**VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY**

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRICAL ENGINEERING

**Advanced Program**

---------------o0o---------------

****

**MICROCOMPUTER LAB**

**HOMEWORK 6 REPORT**

**Instructor: Nguyễn Trung Hiếu**

**Students: Lý Chí Học 1810930**

**Phạm Tấn Khải 1851081**

**Nguyễn Thanh Hiệp**

**Ho Chi Minh City, 5/11/2021**

**LIST OF CONTENTS**

Abstract…………………………………………………………………………….3

Fundamental Knowledge 3

Theory 4

Design 6

Simulation 11

Result 13

References 15

Abstract

In this report, we show how to design a 3\*3 (8 bits) matrix multiplication unit with Quartus and simulate this design on ModelSim-Intel FPGA.

Fundamental Knowledge

In this HomeWork, we have to design a Hardware solution to solve a 3x3 Matrix multiplication below:

**=**

And How we solve these matrix basically:

c1=a1b1 + a2b4 + a3b7

c2=a1b2 + a2b5 + a3b8

c3=a1b3 + a2b6 + a3b9

c4=a4b1 + a5b4 + a6b7

c5= a4b2 + a5b5 + a6b8

c6= a4b3 + a5b6 + a6b9

c7= a7b1 + a8b4 + a9b7

c8= a7b2 + a8b5+ a9b8

c9= a7b3 + a8b6 + a9b9

Then we designed a mulitiplier based on above calculation which have structure as shown below:

**MUX**

**ADDER**

**MUX**

**MUX**

**ADDER**

Theory

***Carry Lookahead Adder***

A carry-lookahead adder improves speed by reducing the amount of time required to determine carry bits. The carry-lookahead adder calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger-value bits of the adder.

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | cin | cout |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(A xor B) and Cin

A and B

From Full Adder truth Table, we have Cout = A **.** B + (A ⊕ B) **.** Cin .Carry-lookahead logic uses the concepts of generating and propagating carries.

In the case of binary addition, A+B generates if and only if both A and B are 1. If we write  G(A,B) to represent the binary predicate that is true if and only if  A+B generates, we have

G(A,B)=A. B

where  A.B is a [logical conjunction](https://en.wikipedia.org/wiki/Logical_conjunction)

In the case of binary addition, A+B propagates if and only if at least one of A or B is 1. If P(A,B) is written to represent the binary predicate that is true if and only if A+B propagates, one has

P(A,B)=A+B

where A+B on the right-hand side of the equation is a [logical disjunction](https://en.wikipedia.org/wiki/Logical_disjunction)

Hence, the Boolean equation can be rewritten as Cout = G + P . Cin. Expanding with any FA blocks, the general equation becomes Cn=Gn+Pn . Cn-1.

With n=1, we have C1 = G1 + P1 . C0

With n=2, we have C2 = G2 + P2 **.** C1 =G2 + P2 **. (**G1 + P1 **.** C0)

With n=2, we have C3 = G3 + P3 **.** C2 = G3 + P3 **. [**G2 + P2 **. (**G1 + P1 **.** C0)]

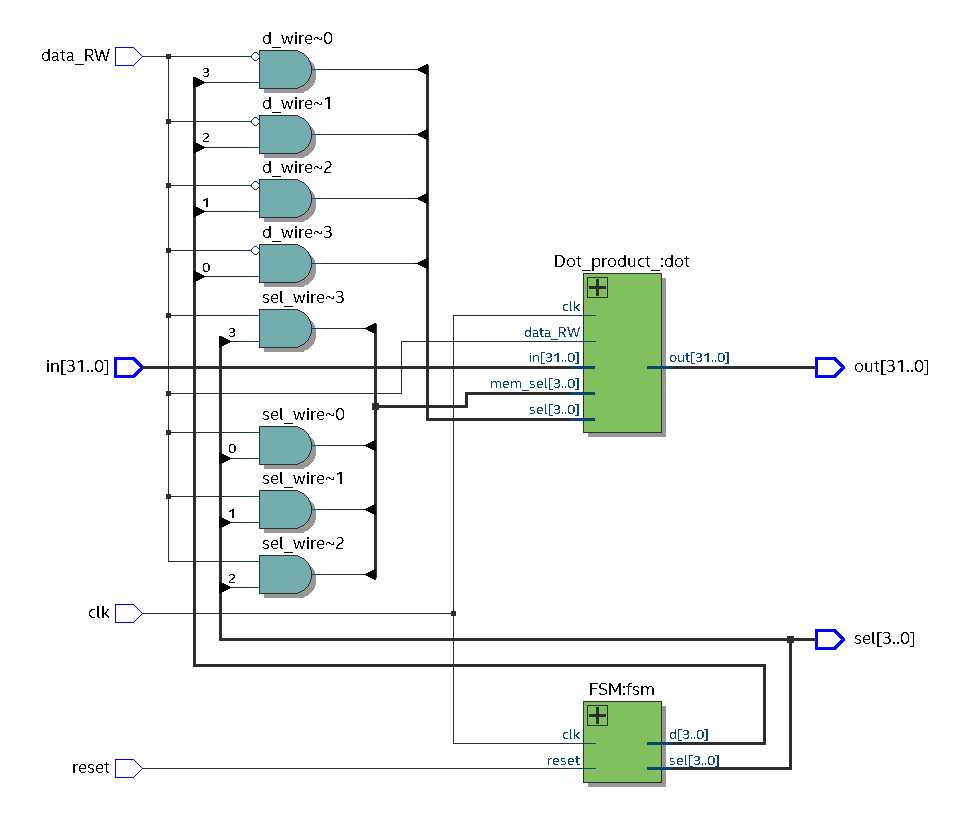
As we can see, reducing the propagation delay time is that Cn does not need to wait the previous Cn-1 and the delay time is the same for any carry bit => adding operation is faster and more effective.

Design

1. **Top deisgn.**

Consist of:

* Dot product unit.
* The Finite State Machine (FSM) unit.

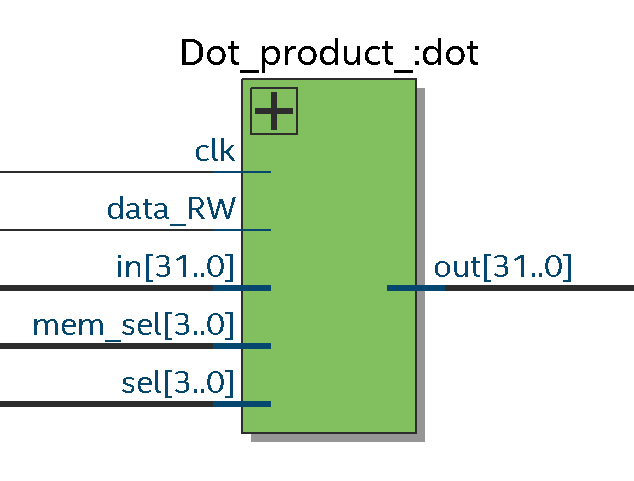


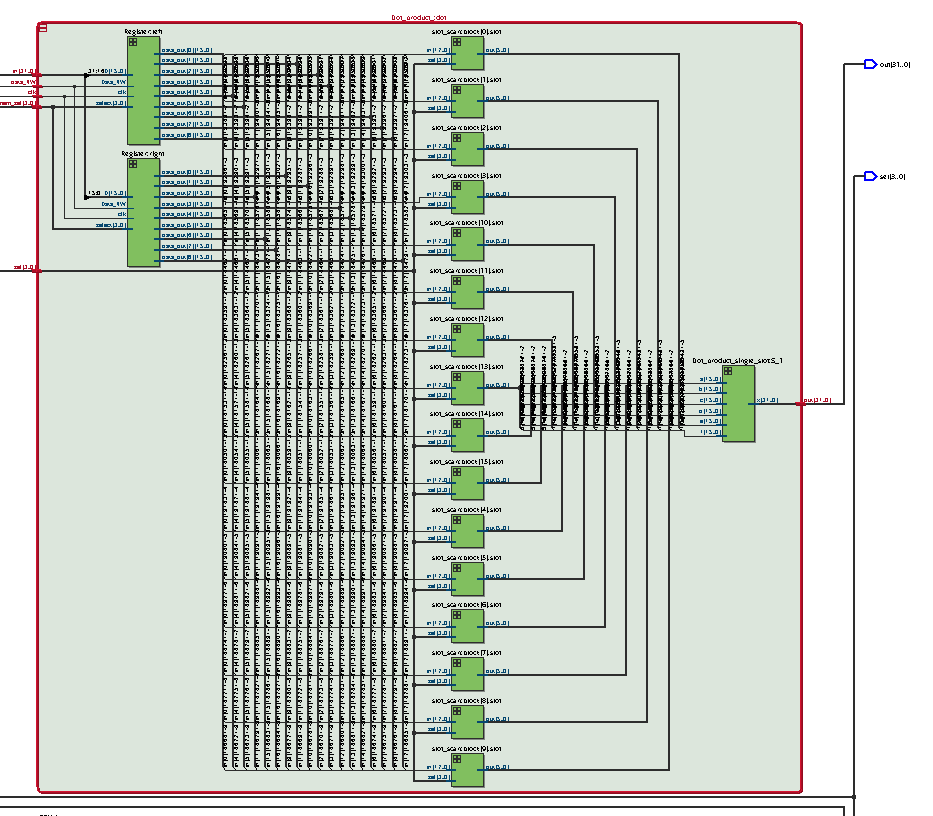


1. **Internal parts.**

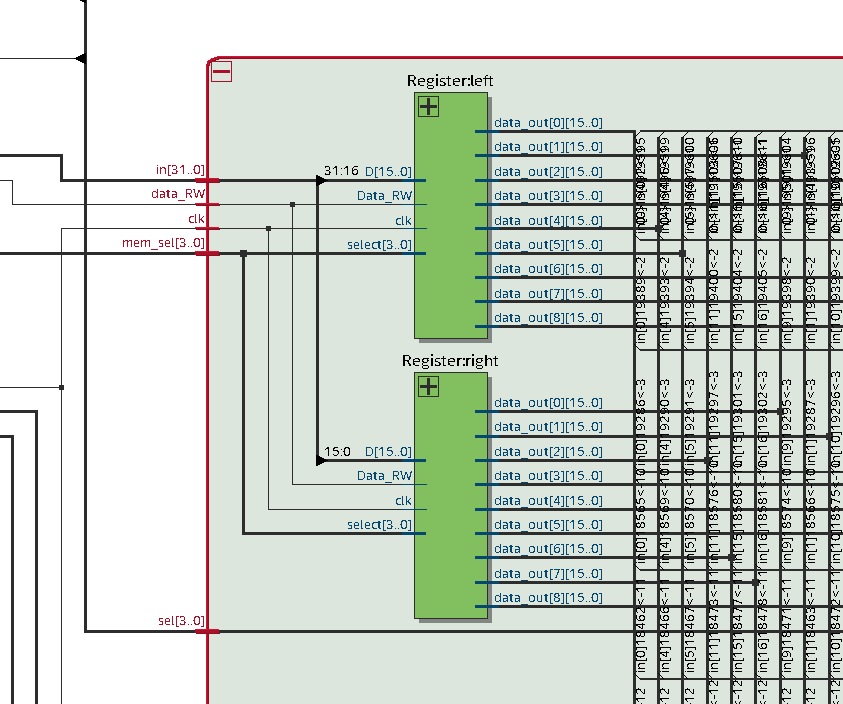
**Dot product unit:**

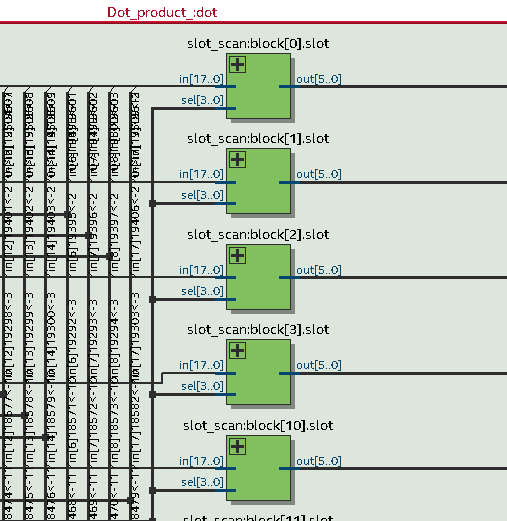
Consist of 3 main components: register memories, row and column of matrix selector

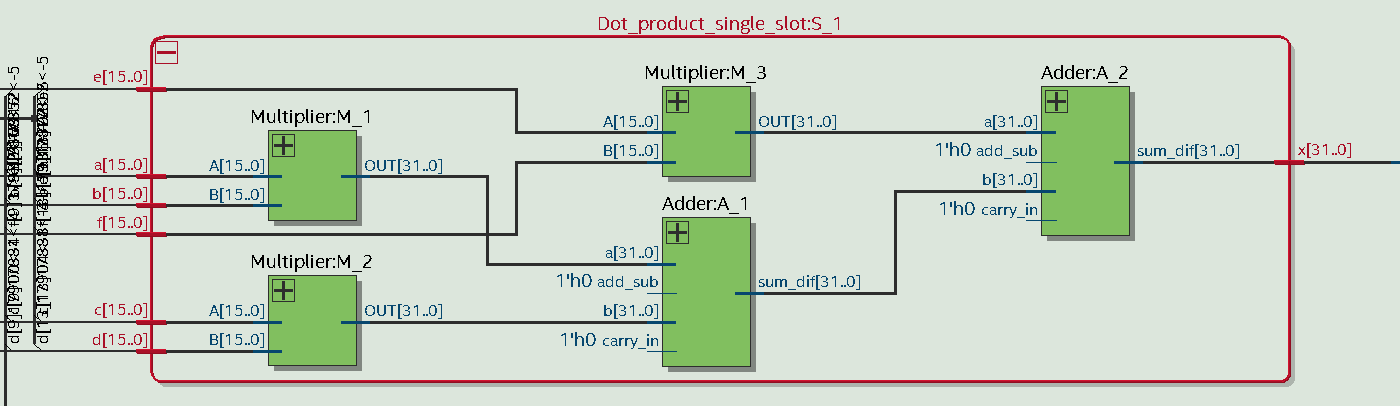




Consist of **2 memory register** 16x16 to get 18 inputs from two matrix 3x3, 18 **input row and column selector** (3x3 to 3x1 mux).a dot product calculator comprised of 16bits **multiplier** and 32bits **adder** all of them are unsigned.

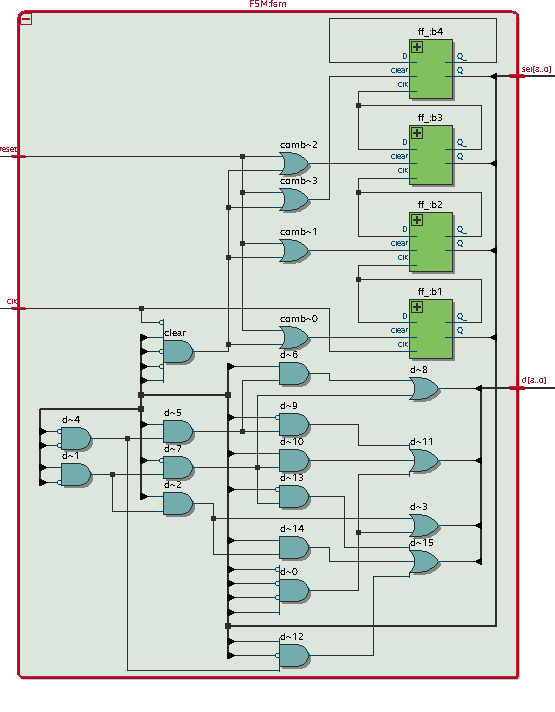
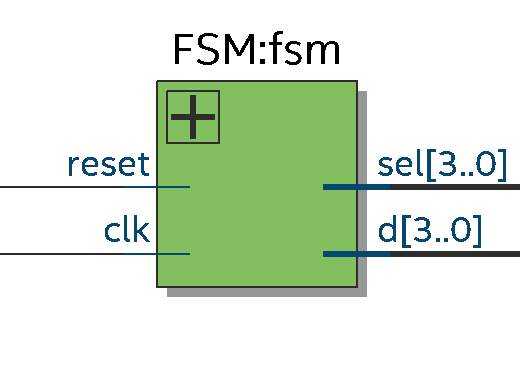






**FSM:**

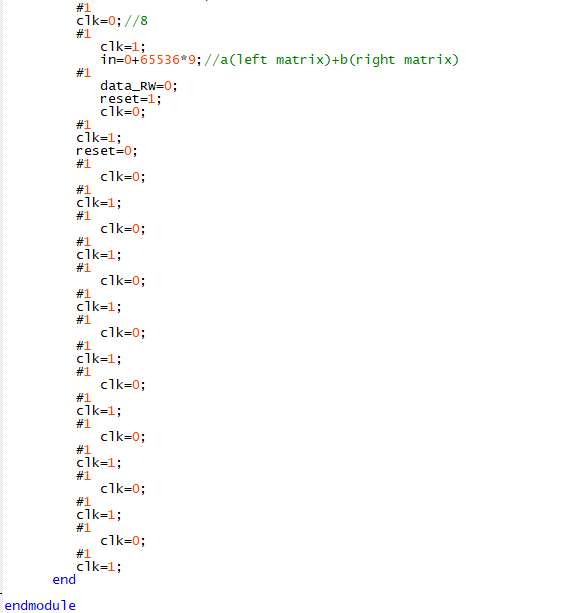
It simply a **counter** from 0 to 9 in order to run a sequence to switch address of the two **registers memory** in the dot product unit that load input into the matrix calculation module, and also another sequence made by the **FSM unit** to calculate single slot of the output matrix and put it in the **register memory at the end of output.**



Simulation

Below is the testbench that have clock, reset signal, selection outputs, read and Write enabler, input, and register memory to save the computed matrix:

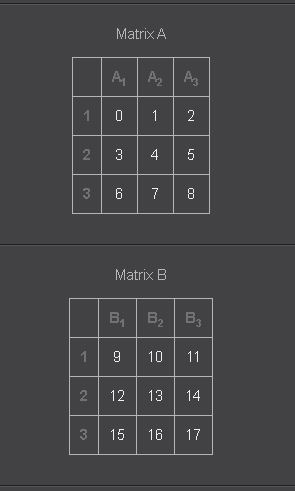
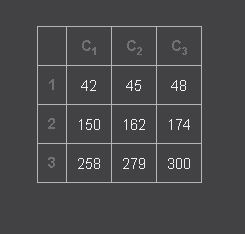




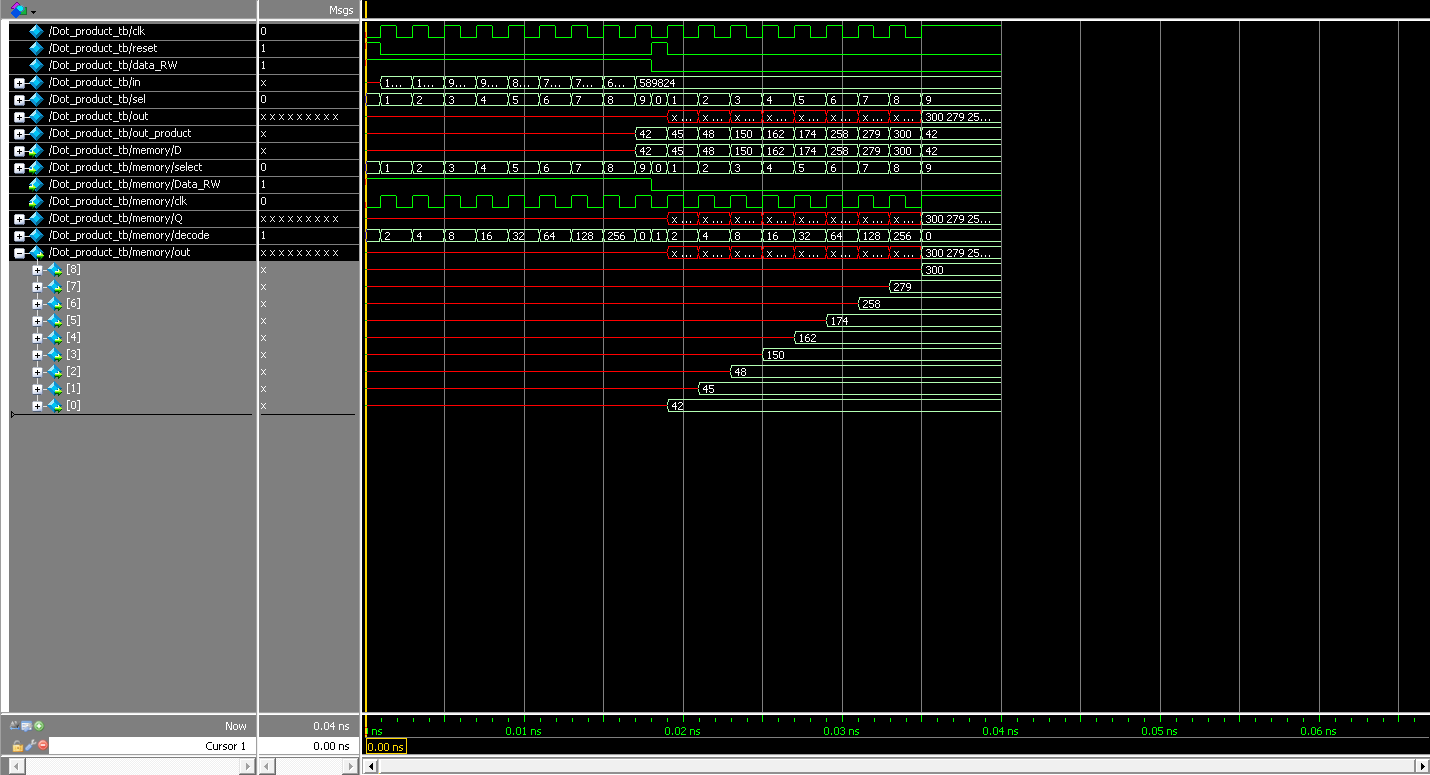
Result

***The result of the dot product of two 3x3 matrices:***

we use the two matrix below to use as an example for demonstrating the dot product calculation.

***Modelsim simulation:***



References

1. Wikipedia, Carry-lookahead adder, <https://en.wikipedia.org/wiki/Carry-lookahead_adder>
2. Ques10,https://www.ques10.com/p/40462/explain-array-multiplier-2/?fbclid=IwAR1c7YzkJHy9gjyEuXzLxSkP9cKcgKJLM9Oy7BYsDA14MjS9u\_qFbjGBtvA