In the context of accelerating computation, many parallel and distributed computing methods are proposed by the researchers. To accelerate computation, parallel computing method with a multi-core and/or multi-processor computer and distributed computing method using a computer cluster are widely used. On the other hand, FPGAs (Field Programmable Gate Arrays) are a programmable VLSI which can be used for implementing parallel and hardware algorithms. As a device for low cost pseudo-specialized LSI, FPGAs are attracting attention to the researchers. With the development of LSI device fabrication, FPGAs are chosen by the researchers for implementing their applications. For the technical improvement of LSI production, their size, capabilities, and speed have been increased. Compared with parallel and distributed computing, the computation granularity of computation using FPGAs would be the finest. Considering programmability, FPGAs can be considered as hardware that has an ability of software. Recently, FPGAs are widely used to implement algorithms with circuits for accelerating computation. Circuit design that minimizes the number of clock cycles is easy if we use asynchronous read operation. However, embedded blocks of memories in the most modern FPGAs support synchronous read and synchronous write operations, but do not support asynchronous read operation. Hence, we can not implement circuits with memories supporting asynchronous read operation in FPGAs. Because of the above background, we are inspired to present circuit rewriting algorithms to convert circuits with memories supporting asynchronous read operation into the equivalent circuits with memories supporting synchronous read operation for implementing in FPGAs. For this purpose, this dissertation shows circuit rewriting algorithms which are as follows:

A Circuit Rewriting Algorithm for Converting Asynchronous ROMs into Synchronous Ones for FPGAs

A *circuit rewriting algorithm* that is used to rewrite a given circuit with AROMs (Asynchronous Read Only Memories) until an equivalent circuit with SROMs (Synchronous Read Only Memories) is generated for implementing in FPGAs. More specifically, a circuit, X with AROMs is given. Our *circuit rewriting algorithm* generates a circuit, Y with SROMs which is an equivalent to X with AROMs for implementing in the current FPGAs. FPGAs have Configurable Logic Blocks (CLBs) to implement combinational and sequential circuits and block RAMs to implement Random Access Memories (RAMs) and Read Only Memories (ROMs). Circuit design that minimizes the number of clock cycles is easy if we use asynchronous read operation. However, embedded memories of the most FPGAs support synchronous read and synchronous write
operations, but do not support asynchronous read operation. Hence, the main contribution of this chapter is to present a potent circuit rewriting approach to resolve this problem. We assume that a circuit using asynchronous ROMs (AROMs) designed by a non-expert or quickly designed by an expert is given. Our goal is to convert this circuit with asynchronous ROMs into an equivalent circuit with synchronous ones (SROMs) automatically. Finally, the resulting circuit with synchronous ROMs can be embedded into FPGAs. We briefly discuss the techniques to improve performance of the AROM-free resulting circuit and also describe a technique for applying our rewriting algorithm even if a user designs a circuit with pipeline structure.

A Circuit Rewriting Algorithm to Obtain Circuits Synchronous RAMs for FPGAs

A circuit rewriting algorithm, presented in this chapter is devoted for converting a circuit with RAMs supporting asynchronous read and synchronous write operations (ARAMs) into an equivalent circuit with RAMs supporting synchronous read and synchronous write operations (SRAMs); more specifically, a circuit using asynchronous RAMs (ARAMs) designed by a non-expert or quickly designed by an expert is given. This rewriting algorithm converts it into an equivalent circuit using synchronous RAMs (SRAMs) for implementing in FPGAs. In our previous work, mentioned earlier (followed by the Chapter 3), we considered only read operation of the memory blocks (ROMs). Particularly, presented circuit rewriting algorithm was used to convert a circuit with AROMs into an equivalent circuit with SROMs. The resulting circuit can be embedded into FPGAs. However, this circuit rewriting algorithm, presented in this chapter considers both read and write operations of the memory blocks (RAMs). In fact, we improved our previous research work, where RAMs can be used as the additional circuit elements to the given input circuits. The conversion of a sequential circuit with ARAMs into an equivalent fully synchronous circuit with no ARAMs for supporting the modern FPGA architecture is not trivial. However, our algorithm can do it automatically. We also briefly discuss the techniques to improve performance of the ARAM-free resulting circuit.

A Modified Circuit Rewriting Algorithm for the Circuits with Cycles

A modified circuit rewriting algorithm is used to convert a circuit with cycles using AROMs into an equivalent circuit using SROMs for implementing in FPGAs. The main contribution of this chapter is to consider a given circuit with cycles using AROMs. Particularly, our new circuit rewriting algorithm can be used to convert circuits which have cycles. In our previous works, mentioned above (followed by the Chapter 3 and Chapter 4), we have presented circuit rewriting algorithms to convert a circuit with asynchronous ROMs or asynchronous RAMs into an equivalent circuit with synchronous ones. The resulting circuit with synchronous ROMs or synchronous RAMs can be embedded into FPGAs. However, these circuit rewriting algorithms can handle circuits represented by a directed acyclic graph (DAG) and do not work for those with cycles. By the work in this chapter, we succeeded in relaxing the cycle-free condition of circuits. More specifically, we present an algorithm that automatically converts a circuit with cycles using asynchronous ROMs into an equivalent circuit using synchronous ROMs. We also briefly discuss the techniques to improve performance of the AROM-free resulting circuit.
Performance Improvement of the Resulting Circuits

In this chapter, we mainly discuss about the performance improvement of the AROM-free and ARAM-free resulting circuits. Basically, our rewriting algorithms move registers towards the output ports, whenever possible. Hence, in general, the resulting circuits may have the longest paths from input ports to registers/SROMs/SRAMs or from registers/SROMs/SRAMs to registers/SROMs/SRAMs or from registers/SROMs/SRAMs to output ports. Therefore, the resulting AROM-free or ARAM-free circuit has large propagation delay and low clock frequency. Hence, we say that performance of the resulting AROM-free or ARAM-free circuit may be degraded in terms of latency and clock frequency. However, it is easier to improve the performance of the resulting circuits than minimizing the number of clock cycles. For the reader’s benefit, performance improvement techniques in terms of latency and clock frequency of the resulting circuits are described in Chapter 6, although these are beyond of this dissertation.