ADDITION, SUBTRACTION, MULTIPLICATION AND DIVISION OF TWO 64BIT NUMBERS IN VHDL

[1]Chandana C S, [2]Chaitra G, [3]Churanta Mondal, [4]Sanju V

Authors affiliation, Department of Computer Science and Engineering, REVA University

Abstract.

In today's world most of the processors and systems runs on a 32-bit processor. At some extent it gives us the error less, fast, and accurate value of inputs, but it has some drawback i.e., it can take the input up to 32-bits only and gives the solution till that extent. So, we propose a solution for a 64-bit processor which can add, subtract, multiply and divide two 64-bit numbers in VHDL code. Our motive is to make the system faster than usual which would provide the fastest, less time complexity and the accurate solution for an input.

Keywords. 32-bit processor, 64-bit processor, VHDL code, resistors, LDRD, STRD, ADC, UMULL.

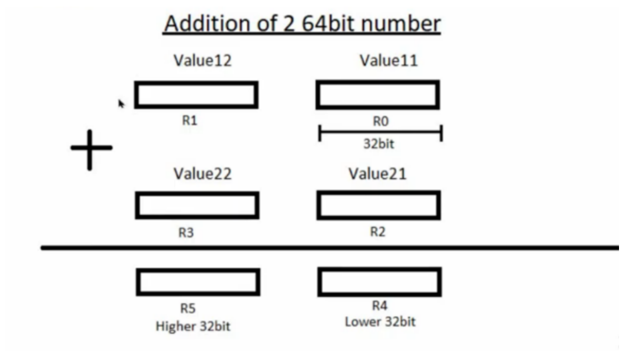
1. INTRODUCTION

We all know that the capability of 32-bit processor is lesser than 64-bit processor, so data handling is lesser in 32-bit compared to 64-bit processor. A 32-bit processor doesn't have much capability of computing the computational value as compared to 64-bit processor. Also 64-bit processor has more memory address compared to 32-bit processor. With the help of our code, we can perform any basic calculation like addition, subtraction, multiplication, and division using two 64-bit numbers. This helps in increasing the processing power and decrease the time complexity also it provides more space. Our code gives an errorless solution and provides the fastest solution of basic mathematical calculation. We aim to make the processor run faster than others and provide the best solution.

2. ADDITION OF TWO 64-BIT NUMBERS

We all know the addition of two 32-bit numbers and 32-bit is the maximum bit for a register so now we will see the addition of two 64-bit number that is double the range of register.

Basically, the logic part is that you should remember that any operation that should be made on the data or the values must be done using registers that is we can only perform operation when the values are in register.

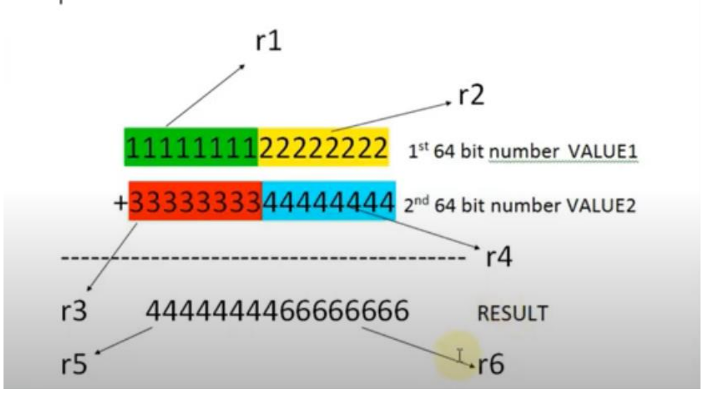


So R1+R0 is the whole one 64-bit number and R3+R2 is another whole 64-bit number, when we perform addition of these you get another 64-bit number.

The number is divided into two 32-bit numbers, higher 32 bit and lower 32 bit and then we will perform normal addition. Lower 32-bit is added and stored in R4, and higher 32-bit is added and stored in R5.

This is done because you cannot add a directly 64-bit number because the maximum value of register is 32 bit and for addition you must use register itself, so we divide the number into 2 parts and add them.

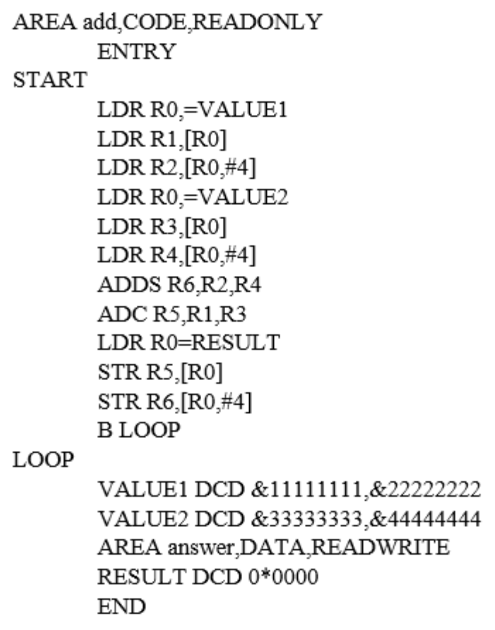
This value 11, value12, value 22 and value 21 are the memory locations the lower and higher 32 bit of first number are stored in value 12 and value 11, the second number are stored in value21 and value21. We will take the numbers from the memory location into the register and then we will perform addition and we will get the result in R5 and R4.



Now the first part will be stored in register R1 the second part will be stored in registerR2. similarly with the second number, the first part will be stored in R3, and the second part will be stored in R4.

The logic is that the content of R2 and R4 will be stored in R6 and the content of R1 and R3 will be stored in R5.

VHDL CODE FOR ADDITION OF TWO 64-BIT NUMBERS



```
AREA add,CODE,READONLY
          ENTRY
START
          LDR R0,=VALUE1
          LDR R1,[R0]
          LDR R2,[R0,#4]
          LDR R0,=VALUE2
          LDR R3,[R0]
          LDR R4,[R0,#4]
          ADDS R6,R2,R4
          ADC R5,R1,R3
          LDR R0=RESULT
          STR R5,[R0]
          STR R6,[R0,#4]
          B LOOP
LOOP
          VALUE1 DCD &11111111,&22222222
          VALUE2 DCD &33333333,&44444444
          AREA answer,DATA,READWRITE
          RESULT DCD 0*0000
          END
```

3. Subtraction of Two 64-bit Numbers

The basic operations are performed while subtraction of two 64-bit numbers is loading, storing, and subtracting the values.

At first, we should load the value in register R1 and load the resister value R1 in R2.

Then we should take two numbers using load registers [R1, #4] and the value will be stored in register R3, as well as the load registers value [R1, #8] will be stored at register R4.

This is a simple code which makes no error and performs task faster.

VHDL CODE FOR ADDITION OF TWO 64-BIT NUMBERS

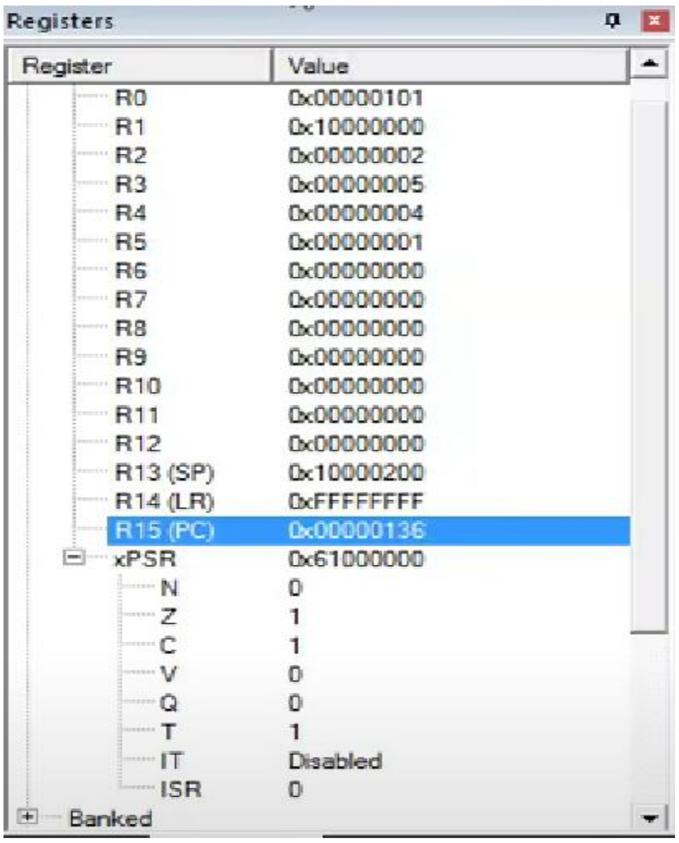
```
AREA BASIC, CODE, RANDONLY
ENTRY
```

4

```
        EXPORT      MAIN
MAIN
        LDR R1=0X10000000
        LDR R2, [R1]            ; COND
        LDR R3, [R1, #4]                ;1ST NUM
        LDR R4, [R1, #8]                ;2ND NUM

        SUB R5, R3, R4
        STRD R5, [R1, #0XC]

        NOP
        END
```



```
Registers
Register        Value
  R0            0x00000101
  R1            0x10000000
  R2            0x00000002
  R3            0x00000005
  R4            0x00000004
  R5            0x00000001
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x10000200
  R14 (LR)      0xFFFFFFFF
  R15 (PC)      0x00000136
  xPSR          0x61000000
    N           0
    Z           1
    C           1
    V           0
    Q           0
    T           1
    IT          Disabled
    ISR         0
Banked
```

4. Multiplication of Two 64-bit Numbers

Compared to addition and subtraction multiplication is a step forward to basic mathematical calculation. In this type of calculation, we usually use 64-bit numbers, we use some different instructions as compared to 32-bit numbers. The VHDL code consists of both addition and multiplication as we use binary multiplication method.

In this code we use LDRD (Load double word (64-bit) from memory to register), STRD (Store double word (64-bit) from memory to the register), UMULL (Unsigned long multiply with 64-bit result). Here at first, we take a 64-bit number and load it to R1 and R3. Then we perform unsigned long multiplication to get the values which next will be added and stored.

VHDL CODE FOR MULTIPLICATION OF TWO 64-BIT NUMBERS

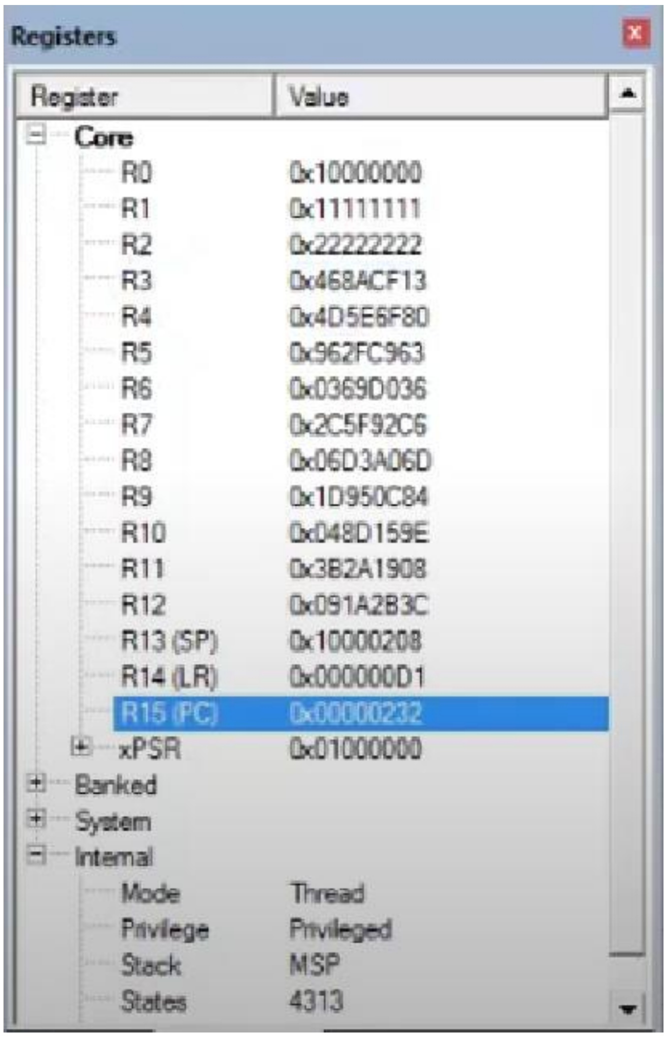
```
        AREA program, CODE, READONLY
        ENTRY
        EXPORT      main
main

        LDR R0, =0X10000000
        LDRD R1, R2, [R0]
        LDRD R3, R4, [R0, #8]

        UMULL R5, R6, R3, R1
        UMULL R7, R8, R3, R2
        UMULL R9, R10, R4, R1
        UMULL R11, R12, R4, R2

        ADDS R4, R6, R7
        ADC R4, R9
        STRD R5, R4, [R0, #16]
        ADC R3, R8, R10
        ADC R3, R8, R10
        ADC R3, R11
        ADC R12, #0
```

```
STRD R3, R12, [R0, #24]
NOP
END
```



Registers

Register	Value
Core	
R0	0x10000000
R1	0x11111111
R2	0x22222222
R3	0x468ACF13
R4	0x4D5E6F80
R5	0x962FC963
R6	0x0369D036
R7	0x2C5F92C6
R8	0x06D3A06D
R9	0x1D950C84
R10	0x048D159E
R11	0x3B2A1908
R12	0x091A2B3C
R13 (SP)	0x10000208
R14 (LR)	0x000000D1
R15 (PC)	0x00000232
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	4313

5. DIVISION OF TWO 64-BIT NUMBERS

As compared to subtraction we are using basic division, operations are performed by dividing two 64-bit numbers by loading, storing, and dividing the values.

At first, we should load the value in register R1 and load the resister value R1 in R2.

Then we should take two numbers using load registers [R1, #4] and the value will be stored in register R3, as well as the load registers value [R1, #8] will be stored at register R4.

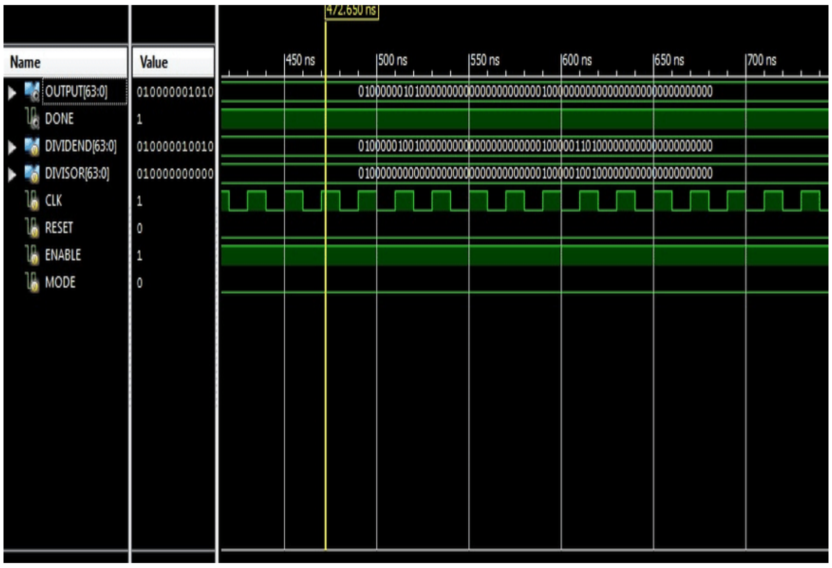
This is a simple code which makes no error and performs task faster.

<u>VHDL CODE FOR DIVISION OF TWO 64-BIT NUMBERS</u>

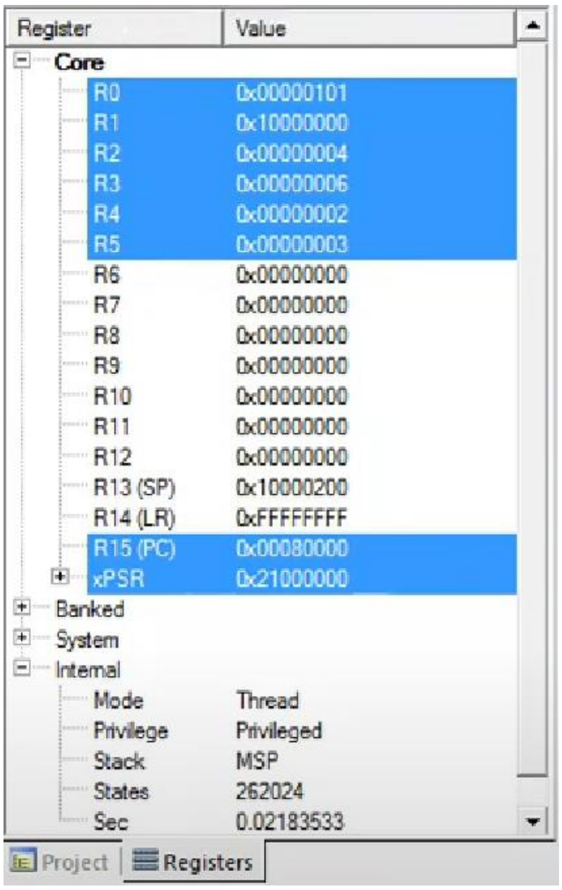
```
AREA BASIC ,CODE,RANDONLY
        ENTRY
        EXPORT          MAIN
MAIN
        LDR R1=0X10000000
        LDR  R2,[R1]            ;COND
        LDR  R3,[R1,#4] ;       1ST NUM
        LDR  R4,[R1,#8] ;       2ND NUM

        UDIV  R5,R3,R4
        STRD R5,[R1,#0XC]

        NOP
        END
```



```
Name            Value
OUTPUT[63:0]    010000001010    0100000010100000000000000000000000001000000000000000000000000000000000
DONE            1
DIVIDEND[63:0]  010000010010    0100000100100000000000000000000000001000001101000000000000000000000000
DIVISOR[63:0]   010000000000    0100000000000000000000000000000000001000001001000000000000000000000000
CLK             1
RESET           0
ENABLE          1
MODE            0
```



Register	Value
Core	
R0	0x00000101
R1	0x10000000
R2	0x00000004
R3	0x00000006
R4	0x00000002
R5	0x00000003
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x10000200
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00080000
xPSR	0x21000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	262024
Sec	0.02183533

6. CONCLUSION

In this paper we have shown the basic addition, subtraction, multiplication, and division of 64-bit numbers using VHDL code. Our code returns very less numbers of cycles and provides faster solution, which decreases the time complexity. We have used LDRD, STRD, ADC, UMULL because it takes 64-bit numbers while performing mathematical tasks. We have shown the VHDL code with the output above and our motive is to make the system faster than usual.

7. REFERENCES

[1] COMPUTER ORGANIZATION AND EMBEDDED SYSTEMS by Carl Hamacher (Queen's University), Zvonko Vranesic (University of Toronto), Safwat Zaky (University of Toronto) Naraig Manjikian (Queen's University)

[2] A Survey of RISC Architectures for Desktop, Server, and Embedded Computers by Steven Przybylskic (A Designer of the Stanford MIPS) x86 Instruction Set Architecture by Tom Shanley, Publisher: MindShare Press