Vinculum Divider Using Flag Method

Ashish Singh 2021uee0132@iitjammu.ac.in

Supervisor: Dr. Rohit Chaurasiya

IIT Jammu

Abstract— This paper presents a comprehensive exploration of Vinculum numbers and their applications in digital systems. The study encompasses understanding Vinculum numbers, and conversion methods from various number systems to Vinculum representation, and vice versa. Notably, it focuses on the Flag method for both decimal and binary divisions, providing insights into its practical implementation. The proposed algorithm outlines the step-by-step process for Vinculum-based operations, offering a structured approach for digital system designers. Moreover, the paper emphasizes the practical implementation aspects, including VHDL coding for digital systems, functional verification using Vivado , and FPGA synthesis.

Introduction

Vinculum numbers, also known as Mishrank numbers, play a crucial role in Vedic mathematics [13]. These numbers consist of positive and negative identical numerals, where negative numerals are indicated by a bar on top of the number. The Flag method is highly efficient for converting digits greater than five into digits less than five. By converting certain digits, particularly those above five, the subsequent arithmetic operations become simpler and faster to execute. Not all digits need to be converted, but the conversion of specific digits can significantly reduce computation efforts. The combination of Mishrank numbers and this intriguing technique proves advantageous and effective.

Tasks-

- To Understand Vinculum Number
- Number Conversion to Vinculum Number
- Vinculum Number to Normal Number
- Binary Number Conversion to Vinculum Number
- Understanding Flag method-based division for Decimal Numbers
- Understanding & Performing Flag method-based division for Binary Numbers Vinculum Flag method for Binary Numbers

Objective -

1. To investigate the advantages of applying Vedic mathematics principles to the binary number system.

2. To explore the potential benefits of incorporating Vinculum numbers in the binary number system.

3. To develop and implement a division algorithm that combines the Vedic mathematics flag method with Vinculum numbers.

Resources-

Paper: https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/joe.2013.0213

Division using Flag Method Vinculum: https://mathlearners.com/vedic-mathematics/division-in-vedic-mathematics/anurupyena-vinculum/

https://instavm.org/

Conversion-

Number Conversion to Vinculum Number

Step 1) Check the Number From Right to left; find a digit that is greater than five (>5), take 10's compliment, and put a bar on it.
Step 2)(a)If the next digit is greater than five, its 9's complement is appended with a bar, and this process is repeated until a digit less than five is found.
or increase the number by one and take 10'complement. (b)Increase the less than five digit by one. Repeat step1 and step2 until the entire number is covered.
Step 3) Repeat steps 1 and step 2 until the entire number is covered.

Example 3.1) 127 to Vinculum number.

We can see as we move from right to left that 7 is greater than 5, 10' complement of 7 has been and converted to the bar with 3, and the next digit is increased with 1, 2 will become 3.

$$127 \longrightarrow 12\overline{7} \longrightarrow 13\overline{3}$$

We can see as we move from right to left that 7 is greater than 5,10' complement of 7 has been converted to the bar with 3, and the next digit is increased with 1, 7 will become 8, take 10's complement of 8 will become 2 bar and next number 2 will become 3.

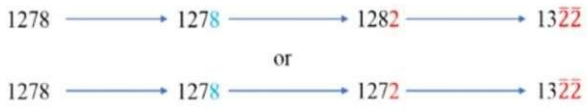


Figure 3.2: Conversion of 1278 to its Vinculum Equivalent

2. Vinculum Number Conversion to Normal Number

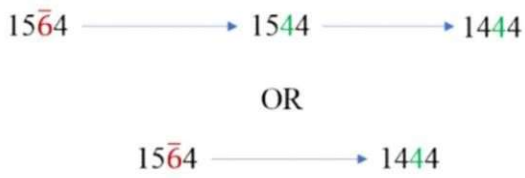
Step 1) Examine the Number From left to right; find the first Bar digit and take its 10's complement.

Step 2) If the following digit is also a bar digit, take its complement of nine and continue with **(step 1)** until a non-bar digit is obtained.

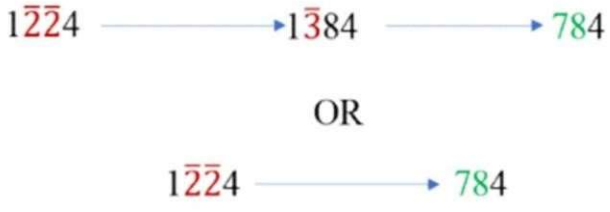
Step 3) Subtract 1 from the non-bar digit.

Step 4) Repeat (1) and (2) until the number has been covered.

Example 3.3) 15$\bar{6}$4



Example 3.4)1$\bar{2}\bar{2}$4



3.1.2 Vinculum on binary number

A vinculum in binary is the same as a decimal number, but in the case of the binary number system, the base is 2, and we use only two symbols to represent two different values. These symbols are 0 and 1 (to represent two different states, i.e., off and on). Instead of 10's complement, we will take compliments using 2 (10) in a binary system and increase the next digit by one.

1. Binary number Conversion to Vinculum Number

Step 1) Choose the digit we want covert as the vinculum number. Choosing the digit that will serve as the vinculum number carefully. The problem context will dictate which digits must be converted in certain instances.

Step 2) Increase the left digit by one after converting the digit. However, caution is required because 1+1 equals 10 in binary. This results in a carry. we need to take forward carry.

Example 3.5) 1001.

We are Changing the green number into the vinculum.

Step a) Take Compliments, i.e., 2(10) -1 = 1, and put a bar on top of it. It is written in red.

Step b) increase the next digit by adding one to the left digit.

1001 into vinculum $101\bar{1}$.

We can check this since,1001 = 8*1+4*0+2*0+1*1 = 8+1 =9 and $101\bar{1}$ = 8*1+4*0+2*1-1*1= 8+2-1=9.

Example-

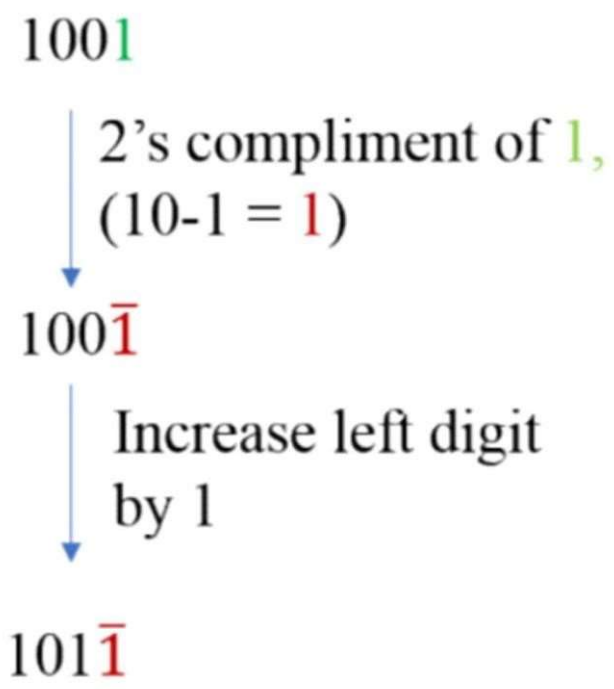


Figure 3.5: 1001 conversion to Vinculum Equivalent

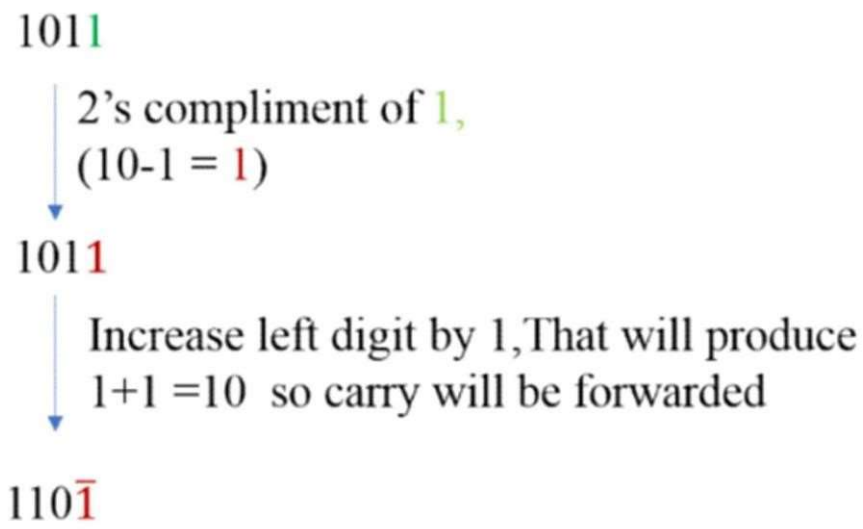


Figure 3.6: 1011 conversion to Vinculum Equivalent

In the introduction, section Flag method has been discussed for decimal; in this section, the application of the vinculum is discussed for the flag method and explores the advantage we can get by using the flag method. In this section, we are taking one example of applying the flag method for both with and without vinculum and observing what advantages we can get. First, we will explore the flag method in decimal numbers; afterward, the binary will be applied for the same.

3.2.1 Applying the Flag method for Decimal number

In this section, we are going to perform the flag method and going to discuss how vinculum can prove beneficial for the flag method. let's take one example and understand.

Example 3.11).Considering the division of 25382 by 77.

Step 1) From the divisor, we are taking the rightmost digit, which is 7, as a flag, as one digit is taken as a flag from the dividend. We will place a colon one digit from the right, i.e., equal to the number of digit in the flag.
Step 2) Division is performed using the single digit that is left after taking the flag, which is 7.
Step 3) First, we start dividing from the left side. Since 2 is less than 7, we take the first two digits, which are 25, as the dividend. When dividing 25 by 7, the quotient is 3, and the remainder is 4.
Step 4) Remainder 4 is carried out and added to the subsequent digit 3 to get the new dividend 43, which is called the gross dividend (GD).
Step 5) Now the flag is multiplied by quotient 3, 7X3 = 21, and subtracted from 43 i.e. 43-21 = 22, which is called the net dividend (ND).
Step 6)22 divided by 7 again, we get quotient 3 and the remainder as 1.
Step 7) Carry the remaining 1 to the following digit 8 again to get 18 (GD).
Step 8) Similar to step 5, the flag is multiplied by quotient 3 and subtracted by 18 i.e. 18-21 = -3 (ND). Which is negative, so division can not proceed further. We need to go back and do the Coloum adjustment process.
Step 9) Instead of writing the quotient as 3 when dividing 22 by 7, we will write only 2 and continue a larger remainder of 8 (22 - 7 * 2) forward.
Step 10) remainder 8 will be carried out to the next digit 8, and we will get 88 (GD). Similar to the above step, multiply quotient 2 by flag 7 and subtract from 88-14 = 74 (ND), which is positive.
Apply the procedure to the remaining columns, adjusting each time the net dividend becomes negative.
The final answer is 329, with the remainder being 49 or 329.63 for the decimal equivalent.

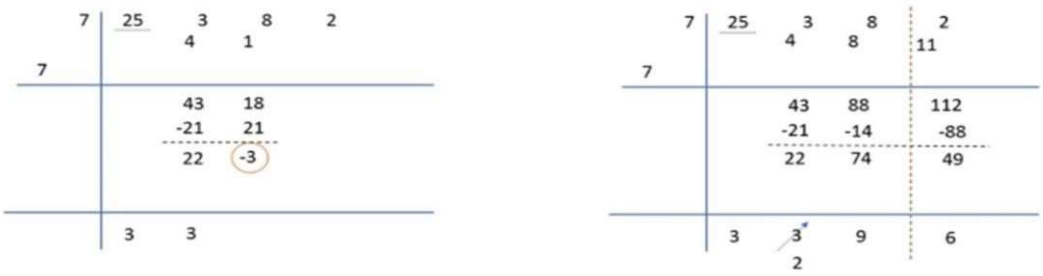
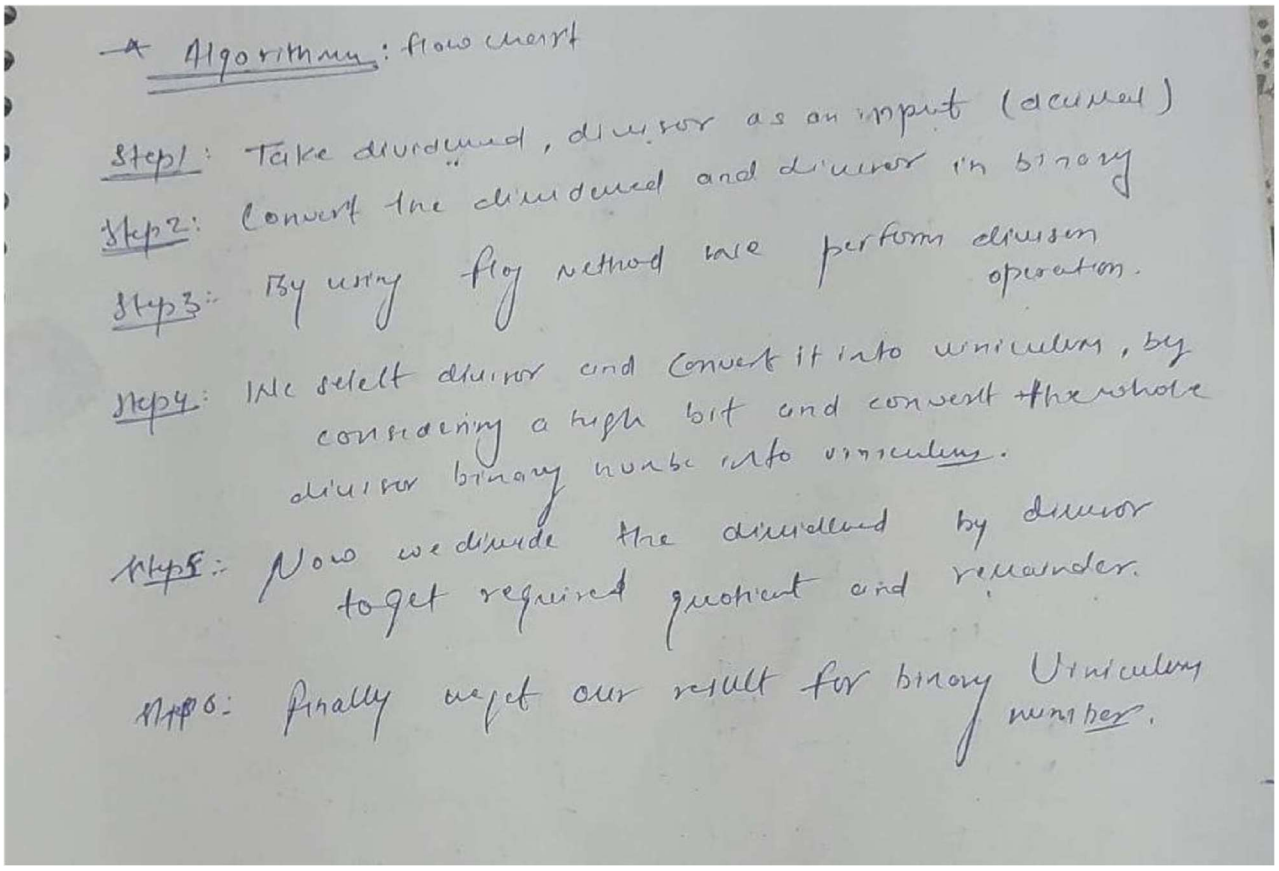


Figure 3.12: 25382 divided by 77 without vinculum



VHDL Project

Conversion :-

1) Venicular Number → Normal number
(R→L)

1. find 1st bar digit, takes it's 10's complement.
2. a) if next digit is again bar digit, then take it's 9's complement continue until
∴ i) a non-bar digit is obtained.
ii) decrement non-bar digit by 1.

2) Normal Number → Venicular Number
(R→L)

1. find 1st digit >5, take it's 10's complement with a bar over it.
2. a) if next digit >=5, take it's 9's complement with a bar over it and continue till a digit <5 is obtained
b) Increment <5 digit by 1.
3. continue till number is complete.

→ change 7329850 to Venicular

7329850 → $1\bar{3}3\bar{3}0\bar{2}50$

→ change $1\bar{3}3\bar{3}0\bar{2}50$ to Normal

$1\bar{3}3\bar{3}0\bar{2}50$ → 7329850

Note :- '0' is also coundered as already to Vinicular number.

* Viniculum on binary Number:-

1) Binary Viniculum Number → Binary Number (R→L)

a) find 1st bar digit and take it's 2's complement

b) if next digit is again bar digit then take it's 1's complement until:-

a) non-bar digit is obtained

b) decrement non-digit by 1.

→ change $1\bar{1}\bar{1}0\bar{1}1$ to binary:-

$1\bar{1}\bar{1}0\bar{1}1 \rightarrow 0001,11$

Note:- Assume '0' also a bar digit (Viniculum binary digit).

11) Binary number Converter to Viniculum binary:-

a) Choose the digit we want to convert of viniculum number (2's complement)

b). Increase the left digit by one by one after converting the digit, However caution is required because $1+1=10$ in binary, we need to take forward carry.

change:- 1001 → this to Viniculum

$1001 \rightarrow 101\bar{1}$

change:- $101\bar{1} \rightarrow 11\bar{0}\bar{1}$

↓ this to viniculum

Applications :-

① ⇒ 8988 − 6869

```
 8988
+6869
------
 212T
```

2121̄ → 2119

(N) → (N)

②

```
 1101011
-111110
---------
 10T0T01
```

10T0T01 → 0101101

(BV) → (B)

Note:- Consider 0 also Binary Universalis ∴ take its 1's complement

* flag method for division :-

i) 1234 ÷ 12 (single digit flag)

1²	1	0	2	0	3	4
	1	0	2			10

(02 − 2×1) = 0

(03 − 0×2) = 03

(04 − 06) = −ve (✗)

⇒ decrease the quotient by 1.

(14 − 2×2) = 10

= 102/10

ii) (2823 ÷ 22 15623 ÷ 12)

1²]	1	5	6	2	3]
	1	7	9	2	

127/2

* Note :-

In flag method on subtraction −ve term occurs then decrease the quotient by 1.

* Division in binary:

```
      016
   8 | 128
       0
      ---
       12
       8
      ---
       48
       48
      ---
        x
```

```
        000111
   110 | 101010
         0
        ------
         10
          0
        ------
         101
          0
        ------
         1010
          110
        -------
         01001
           110
        -------
          0110
           110
        -------
            x
```

```
        011010
   110 | 100111 00
         0
        ------
         1001
          110
        ------
          0011 1
           110
        -------
          0001 11
             0
          ------
           110
           110
          -----
             0
```

11010 → quotient

```
        111
   111 | 110010
         111
        ------
         01011
          111
        ------
         01000
           111
        ------
         00011
```

0-1 = 1 (with 1 borrow)

Quotient = 111
Remainder = 011

Subtraction:-

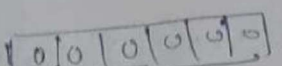
```
  1 1 0 0
  0 1 0 1
  -------
  0 1 1 1
```

```
  1 1 0 0 0
  0 0 0 0 1
  ---------
  1 0 1 1 1
```

```
  1 1 0 0 0
  0 1 1 1 1
  ---------
  0 1 0 0 1
```

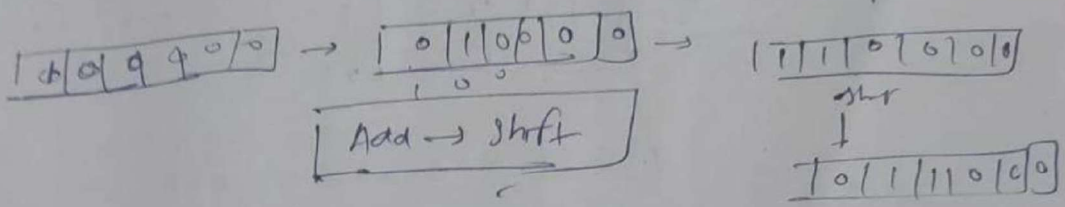
Multiplication in binary:

```
  100
x 110
```

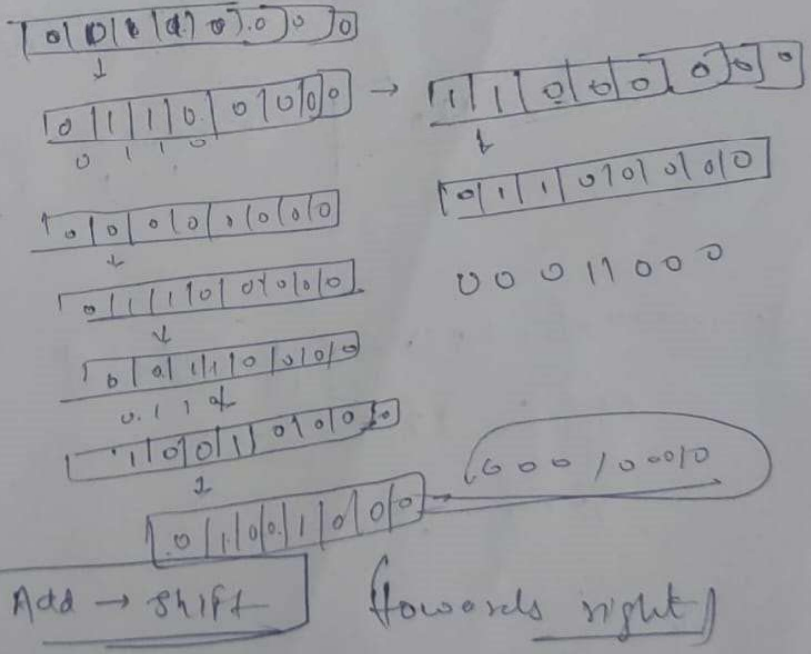


i) Put Multiplication result of 1st multiplier bit and then shift by 1.

ii) Again do the same until all bits are completed




```
→ 0 1 1 0
  0 0 1 1
  -------
  0 1 1 0
+ 0 1 1 0
  -------
  1 0 0 1 0
+ 0 1 1 0
  -------
  1 0 1 0 0
+ 0 1 1 0
  -------
  1 0 0 0 1 0
```



→ Division using flag Method for binary Numbers:-

i) $25 \div 12$ $(11001) \div (1100)$

for single flag method :-

1100	11001	01
	1001	01

$00 - (0 \times 1) = 0$

$11001 = 1100 \times 01 + 01$

$25 = 12 \times 2 + 1$

ii) $257 \div 12$ $(100000001) \div (1100)$

128 64 32 16 8 4 2 1

1100	100000001	01
	10101	

$1000 - 110$

$$\begin{array}{r} 1000 \\ -110 \\ \hline 0010 \end{array}$$

$100000001 = 1100 \times 10101 + 01$

$257 = 12 \times 21 + 5$

$257 = 257$

→ $100 - 0 \times 1 = 100$

→ $1000 - 110 = 0010$

→ $100 - 0 \times 1 = 100$

→ $1000 - 110 = 0010$

(iii) $312 \div 44$

$(100111000) \div (101100)$

101100	100111000	0
	111	

$$\begin{array}{r} 100111 \\ 10110 \\ \hline 10001 \end{array}$$ = Remainder

100111000 = 101100 × 111 + 100

312 = 44 × 7 + 5

1) 100111 − 10110 = 10001 (Repeat

(1) 100010 − 0×1 = 100010

(2) 100010 − 10110 = 001100 → Rem

(3) 0011000 − 0×1 = 11000

(4) 11000 − 10110 = 0010

100 − 1× = 100 → Rem

→ for 2 bit flag:-

590 ÷ 14

(1001001110) ÷ (1110)

11 | 10 | 10010011 | 10
| | 101010 | 10

(i) 100 − 11 = 01 (Remainder)

(ii) (01) − (1×1) = 10

(iii) 100 − 11 = 01

10 − (1×1 + 0×0) = 1

(iv) 11 − 11 = 0 (Rem

(v) 01 − (1×1 + 0×0) = 0

010 − (0×1 + 1×0) = 10

1001001110 = 1110 × 101010 + 10

590 = 14 × 42 + 2

= 588 + 2

= 590

* Working for Minimum Numbers:-

1) 828432 ÷ 38

(828432 ÷ 42)

$4^{\bar{2}}$	$8^{0}\ 2^{2}\ 8^{2}\ 4^{2}\ 3^{1}$	2
	2 1 7 9 10	32

→ $(02 - 2\times\bar{2}) = 6$

→ $(28 - \bar{2}\times 1) = 30$

→ $(24 - \bar{2}\times 7) = 38$

→ $(23 - 2\times 9) = 41$

→ $(12 - \bar{2}\times 10) = 32$

$\frac{21800}{32}$

ii) 170 ÷ 15 ((10101010) ÷ 1111)

1111 → $1000\bar{1}$

$1000^{\bar{1}}$	1 0 10 ^{10} 10 ^{100} 1 ^{10} 0
	1 · 0 1 1 101

→ 1010 − 1000 = 10

→ 101 + 1 = 110

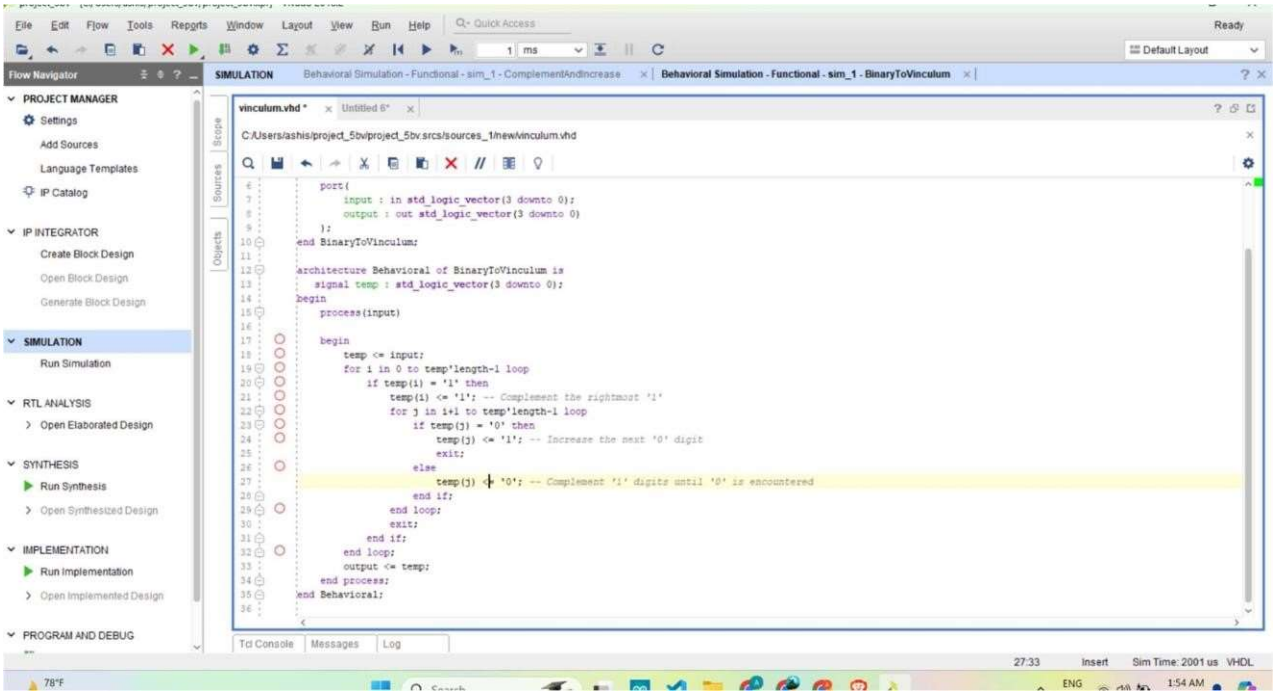
→ 1100 − 1000 = 100

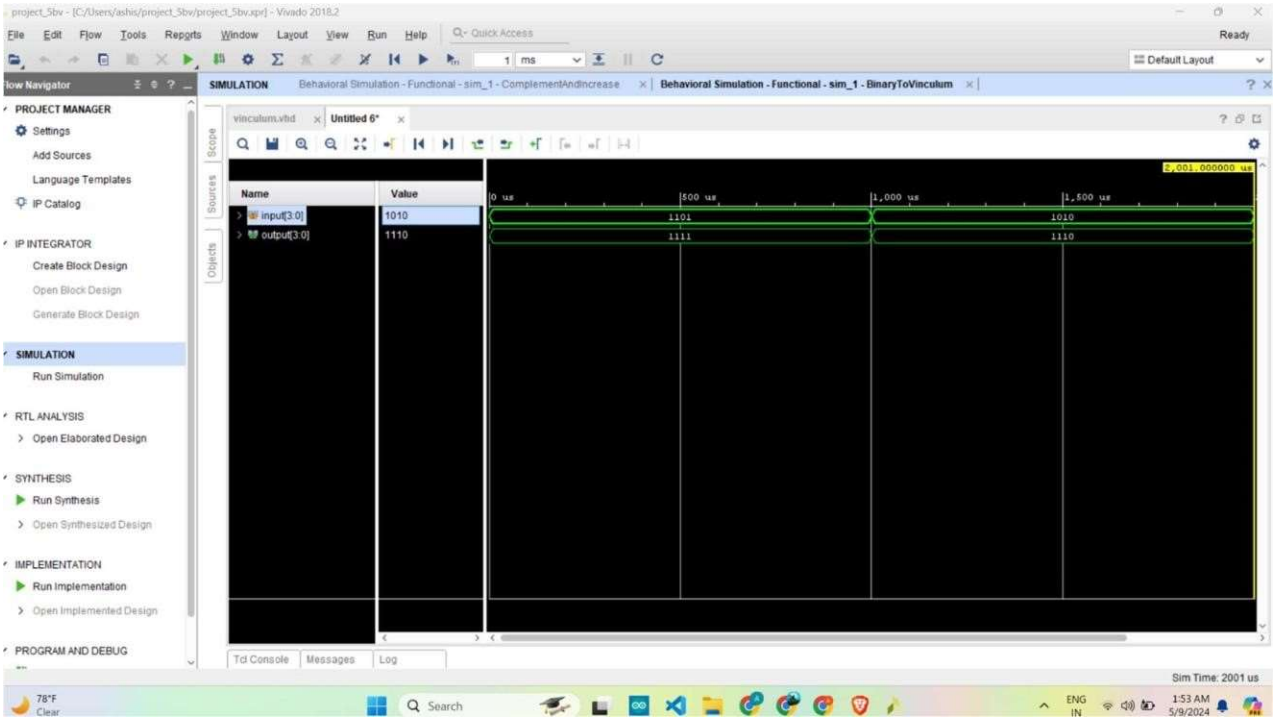
→ (101 11 ∠ 1010

→ 1010 − 1000 = (10)

100 − 1 = 101

Code for Binary to Vinculum





Code for binary viniculum number divison using flag method

binary_viniculum.vhd

C:/Users/yadav/OneDrive/Desktop/VHDL/binary_viniculum_number_end_sem/binary_viniculum_number_end_sem.srcs/sources_1/new/binary_viniculum.vhd

```
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.NUMERIC_STD.ALL;
23
24  entity Binary_Vinculum_Division is
25      generic (
26          DIVIDEND_LENGTH : integer := 8;  -- Length of the dividend
27          DIVISOR_LENGTH : integer := 4    -- Length of the divisor
28      );
29      Port ( Dividend : in  std_logic_vector(DIVIDEND_LENGTH - 1 downto 0);
30             Divisor : in  std_logic_vector(DIVISOR_LENGTH - 1 downto 0);
31             Quotient : out  std_logic_vector(DIVIDEND_LENGTH - DIVISOR_LENGTH downto 0);
32             Remainder : out  std_logic_vector(DIVISOR_LENGTH - 1 downto 0));
33  end Binary_Vinculum_Division;
34
35  architecture Behavioral of Binary_Vinculum_Division is
36  begin
37      process(Dividend, Divisor)
38          variable quotient_reg : std_logic_vector(DIVIDEND_LENGTH - DIVISOR_LENGTH downto 0);
39          variable remainder_reg : std_logic_vector(DIVISOR_LENGTH - 1 downto 0);
40          variable temp_dividend : std_logic_vector(DIVIDEND_LENGTH - 1 downto 0);
41          variable temp_divisor : std_logic_vector(DIVISOR_LENGTH - 1 downto 0);
42      begin
43          temp_dividend := Dividend;
44          quotient_reg := (others => '0');
45          remainder_reg := (others => '0');
46          temp_divisor := Divisor;
47
48          for i in 0 to DIVIDEND_LENGTH - DIVISOR_LENGTH loop
49              if temp_dividend(DIVIDEND_LENGTH - 1) = '1' then
50                  -- If the most significant bit of the dividend is 1, complement the dividend
51                  temp_dividend := std_logic_vector(unsigned(not(unsigned(temp_dividend))) + 1);
```

binary_viniculum.vhd

C:/Users/yadav/OneDrive/Desktop/VHDL/binary_viniculum_number_end_sem/binary_viniculum_number_end_sem.srcs/sources_1/new/binary_viniculum.vhd

```
43          temp_dividend := Dividend;
44          quotient_reg := (others => '0');
45          remainder_reg := (others => '0');
46          temp_divisor := Divisor;
47
48          for i in 0 to DIVIDEND_LENGTH - DIVISOR_LENGTH loop
49              if temp_dividend(DIVIDEND_LENGTH - 1) = '1' then
50                  -- If the most significant bit of the dividend is 1, complement the dividend
51                  temp_dividend := std_logic_vector(unsigned(not(unsigned(temp_dividend))) + 1);
52                  -- Complement the divisor
53                  temp_divisor := std_logic_vector(unsigned(not(unsigned(temp_divisor))) + 1);
54                  -- Increment the remainder by 1
55                  remainder_reg := std_logic_vector(unsigned(remainder_reg) + 1);
56              end if;
57
58              remainder_reg := remainder_reg(DIVISOR_LENGTH - 2 downto 0) & temp_dividend(DIVIDEND_LENGTH - 1);
59              if remainder_reg >= temp_divisor then
60                  remainder_reg := std_logic_vector(unsigned(remainder_reg) - unsigned(temp_divisor));
61                  quotient_reg(i) := '1';
62              else
63                  quotient_reg(i) := '0';
64              end if;
65
66              temp_dividend := temp_dividend(DIVIDEND_LENGTH - 2 downto 0) & '0'; -- Shift dividend left by 1
67          end loop;
68
69          -- Output Quotient and Remainder
70          Quotient <= quotient_reg;
71          Remainder <= remainder_reg;
72      end process;
73  end Behavioral;
74
```