += 1;
```

```
                fprintf(stdout, "PASS : div(%d, %d) == %d\n", a[curr_testcase],
                ↪  b[curr_testcase], c);
        }
        else
        {
                fprintf(stdout, "ERROR : div(%d, %d) != %d : Expected %d\n",
                ↪  a[curr_testcase], b[curr_testcase], c, a[curr_testcase] /
                ↪  b[curr_testcase]);
        }
        curr_testcase += 1;
    }

    fprintf(stdout, "%d testcases out of %d PASSED\n", pass_tests, 52);
    return (0);
}
```

1(b) Generating the VHDL and verifing it using the GHDL simulator and the C test-bench.

```
make
```

After compilation, use tmux and make two terminals for the docker container and run the following

```
# terminal 1
./testbench_hw
```

```
# terminal 2
./ahir_system_test_bench --wave=waveform.ghw
```

1.5 Terminal Output Obtained for ./testbench_hw

```
PASS : div(229, 253) == 0
PASS : div(248, 68) == 3
PASS : div(9, 80) == 0
PASS : div(217, 30) == 7
PASS : div(130, 163) == 0
PASS : div(194, 185) == 1
PASS : div(162, 173) == 0
PASS : div(246, 46) == 5
PASS : div(33, 86) == 0
PASS : div(203, 157) == 1
PASS : div(153, 179) == 0
PASS : div(59, 69) == 0
52 testcases out of 52 PASSED
```

1.6 GTKWave Waveform generated by GHDL Simulation

Using the screenshot of `waveform.ghw` generated while solving this assignment

```
# use on host system to view waveform
gtkwave waveform.ghw
```

Signals for last few multiplications is attached below, corresponding to those shown in above terminal