S. Ichikawa, S. Yamamoto: "Data Dependent Circuit for Subgraph Isomorphism Problem," Proceedings of 12th Int'l Conf. on Field Programmable Logic and Applications (FPL 2002), LNCS 2438, Springer, pp. 1068-1071 (2002).

Data Dependent Circuit for Subgraph Isomorphism Problem

Shuichi Ichikawa and Shoji Yamamoto

Department of Knowledge-based Information Engineering
Toyohashi University of Technology
1-1 Tempaku, Toyohashi, Aichi 441-8580, JAPAN
ichikawa@tutkie.tut.ac.jp
http://meta.tutkie.tut.ac.jp/~ichikawa/index-e.html

Abstract. The subgraph isomorphism problem has various important applications, although it is generally NP-complete and difficult to solve. This paper examines the feasibility of a data dependent circuit for the subgraph isomorphism problem, which is particularly suitable for FPGA implementation. For graphs of 32 vertices, the average logic scale of data dependent circuits is only 5% of the corresponding data independent circuit. The circuit is estimated to be 460 times faster than the software for 32 vertices. Even if the circuit generation time is included, a data dependent circuit is expected to be two times faster than software when there are 32 vertices. For larger graphs, the performance gain would be far larger.

1 Subgraph Isomorphism Problem

The subgraph isomorphism problem is a simple decision problem. Given two graphs G_α and G_β, it is determined whether G_α is isomorphic to any subgraph of G_β. For example, see Figure 1. In this figure, G_β has a subgraph that is isomorphic to G_α, while G_γ does not.

The subgraph isomorphism problem has many applications, including scene analysis in computer vision and search operation in chemical structural formula database. However, the subgraph isomorphism problem is generally NP-complete [1] and computationally difficult to solve.

To solve the subgraph isomorphism problem practically, several algorithms have been proposed. Ullmann [2] proposed a depth first search algorithm with a

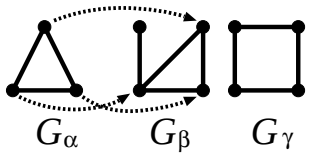

Fig. 1. Subgraph Isomorphism

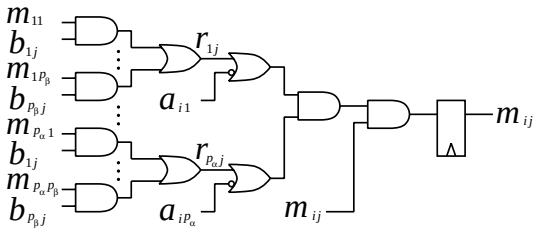

Fig. 2. Element Circuit for Refinement Procedure

smart pruning procedure (*refinement procedure*), which is now the most popular and frequently used algorithm for this problem.

2 Custom Circuit for Subgraph Isomorphism Problem

Ullmann pointed out that his refinement procedure can be implemented with asynchronous hardware [2]. Let p_α and p_β be the number of vertices of graph G_α and G_β, respectively. The adjacency matrices of graph G_α and G_β are represented as $A = [a_{ij}]\,(1 \leq i, j \leq p_\alpha)$ and $B = [b_{ij}]\,(1 \leq i, j \leq p_\beta)$. The temporary matrix $M = [m_{ij}]\,(1 \leq i \leq p_\alpha, 1 \leq j \leq p_\beta)$ is also used in the refinement procedure. Figure 2 illustrates the element circuit to calculate m_{ij}, which was proposed by Ullmann [2]. The whole circuit includes a $p_\alpha \times p_\beta$ array of this element circuit, which requires $O(p_\alpha {p_\beta}^2)$ logic gates. Experimental evaluations revealed, however, the fact that the Ullmann circuit requires too many logic gates for FPGA [3].

In this paper, we examine the data dependent implementation of the Ullmann circuit, which drastically reduces the number of logic gates. Generally, the circuit can be reduced if any input of the logic circuit is set to a constant. This reduction can be recursively applied. Therefore, a data dependent circuit can be smaller than a data *independent* circuit for the same function. Smaller circuits usually work at a higher frequency, and thus can be faster. A data dependent circuit would be more cost-effective, because it is smaller (thus cheaper) and faster than a data independent circuit.

Data dependent circuits are well suited for FPGA implementation due to their nature. On the other hand, the evident problem is that a circuit must be designed (or generated) for each input instance. Hence, the time for logic synthesis, mapping, placement and routing is also important, along with the execution time itself.

For computationally difficult problems such as subgraph isomorphism, the execution time for larger problems grows very quickly. As we show later, the circuit generation time could be inferior and negligible compared to the execution time. This is one of the reasons the authors considered a data dependent approach for computationally difficult problems.

3 Design Approaches

First of all, the original Ullmann circuit should be evaluated as a basis for further evaluation. The VHDL source code was taken from the previous project [3], which is detailed in another paper [4]. This design is denoted "INDEP" in this paper, because it is independent from input graph instances.

Next, we examine a data dependent design. If the input graph G_α is fixed, each a_{ij} in Figure 2 becomes constant. That is, we can replace a_{ij} with the corresponding constants in VHDL source code. At the same time, we can remove the flipflops to store the adjacency matrix A. Once a_{ij} is replaced by a constant, unnecessary logic gates are automatically reduced by logic synthesis software. It is all the same when fixing the adjacency matrix B. We will obtain a maximum reduction when both A and B are fixed.

Though it is easy and natural to leave logic reduction for the logic synthesis system, it takes much memory and execution time. As we mainly need simple and problem-specific optimization, we can manage a substantial part of the reduction in the source code generation phase. This preprocessing drastically reduces the total circuit generation time. Let us denote this design process, in which both input graphs are fixed, by "BOTH2".

4 Evaluation

Each of the results shown in this section is the average value of 50 pairs of G_α and G_β, which are randomly generated trees. Trees were chosen for inputs because they are the sparsest connected graphs. To reduce simulation parameters, $p_\alpha = p_\beta$ is also assumed in simulations. It would be better to investigate dense graphs, disconnected graphs, and the graphs of $p_\alpha < p_\beta$, but these are left for future studies.

Logic synthesis was performed by Synopsys FPGA Compiler II on a Duron 800 MHz PC. Mapping to Lucent OR2C FPGA was done with ORCA Foundry 9.4a on a Pentium II 450 MHz PC. In this study, we did not try the placement and routing. Software implementation of Ullmann's algorithm was evaluated on a Pentium III 600 MHz PC for comparison.

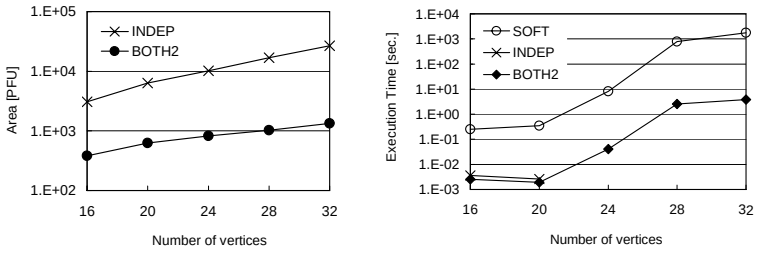

Fig. 3. Logic Scale (Left) and Execution Time (Right)

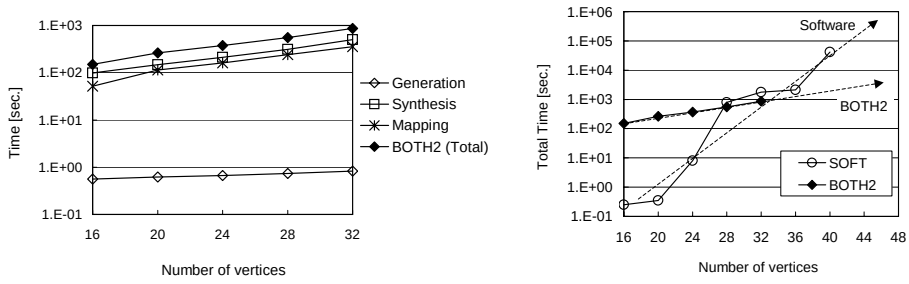

Fig. 4. Circuit Generation Time (Left) and Total Processing Time (Right)

Figure 3 (left) summarizes the logic scale, which is taken from the synthesis report. The logic scale is measured by the number of PFU (programmable function unit) of OR2C FPGA. The logic scale of BOTH2 is only 12%–5% that of INDEP for 16–32 vertices. Greater reduction is expected for larger graphs. Figure 3 (right) shows the expected execution time, which was estimated by the cycle count derived from the simulator and the operational frequency derived from the mapping report. SOFT denotes the software execution time, and BOTH2 denotes the estimated execution time of BOTH2 hardware. The acceleration ratio is 99–460 for 16–32 vertices. The ratio becomes larger in larger graphs.

Figure 4 (left) shows the details of circuit generation time in BOTH2. Circuit generation time grows gradually, and the execution time is negligible compared to generation time. Figure 4 (right) compares the total time (generation + execution) of a data dependent circuit to the software execution time. The data dependent circuit is faster for 28-vertice or larger graphs. The performance advantage would be greater for larger graphs.

Acknowledgment

This work was partially supported by a Grant-in-Aid for Scientific Research from the Japan Society for the Promotion of Science (JSPS) and a grant from the Telecommunications Advancement Foundation (TAF).

References

[1] Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman (1979)
[2] Ullmann, J.R.: An algorithm for subgraph isomorphism. J. ACM **23** (1976) 31–42
[3] Ichikawa, S., Saito, H., Udorn, L., Konishi, K.: Evaluation of accelerator designs for subgraph isomorphism problem. In: Proc. 10th Int'l Conf. Field-Programmable Logic and Applications (FPL2000). LNCS1896, Springer (2000) 729–738
[4] Saito, H.: A study on hardware implementation of Ullmann's algorithm. Master's thesis, Dept. Knowledge-based Information Engineering, Toyohashi University of Technology (2000) (in Japanese).