Shortening of processing time of optimal design of multiple constant multiplication using FPGAs

Masao Nakayama[1], Takao Sasaki[1] and Hisamichi Toyoshima[1]
[1]Faculty of Engineering, Kanagawa University
3-27-1 Rokkakubashi, Kanagawa-ku, Yokohama, Kanagawa 221-8686, Japan
E-mail : [1]nakayama@tysm.ee.kanagawa-u.ac.jp

Abstract: Problem of designing multiple constant multiplication (MCM) circuits with minimum cost is known to be an NP-complete problem. Several techniques using combinatorial optimization algorithms such as genetic algorithms (GAs), etc., have been proposed. However, if implemented as software, as the circuit scale increases, a great deal of time is needed for optimization. The purpose of this study is to shorten the time spent on optimizing MCM circuit design. We propose a hardware-oriented algorithm suitable for FPGAs for both circuit synthesis and optimization.

1. Introduction

A multiple constant multiplication (MCM) circuit is designed to multiply a single input by multiple constants, and is often used in signal and image data processing. For synthesis of MCM circuit, it is important to minimize circuit scale and delay time. The circuit scale can be reduced by replacing each multiplier by shifts and additions and then sharing them; however, as the number of constants or word length increases, the combination of shared patterns becomes huge. Designing MCM circuits at minimum cost is therefore an NP-complete problem. Several techniques using combinatorial optimization algorithms such as genetic algorithms (GAs), etc., have been proposed. However, if implemented as software, as the circuit scale increases, a great deal of time is needed for optimization. The purpose of this study is to shorten the time spent on optimizing MCM circuit design. We propose a hardware-oriented algorithm suitable for FPGAs for both circuit synthesis and optimization.

2. Optimal design of MCM circuit

An MCM circuit is described by the next equation.

$$y_i = a_i x \quad (i = 0, 1, 2, \cdots N-1) \tag{1}$$

where x is an input, and each a_i is a coefficient.

For synthesizing MCM circuits, several techniques have been proposed for reducing the cost of circuit by replacing multipliers with shifts and adders and sharing them. In the optimization algorithm, the generation and evaluation of solutions are processed by means of a great number of iterations. Sequential processing is also required for synthesizing MCM circuits. It is therefore imperative to shorten the iterative processing cycle when implementing an optimization algorithm. However, if implemented as software, optimization becomes time-consuming, due to sequential processing in software. For realistically optimizing the design of MCMs, a shortened processing time is required that is far shorter than that taken by software.

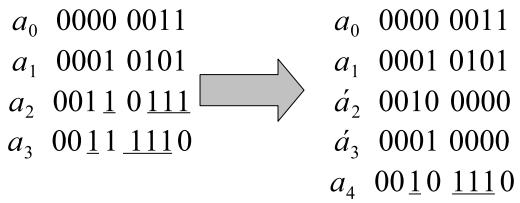

Figure 1. Sharing of coefficient by binary representation

3. Proposed method

To solve problems with software implementation, a circuit synthesis technique for circuit design is proposed, and the optimization algorithm is changed to an hardware algorithm and implemented on FPGAs. The proposed algorithm contains circuit synthesis method and optimization method, shown as follows.

3.1 Circuit synthesis method

The MCM circuit synthesis technique can be categorized into the two methods listed below.

- Binary representation(IMA[1],HCA)
- Directed acyclic graph(BHA,BHM)

We adopt the technique of binary representation in this study, since bit operation is suited to hardware implementation. As an example of one technique using binary representation, the coefficient set $\{a_i\} = \{3, 21, 55, 62\}$ is given. Figure 1 shows the method of sharing each coefficient.

In case that the coefficients are not shared, 11 adders are required. On the other hand, $\{a_4 = (a_2 << 1)\ and\ a_3\}$ can be made by sharing a_2 and a_3. By expressing as $\{a_2 = (a_4 >> 1) + \acute{a}_2, a_3 = a_4 + \acute{a}_3\}$, the number of adders can be reduced to eight.

3.2 Hardware implementation of circuit synthesis method

Figure 2 shows the configuration of the hardware algorithm based on binary representation. In this configuration, the initial-state coefficients are externally transferred to the coefficient-list RAM via a shared module. Then, with a command to start the sharing process, the shared module transfers the coefficients to the search module by serial transfer. The search module searches for candidates that can be shared at high speed, using a systolic array, and sends the candidates (sharing candidates) with the largest number of bits that can be shared to the shared module. In the shared module, the coefficients are shared based on the sharing candidates, and the

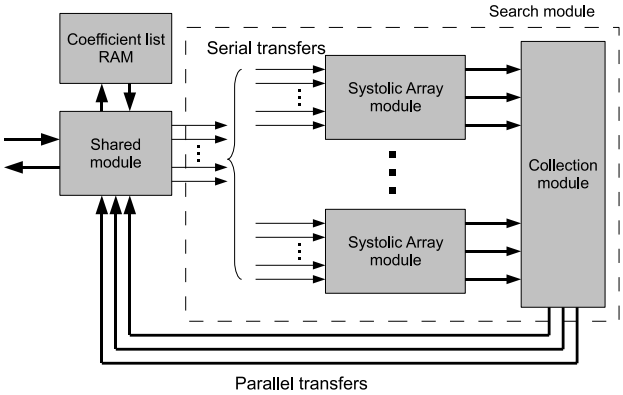

Figure 2. Framework for hardware implementation of sharing of coefficient by binary representation

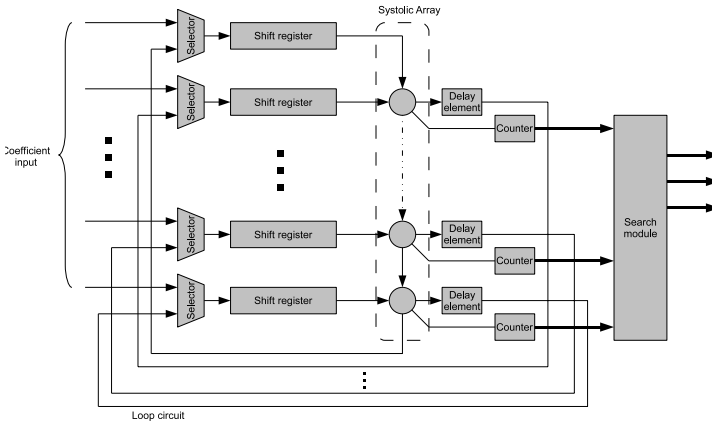

Figure 3. Loop structure of systolic array

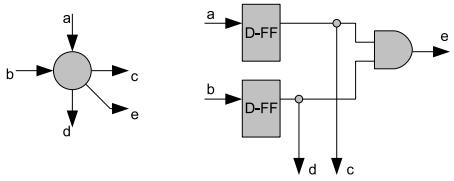

Figure 4. Structure of Processing Element

result of the coefficient-sharing process is applied to the coefficient list RAM. These steps are repeated either until there are no more sharing candidates or until the coefficient-list RAM is full.

The implementation method of each module is described below.

- Search module

There are numerous hardware-based techniques to reduce operation time. In this study, systolic arrays are used. A systolic array is achieved by regularly arranging processing elements (PEs) that perform a simple calculation and connecting only adjacent processing elements. A large amount of data can thus be processed at high speed by parallel and pipeline processing. In addition, this configuration is similar to that of an island-style FPGA where programmable logic blocks are regularly arranged and wires are installed between the blocks, making it suitable for FPGAs. As shown in Figure 3, a configuration equivalent to the bit shift of coefficients can be achieved by connecting the output of a systolic array via a delay element. In other words, the search range can be expanded while minimizing modifications to the circuitry. Due to the simple configuration of the PE, illustrated in Figure 4, a drop in processing speed is unlikely to occur. In addition, the use of systolic arrays minimizes circuit modifications when circuit expansion is needed.

- Transfer method

Transfer methods can be either parallel or serial. Parallel transfer can transfer more data per clock cycle than serial transfer. However, for parallel transfer, synchronization with the clock tends to become difficult at higher clock frequencies. In serial transfer, the amount of data transferable per clock cycle is lower than in parallel transfer, but synchronization with the clock is easier, allowing operation at higher frequencies. For this reason, serial transfer is used, since it is faster than parallel transfer when taking into account the maximum operating frequency and the transfer clock cycle during FPGA implementation.

3.3 Optimization algorithm

Combinatorial optimization algorithms are categorized into multipoint search and single point search types. In this study, a genetic algorithm (GA), a multipoint search algorithm, is employed. GAs can be implemented in parallel, and have the potential to shorten the processing time. A gene representation is shown in Figure 5. Each gene is composed of a pair of shared coefficient indices plus a bit shift coefficient.

As one example of the coefficient sharing shown in Figure 1 , an intermediate coefficient a_4 is generated by one bit shifted to a_2 and a_3. This operation is represented as one gene with i = 2, j = 3 and s = 1.

3.4 Hardware implementation of GA

Figure 6 shows the configuration of the hardware algorithm for the GA. In this configuration, initial-state coefficients are first externally transferred to the coefficient-list RAM. Then, with a command to start the sharing process, the GA main module (GMM) generates an initial solution and transfers it to the population RAM. This forms a population. After generation of the initial solution is complete, the GMM transfers chromosomes from population one after another to the MCM circuit synthesis module (MCSM). In the MCSM, the sharing process is first performed according to the chromosome, and then a search for shared coefficients is performed. After completion of the sharing process in the MCSM, the coefficient list is transferred to the GMM. In the GMM, the required number of gates is counted, and this is set to the fitness of the

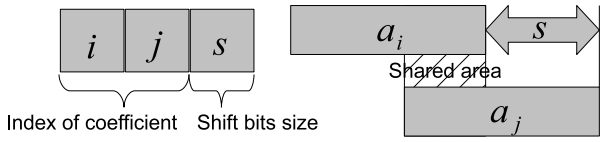

Figure 5. Structure of chromosome

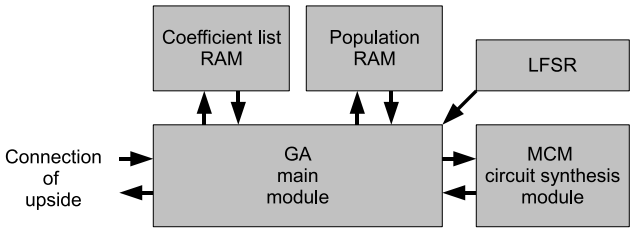

Figure 6. Framework for hardware implementation of GA

initial solution. With the above steps, calculation of fitness of the initial solution is complete. The GMM then randomly selects two chromosomes and transfers them to the MCSM after crossover and mutation. If the fitness of the generated individual is greater than that of the parent, a crossover operation is performed. The above operations are repeated for a number of times determined by the number of generations and the crossover rate.

The implementation method used to change GA to the hardware algorithm is described below.

- Selection model

In the selection and crossover operation of a GA, each chromosome is sorted in order of fitness and saved in the memory, causing an increase in the number of registers and memory required in the case of hardware implementation. In this study, an alternative evolution model called ER (Elite Recombination[5]), suitable for hardware implementation, is used.

In ER, two chromosomes are selected randomly, and using crossovers, four chromosomes are generated. The two chromosomes with the greater fitness are left in the next generation. This method requires no sorting and can reduce demand for memory and registers. ER is better capable of converging a solution and performing a search than other generation alternative models.

- Random numbers generation

M-sequence random number, which can be configured with shift registers and XOR and achieved with small circuit sizes, is used. One random number can be generated by one XOR operation, and thus the random number generation speed is very fast. This method is commonly used in hardware implementation and is called a linear feedback shift register (LFSR). In this study, a 13-bit LFSR was used.

- Crossover

The random number generated by the LFSR is 0 or 1. It is then saved to a shift register for the length of the chromosome.

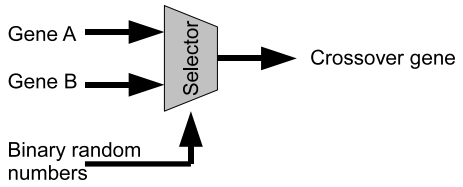

Figure 7. Crossover circuit

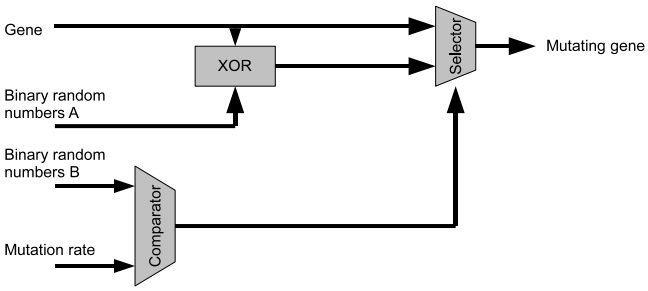

Figure 8. Mutation circuit

For the chromosome generated, the saved random number determines whether either chromosome A or B is passed to the bit unit. As shown in Figure 7, a selector is used for this operation. The use of a selector achieves uniform crossover with one operation, and therefore considerably speeds up the crossover operation.

- Mutation

Mutation rate and random value are converted to integer values. When the random value becomes smaller than the mutation rate, mutation is performed. The mutation rate is converted to an integer value by multiplication by a constant, R. The random value is converted to an integer value by a RB register with the number of bits required for describing the constant R. In this study, R was set to 1024. Figure 8 shows a mutation operation. As with the crossover circuit, the mutation circuit can perform mutation with one operation, and thus the mutation speed is very fast.

- Evaluation

The number of adders in the circuit generated as fitness is counted. The synthesis result sent by the MCSM is information on coefficients after the sharing process and which coefficients are shared. Upon reception of this information, the number of adders required for each coefficient is counted. In this study, the number was set to [(the number of 1s in the binary representing the coefficient) - 1]. This sum is the required number of adders for the coefficient. The number of pieces of information on shared coefficients is then added to the sum. The thus generated number of adders of the MCM circuit is counted.

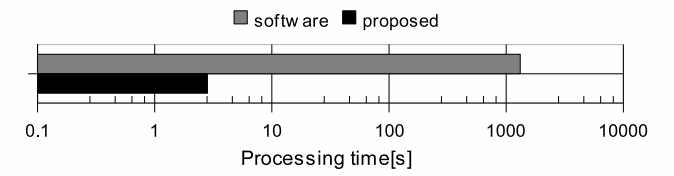

Figure 9. Speed comparison

- Initial solution generation

An initial solution is generated by the GMM using a random value. Each chromosome is composed of the indexes for a pair of shared coefficients and amount of bit shift of the coefficients. The generated chromosome and the coefficients are transferred to the MCSM to calculate fitness. The calculated fitness is saved to the population RAM. These are the steps required to generate an initial solution.

4. Evaluation

To evaluate the software implementation and the proposed method, the following environment and development environment(DE) are used. Since software implementation and the proposed method use the same search technique, they have comparable ability in searching for solutions.

- FPGA Xilinx VirtexII Pro 33088Slices 66MHz
  (DE:Xilinx ISE Foundation 8.2i)

- PC WindowsXP Xeon 3.4GHz Memory:1GB
  (DE:Microsoft Visual C++ 2005)

The number of coefficients is 50. The bit length of the coefficient is 9bit. The parameter of GA is as follows. The gene length is 48bit. The number of generations is 1000. The number of individuals is 100. The crossover rate is 0.8. The mutation rate is 0.08.

Figure 9 shows the result of the speed comparison. As seen from this figure, the proposed hardware implementation has been executed 460 times faster than in software.

5. Conclusions

This study proposed a hardware algorithm for a high-speed method of synthesizing MCM circuits. We focused on the MCM circuit synthesis techniques using binary representation and achieved systolic array-based implementation. In addition, for GA implementation, a reduction in circuit size and increased speed was achieved using ER, a generation alternative model suitable for hardware algorithm.

The above proposal has made it possible to achieve processing up to 460 times faster than software implementation. As the next step in this study, for the effective use of multipoint searches, a search circuit will be implemented on several FPGAs and the processing speed will be increased using parallel searches. The rate of processing speed improvement by parallel search and the pattern of increase of circuit size will also be investigated.

acknowledgment

This work is supported in part by “High-Tech Research Center Project” from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, “Multiple constant multiplication: Efficient and versatile framework and algorithms for exploring common subexpression elimination”, IEEE Trans. Computer-Aided Des. Integrated Circuits & Syst., vol.15, no.2, pp.151-165, Feb. 1996.

[2] A.Matsuura, M.Yukishita, and A.Nagoya, “A Hierarchical Method for the Multiple Constant Multiplication Problem“, IEICE Trans. Fundamentals vol. E80-A, No.10, pp.1767-1773, Oct. 1997.

[3] David Bull, David Horrocks, “Primitive operator digital flters“, IEE Proceedings-g Vol.138, pp.401-411, June 1991.

[4] Hiroshi Umeo, “Systolic Array”, Infomation Processing Society of Japan, Vol.30, No.1, (1988)

[5] Thierens, D. and Goldberg, D.E.: Elitist Recombination: an integrated selection recombination GA, Proceedings of the First IEEE Conference on Evolutionary Computation, pp.508-512(1994)