Quick evaluation of technical skills and research aptitude - Answer

AmirrezaGh

***Abstract*— In this report, one method is proposed to each problem described in the Quick evaluation of technical skills and research aptitude documents, minimum hamming distance and GCD VHDL hardware. Also, the methods are evaluated against previous works.**

***Keywords***— **Hamming distance – GCD hardware**

1. Introduction

As it was mentioned before, in this report, minimum hamming distance is studied, and minimum hammering distance of the elements of an array is calculated in C++ language. Also, the RTL hardware of the Euclidean GCD algorithm is written in VHDL.

1. Min Hamming Distance

Hamming distance is a concept used in information theory and computer science to measure the difference between two strings of equal length. It quantifies the minimum number of substitutions required to change one string into another by flipping individual bits.

It has various applications in areas such as error detection and correction, data clustering, cryptography, and DNA sequence analysis.

To calculate the Hamming distance, you compare each corresponding pair of bits in the two strings and count the number of positions where they differ. This count represents the Hamming distance between the strings. The strings must be of the same length for a valid comparison.

To calculate the minimum hamming distance in an array of unsigned integer values, there are two famous approaches. First use shift operator when the number of ones is calculated.

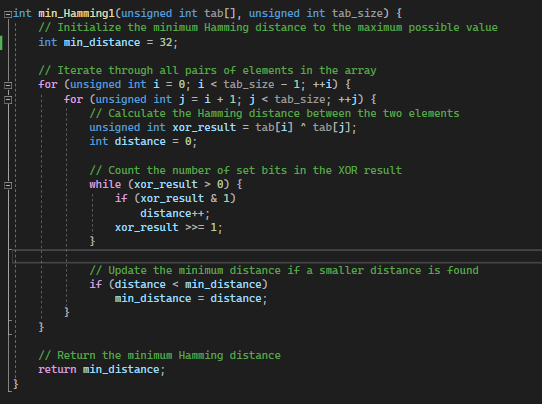


Fig. 1 Min Hamming Method 1

The second approach is to use hamming weight in order to count the number of dissimilarities between two elements.

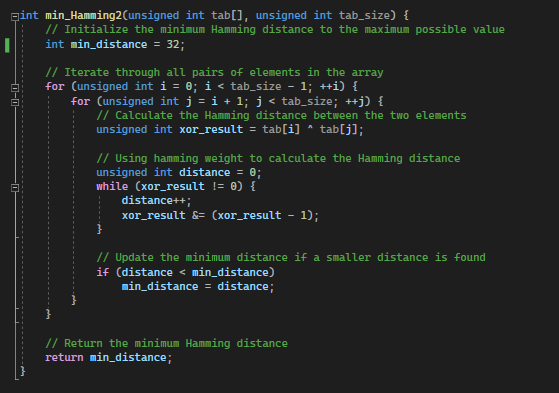


Fig. 2 Min Hamming Method 2

The proposed method is to terminate the dissimilarity count loop early using the temporary minimum distance is less than the computed distance until the current state of calculation of dissimilarities between the two elements. Also, if the minimum distance is zero, it terminates the process and returns zero.

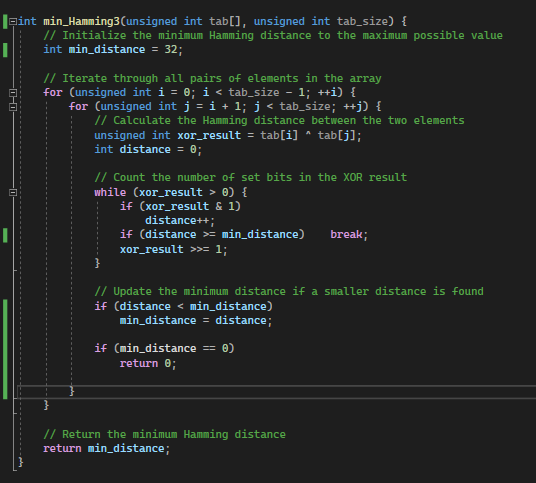


Fig. 3 Min Hamming Proposed Method

The result obtained from running three methods on a table of 10000 integer numbers are shown below.

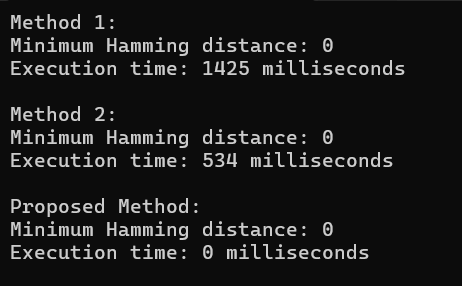


Fig. 4 Evaluation of 3 methods

Because of early termination of the whole function using the condition whether minimum is zero or not, the execution time gets very small. But in order to show the impact of early termination in count loop, the condition gets commented. So, the result is shown below.

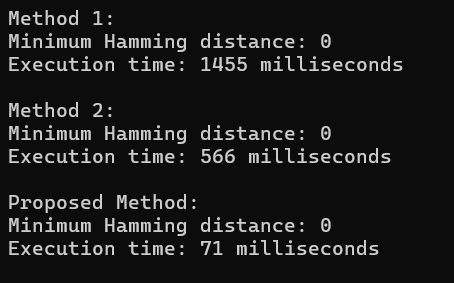


Fig. 5 Evaluation of 3 methods without the condition on minimum distance

1. GCD VHDL Code

In this report, the Euclidean algorithm is used to compute greatest common divisor (GCD). The algorithm is based on the principle that the GCD of two numbers does not change if the smaller number is subtracted from the larger number repeatedly until the two numbers become equal. At this point, the common value is the GCD of the original two numbers.

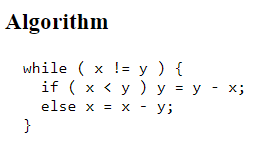


Fig. 6 Euclidean Algorithm

First of All, the following behavioural finite state machine is implemented in the previous works in order to show the correctness of the algorithm and behaviour of the next implementations.

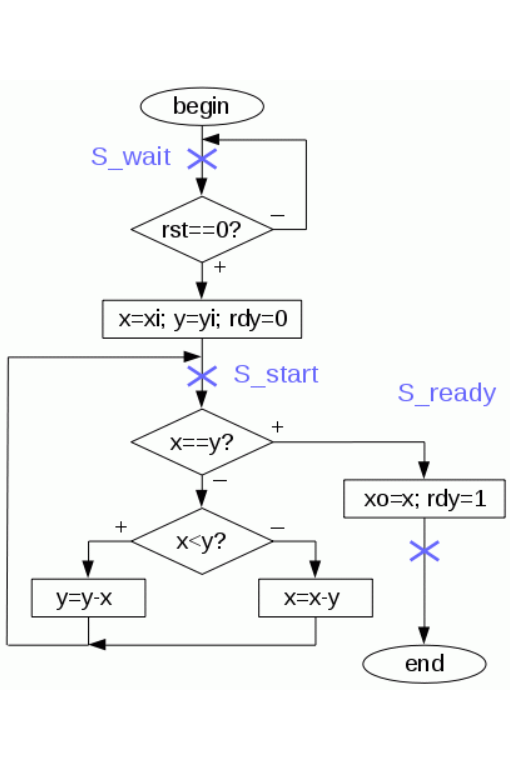


Fig. 7 behavioural FSM

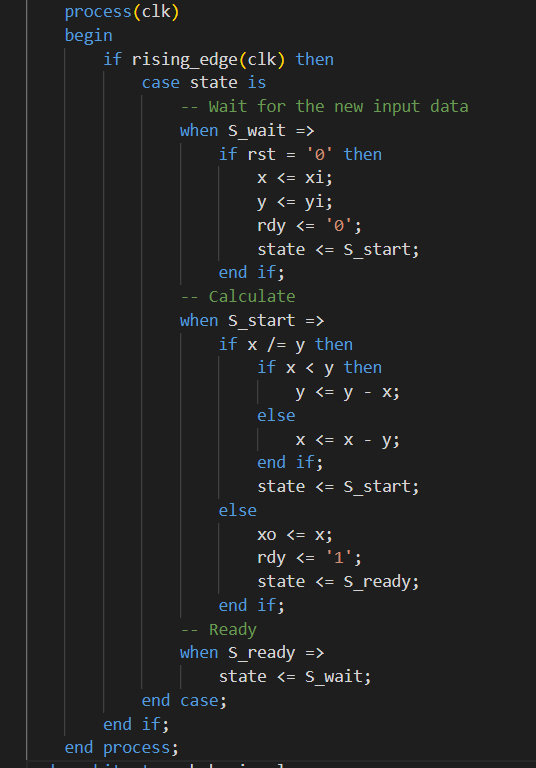


Fig. 8 Behavioural FSM code

The first method of RTL implementation of GCD uses operation reuse using only one ALU for all operations, such as subtraction and comparison. The DataPath is shown below.

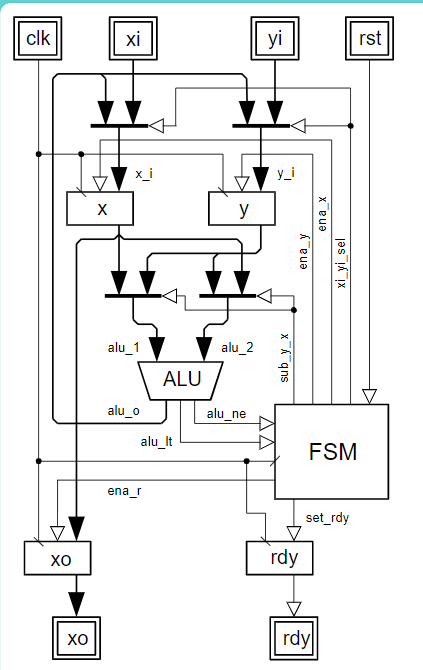


Fig. 9 DataPath of RTL method 1

The most simple FSM for the DataPath is as follows.

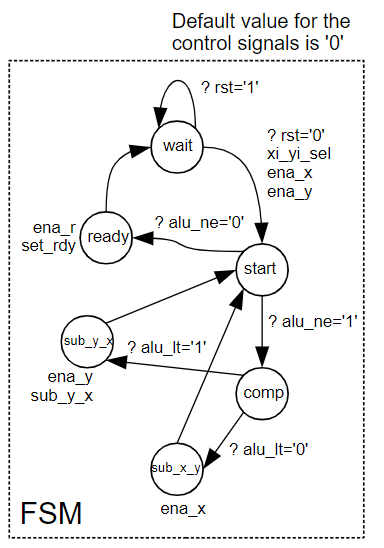


Fig. 10 FSM 1 of RTL method 1

Then the better FSM with less states and clocks is as follows using mealy state machine

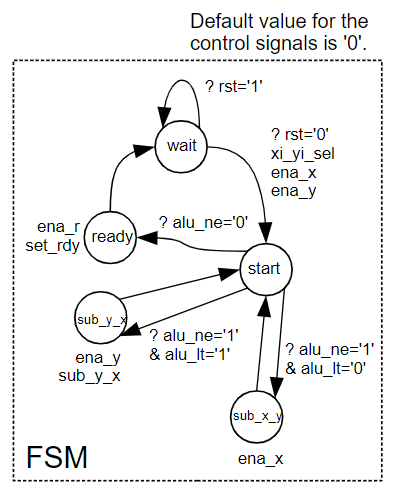


Fig. 11 FSM 2 of RTL method 1

The second method is using a separate comparator besides ALU in order to achieve better speed.

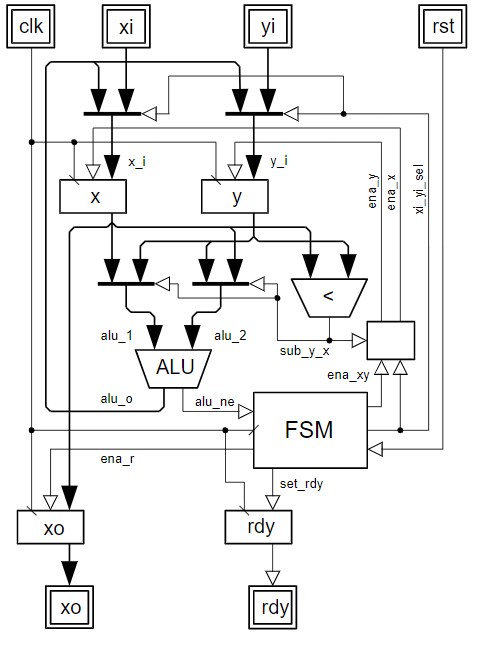


Fig. 12 DataPath of RTL method 2

The third method is using a separate subtractor besides ALU in order to achieve better speed.

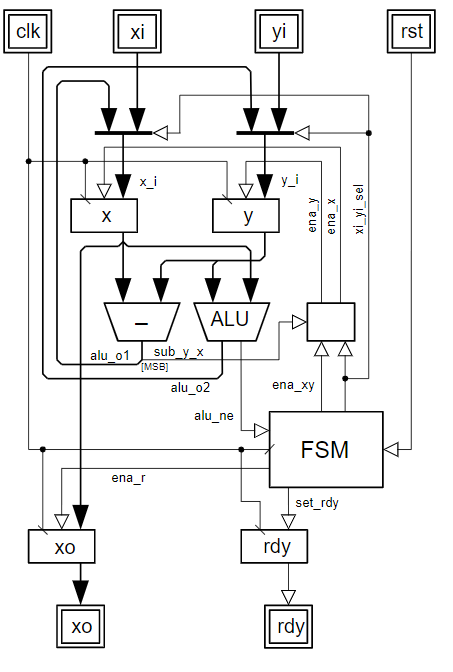


Fig. 13 DataPath of RTL method 3

The forth method in previous works is using two subtractor and a comparator instead of ALU.

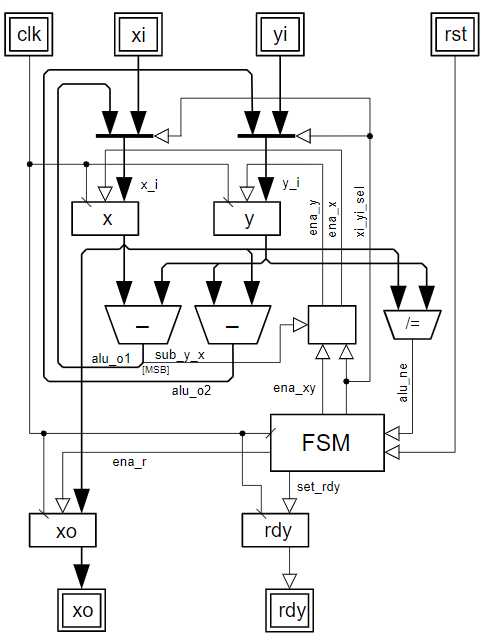


Fig. 14 DataPath of RTL method 4

The proposed method is using less-than, not-equal comparators and two subtractor in order to maximize the speed.

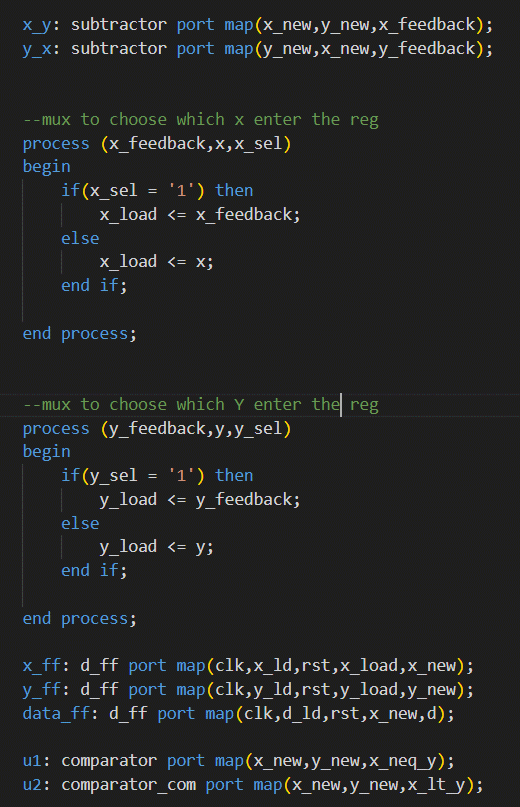
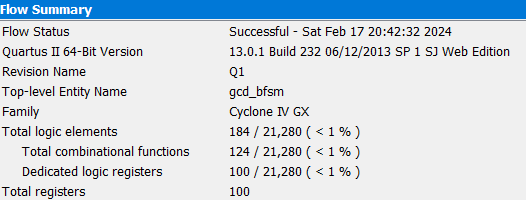


Fig. 15 Proposed Method code

The Evaluation Results of the above methods using Quartus software to implement on Cyclone IV GX are as follows.



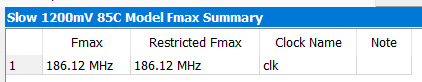
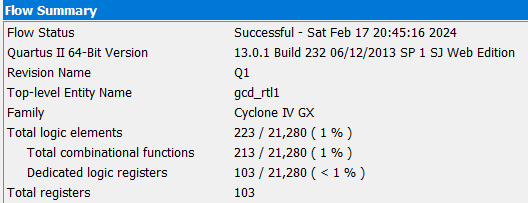


Fig. 16 Synthesis Result of behavioural FSM



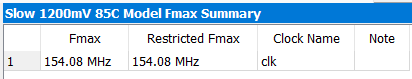
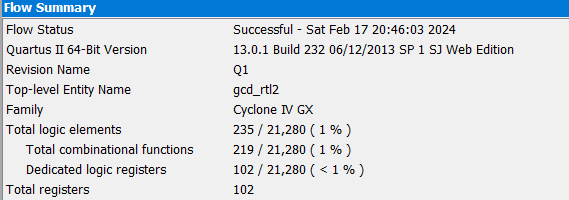


Fig. 17 Synthesis Result of second method with first FSM



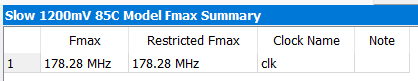
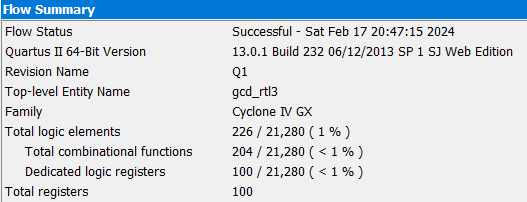


Fig. 18 Synthesis Result of second method with the second FSM



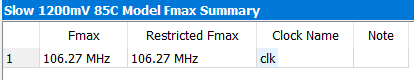
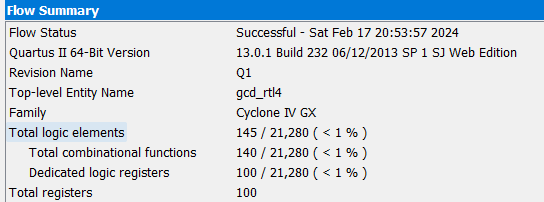


Fig. 19 Synthesis Result of third method with first FSM



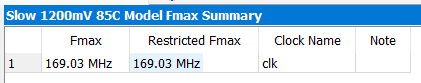
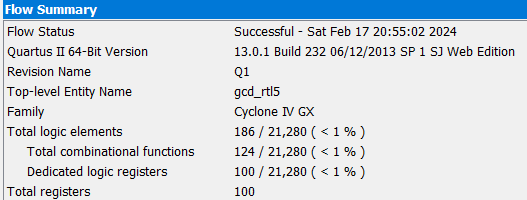


Fig. 20 Synthesis Result of third method



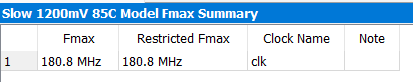


Fig. 21 Synthesis Result of forth method

The result of calculation of GCD(14, 161) using the proposed method is as follows.

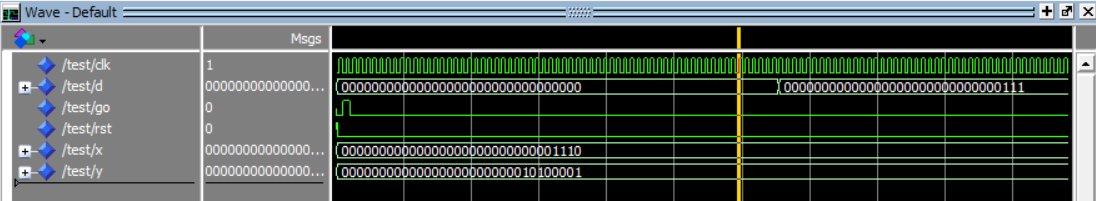
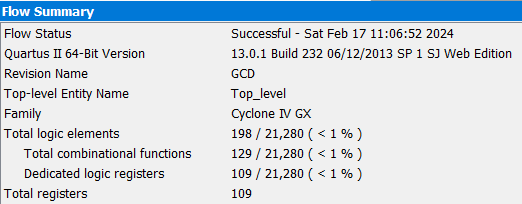


Fig. 22 GCD Result of proposed method



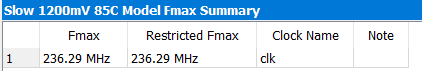


Fig. 23 Synthesis Result of proposed method

1. Conclusion

To put it in a nutshell, in the first problem, the evaluations are as follows.

Tab. 1 Conclusion of Problem 1

|  |  |
| --- | --- |
| Method | Execution time(ms) |
| Simple method | 1455 |
| Weight hamming | 566 |
| Proposed method | 71 |

In the second problem, the following table concludes every method result.

Tab. 2 Conclusion of Problem 1

|  |  |  |
| --- | --- | --- |
| Method | Area(Logic Block) | Fmax(MHz) |
| Behavioural | 184 | 186 |
| ALU1 | 223 | 154 |
| ALU2 | 235 | 178 |
| ALU + comp | 226 | 106 |
| ALU + sub | 145 | 169 |
| 2Sub + comp | 186 | 180 |
| 2 Sub + 2 comp (Proposed Method) | 198 | 236 |

Therefore, the best method in terms of area is the forth method (2 subtractor + 1 comparator), and the best method in terms of speed is the proposed method (2 subtractor + 2 comparator).

References:

* [ihabadly/FSMD-GCD: Basic FSMD example to calculate the Greatest Common Divisor of two 8-bit numbers (github.com)](https://github.com/ihabadly/FSMD-GCD)