Embedded Hardware Design(EL203)

Signed Divider

Assigned by ***Prof. Biswajit Mishra***

Group Members

Ruchit Solanki	201701199	(24%)
Rahul Chhatraliya	201701200	(22%)
Jaymin Parmar	201701203	(30%)
Jaykumar Dave	201701204	(24%)

Problem Statement

A Divider for signed (2's complement) binary numbers that divides a 32-bit dividend by a 16-bit divisor to give a 16-bit quotient.

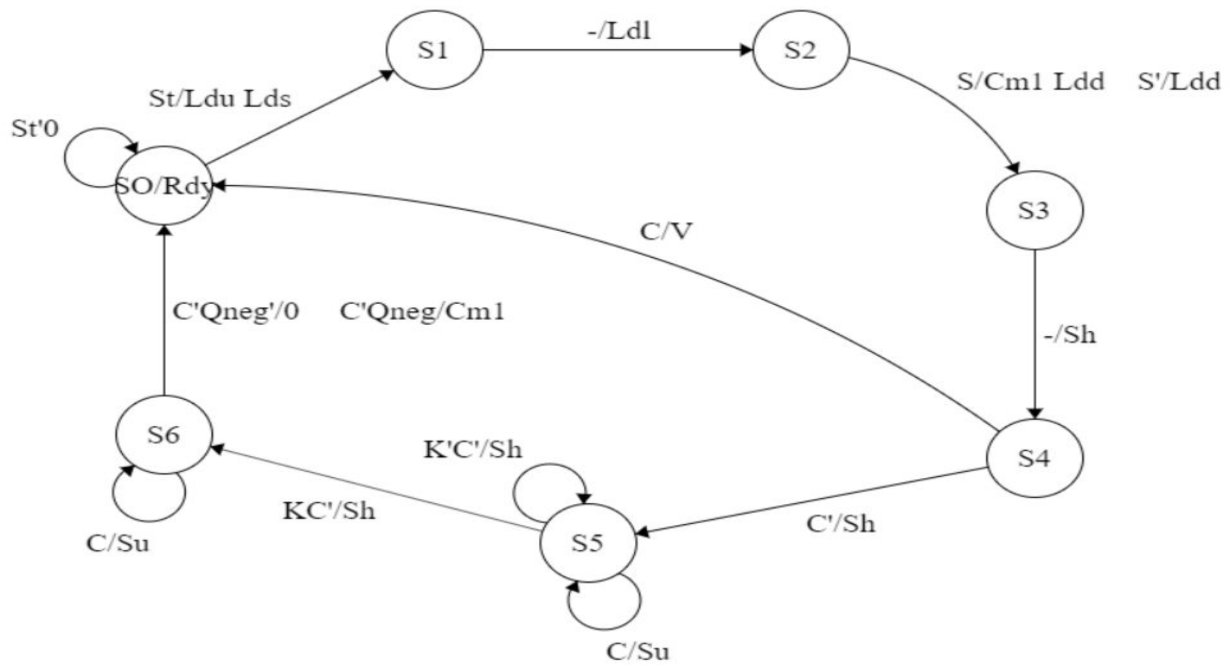
Description

Here we have developed a signed divider by using a successive shift and subtraction operations.We complement the dividend if they are negative, and after completion of division we change the sign of quotient. We need to subtract divisor from dividend so when we have negative divisor, we don’t have to complement. For overflow check we first left shift the dividend and then compare first 16 bits if they are greater or equal to divisor we would give overflow because quotient wouldn’t able to fit in 16 bits.

Control signals:

- **LdU** Load upper half of dividend from bus.
- **LdL** Load lower half of dividend from bus.
- **Lds** Load sign of dividend into sign flip-flop.
- **S** Sign of dividend.
- **Cm1** Complement dividend register (2’s complement).
- **Ldd** Load divisor from bus.
- **Su** Enable adder output onto bus (Ena) and load the upper half of dividend from bus.
- **Cm2** Enable complementer. (Cm2 equals the complement of the sign bit of the divisor, so a positive divisor is complemented and a negative divisor is not.)
- **Sh** Shift the dividend register left one place and increment the counter.
- **C** Carry output from adder. (If C _ 1, the divisor can be subtracted from the upper dividend.)
- **St** Start.
- **V** Overflow.
- **Qneg** Quotient will be negative. (Qneg _ 1 when the sign of the dividend and divisor are different

State Diagram



Algorithm/ Pseudo code

1. Load the upper half of dividend through bus
2. Load the Lower half of dividend through bus
3. If sign of dividend is negative we do complement to make it positive. Load the divisor and shift it to check overflow.
4. If it’s overflow go to 7
5. If c=1, we get non negative sum. We subtract divisor from acc(first 16 bits of dividend) and go to step 5. else we left shift dividend to make sum non negative and go to step 5. If k=1 which means we have done 16 shifts. If k=1 and c=0, we go to step 6
6. If c=1, we subtract it then change the sign of quotient accordingly.
7. Here we have quotient or overflow. If new division comes then we will go to 1.

Code snippets

1. We complement the divisor if it is positive, else not. As stated above we do division using shift and subtraction (Addition with complement). If divisor is positive we would add complement of divisor but for negative divisor, it is unnecessary to do that.

```
Cm2 <= not divisor(15);
compout <= divisor when Cm2 = '0'
           else not divisor;
```

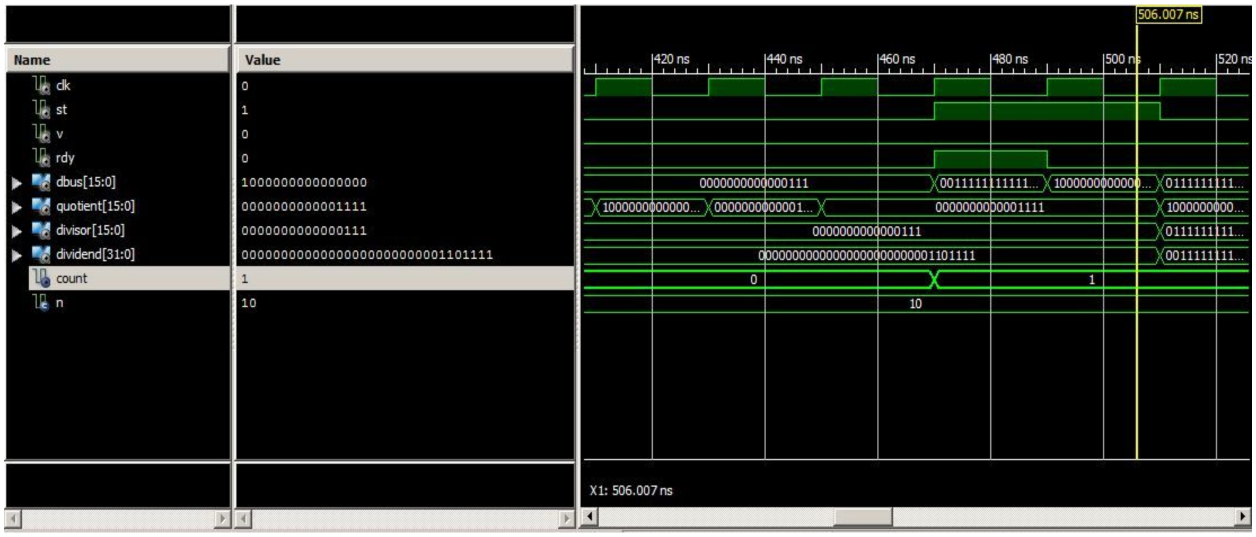
2. After the shift and subtractions, quotient will change effectively by these lines.

```
elsif (Sign xor Divisor(15)) = '1' then
  Dividend <= not Dividend + 1;
```

Code:- https://github.com/Ruchit22solanki/SignedDivider

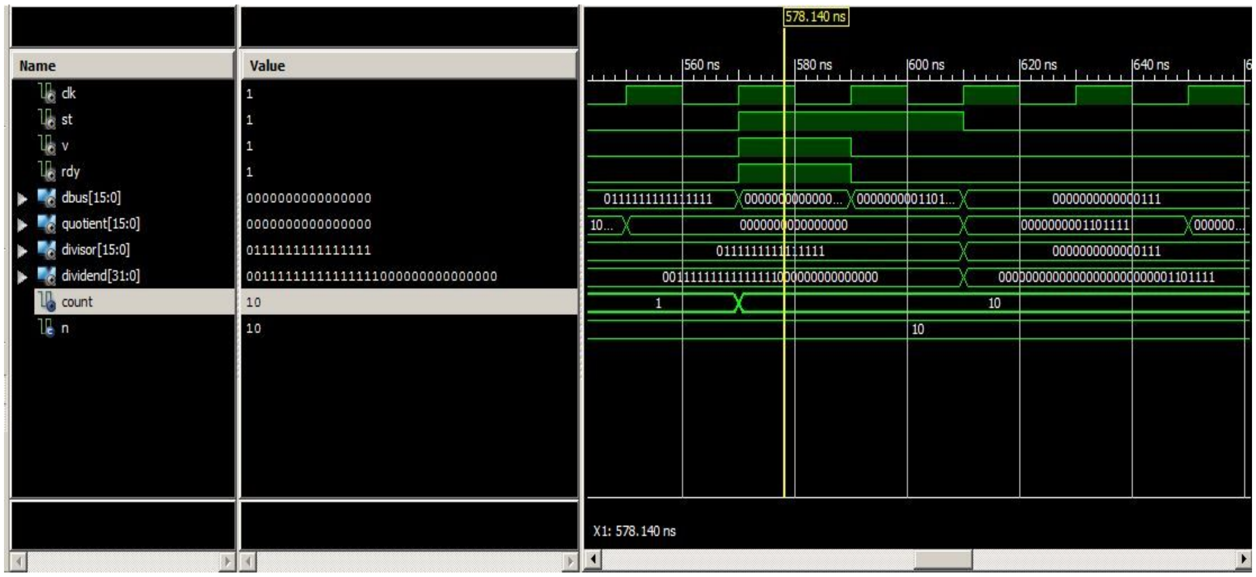
RESULTS

1) 111 / 7



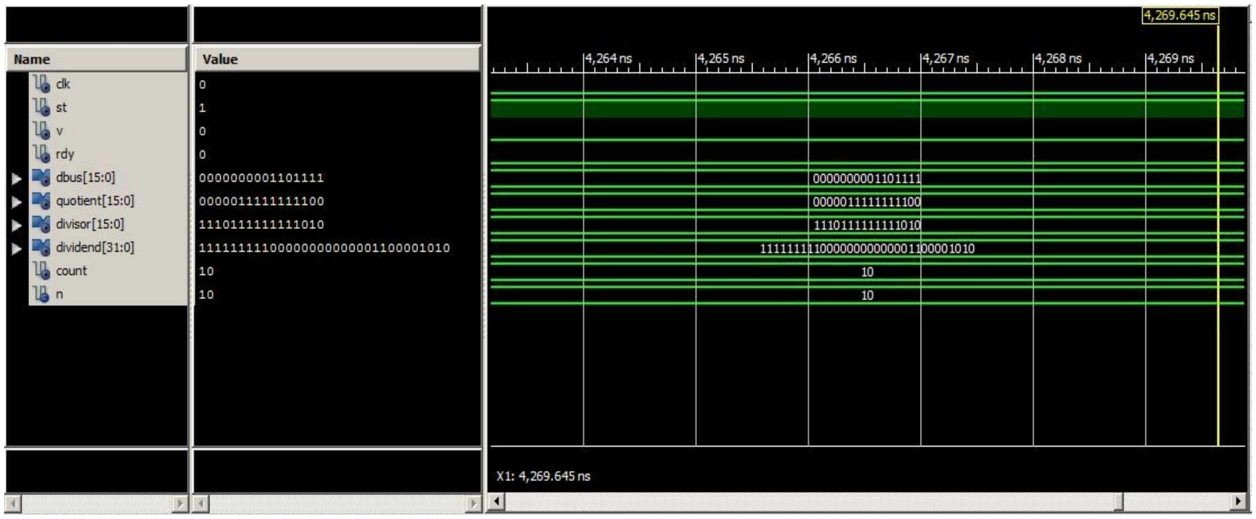
```
Name            Value
clk             0
st              1
v               0
rdy             0
dbus[15:0]      1000000000000000
quotient[15:0]  0000000000001111
divisor[15:0]   0000000000000111
dividend[31:0]  00000000000000000000000001101111
count           1
n               10
```

2) 0X3FFF 8000 / 0X7FFF **(Overflow)**



Name	Value
clk	1
st	1
v	1
rdy	1
dbus[15:0]	0000000000000000
quotient[15:0]	0000000000000000
divisor[15:0]	0111111111111111
dividend[31:0]	00111111111111111000000000000000
count	10
n	10

3) 0xFF80030A / 0xEFFA **(Dividend<0 and Divisor<0)**



Name	Value
clk	0
st	1
v	0
rdy	0
dbus[15:0]	0000000001101111
quotient[15:0]	0000011111111100
divisor[15:0]	1110111111111010
dividend[31:0]	11111111100000000000001100001010
count	10
n	10

CONCLUSION

In this project. We have designed a signed divider using VHDL. We also described algorithm for signed binary numbers. We added state diagram, ASM chart and result of some cases like Overflow, Negative numbers division. Used seven segment for output.

REFERENCES

1. Digital Systems Design Using VHDL (Charles Roth)
2. https://www.nandland.com/vhdl/modules/binary-to-7-segment.html
3. https://www.pantechsolutions.net/matrix-keypad-interfacing-with-spartan3e-fpga-development-kit