S. Yamamoto, S. Ichikawa, H. Yamamoto: "Data Dependent Circuit Design: A Case Study," Proceedings of 13th Int'l Conf. on Field Programmable Logic and Applications (FPL 2003), LNCS 2778, Springer, pp. 1024-1027 (2003). ©Springer-Verlag

Data Dependent Circuit Design: A Case Study

Shoji Yamamoto, Shuichi Ichikawa, and Hiroshi Yamamoto

Department of Knowledge-based Information Engineering
Toyohashi University of Technology
1-1 Hibarigaoka, Tempaku, Toyohashi, Aichi 441-8580, JAPAN
{shoji, ichikawa, i8hyama}@ich.tutkie.tut.ac.jp
http://ich.tutkie.tut.ac.jp/en/

Abstract. Data dependent circuits are logic circuits specialized to specific input data. They are smaller and faster than the original circuits, although they are not reusable and require circuit generation for each input instance. This study examines data dependent designs for subgraph isomorphism problems, and shows that a simple algorithm is faster than an elaborate algorithm. An algorithm that requires many hardware resources consumes an accordingly longer circuit generation time, which outweighs the performance advantage in execution.

1 Data Dependent Hardware

If any input of a logic circuit turns out to be constant, the circuit can be reduced. For example, if any inputs of an AND gate turn out to be zero, the output becomes zero (*constant propagation*). This reduction can be applied recursively, consequently reducing the logic scale of the circuit. The derived circuit would operate at a higher frequency than the original, because the logic depth and wiring delay would also be reduced by this reduction.

Since the consequent circuit becomes dependent on the input data instance, such a circuit is called a *data dependent circuit* in the following discussion. The obvious drawback of a data dependent approach is that the derived circuit is *not reusable*. This naturally means that (1) the circuit must be generated for each input instance, and (2) reconfigurable devices such as FPGA must be used.

The total execution time T of a data dependent circuit is given by the sum of the circuit generation time T_{gen} and the execution time T_{exec}. T_{gen} consists of the time for HDL source code generation, logic synthesis, technology mapping, placement, routing, and FPGA configuration. T_{gen} depends on the logic scale, since a larger circuit usually requires accordingly larger generation time. T_{exec} is the product of cycle count and cycle time. Here, the cycle count depends on the algorithm, and the cycle time depends on the implementation. Fast algorithms can make T_{exec} smaller, but they often require more hardware resources and make T_{gen} larger. The total execution time T is thus not so obvious without empirical studies.

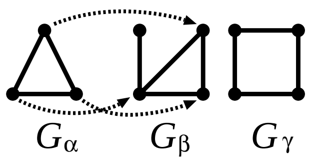

Fig. 1. Subgraph Isomorphism

2 A Case Study: Subgraph Isomorphism Problem

Hereafter, we examine a problem called a *subgraph isomorphism problem* as an example application. A 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 (Fig. 1). In Fig. 1, G_β has a subgraph that is isomorphic to G_α, while G_γ does not.

Ullmann [1] proposed a depth first search algorithm with a smart pruning procedure (*refinement procedure*) for subgraph isomorphism problems. He pointed out that his procedure can be implemented with parallel hardware, but Ichikawa et al. [2] later revealed that his circuit is too large to handle practical problems with the state-of-the-art FPGAs. Ichikawa, Udorn, and Konishi [3] proposed a new algorithm (Konishi's algorithm), which has a simpler pruning procedure than Ullmann's. Konishi's algorithm is generally slower than Ullmann's, but it can be implemented in a much smaller logic circuit than Ullmann's.

Ichikawa et al. [4] [5] previously suggested that data dependent implementations of Ullmann's circuit can be much smaller than the original circuit. The present study confirms this by showing the evaluation results with a Xilinx Virtex-II FPGA.

A data dependent Konishi circuit has not yet been investigated. This study also shows the implementation results of data dependent Konishi circuits, and compares them with data dependent Ullmann circuits. As the original Konishi circuit [3] was not suited for data dependent implementation, we designed a brand-new logic circuit with Konishi's algorithm in this study. In this design, the adjacency check circuits are implemented by parallel hardware. Although this design is an interesting example of a data dependent circuit, we do not have the space to detail it here.

3 Evaluation

This section describes the evaluation results for data dependent circuits. Each result in this section is the average of 100 pairs of G_α and G_β, which are randomly generated. Let p_α and p_β be the number of vertices of G_α and G_β, respectively. We only deal with the cases of $p_\alpha = p_\beta$ in this study. We implemented data dependent circuits for various graph sizes, and measured the execution time on a XC2V1000 FPGA. Our evaluation environment is summarized in Table 1.

Table 1. Evaluation Environment

Item	Note
Circuit Generation	Athlon XP 1800+, Memory 1GB, Windows2000 SP3 Synopsys FPGA Compiler II (2001.08-FC3.7) Xilinx ISE 4.2i (Target device: Virtex-II XC2V1000)
FPGA platform	Insight MicroBlaze Development Kit (XC2V1000, 24MHz)
Software Implementation	Celeron 1.2GHz, Memory 512MB, Red Hat Linux 7.2 Written in C, compiled with gcc-2.95.3

We examined the cases of $(ed_\alpha, ed_\beta) = (0.3, 0.6)$, where ed_α and ed_β indicate the edge density of G_α and G_β, respectively. Edge density ed is defined by the equation $ed = (2\,q)/(p\,(p-1))$, assuming that p is the number of vertices and q is the number of edges. In other words, ed is the ratio of the number of edges to that of the perfect graph K_p. It is clear that $0 \leq ed \leq 1$ holds.

In this study, we examine 4 designs. **Ko** and **Uo** designate the original Konishi circuit and the original Ullmann circuit, respectively. **Kd** and **Ud** designate the data dependent versions of a Konishi circuit and Ullmann circuit for the above-mentioned input graph set.

Figure 2 (left) displays the average logic scale of Ud and Kd, shown by the number of *slices* of Virtex-II FPGA. For the same number of vertices ($8 \leq p_\alpha = p_\beta \leq 16$), the logic scale of Uo is estimated to be 2.4–3.6 times larger than Ud. Meanwhile, Ko is 1.7–1.9 times larger than Kd.

Figure 2 (right) displays the average execution time on XC2V1000 FPGA with a 24 MHz system clock. For comparison, the software implementation of Ullmann's algorithm was also evaluated. The evaluation environment is summarized in Table 1. The execution time of software is denoted by **S** in Fig. 2 (right). For $p_\alpha = p_\beta = 16$, Ud and Kd is 63 and 32 times faster than S, respectively. This performance gain becomes larger in larger graphs.

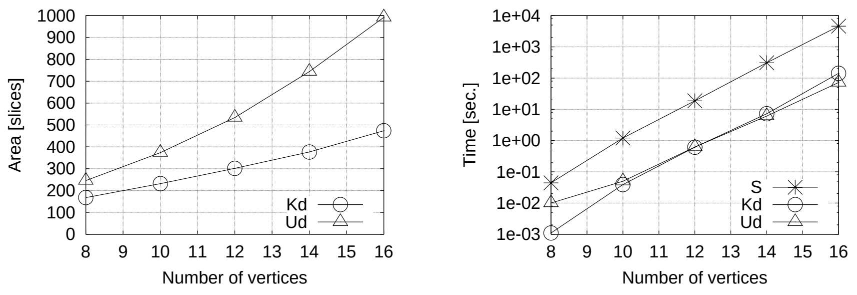

Fig. 2. Logic Scale (left) and Execution Time (right)

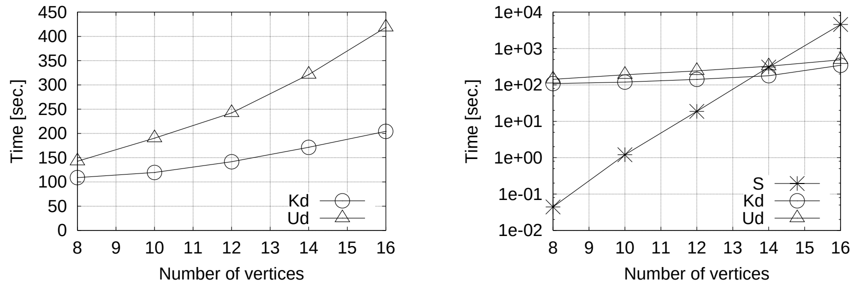

Fig. 3. Circuit Generation Time (left) and Total Execution Time (right)

Figure 3 (left) displays the circuit generation time. It is readily seen that the circuit generation time of Ud is far larger than that of Kd. This comes from the difference of logic scale. Figure 3 (right) shows the average total execution time of Ud and Kd, which is the sum of the circuit generation time and the execution time. The software execution time (S) is also shown for comparison.

Ud and Kd are faster than the software for $p_\alpha = p_\beta > 14$, even reckoning the circuit generation time. When $p_\alpha = p_\beta = 16$, Kd and Ud are 13.3 and 9.4 times faster than the software, respectively. As is readily seen, this performance advantage becomes larger when p_α and p_β are larger. It is also worth noting that Kd is faster than Ud after all, because the long circuit generation time of Ud outweighs its performance advantage over Kd.

Acknowledgment

This work was partially supported by a grant from the Okawa Foundation for Information and Telecommunications. The custom circuits in this study were designed with Synopsys CAD tools through the chip fabrication program of VDEC (the University of Tokyo).

References

1. Ullmann, J.R.: An algorithm for subgraph isomorphism. J. ACM **23(1)** (1976) 31–42
2. Ichikawa, S., Saito, H., Udorn, L., Konishi, K.: Evaluation of accelerator designs for subgraph isomorphism problem. In: Proc. FPL2000. LNCS1896, Springer (2000) 729–738
3. Ichikawa, S., Udorn, L., Konishi, K.: An FPGA-based implementation of subgraph isomorphism algorithm. IPSJ Trans. High Performance Computing Systems **41(SIG5)** (2000) 39–49 (in Japanese).
4. Ichikawa, S., Yamamoto, S.: Data dependent circuit for subgraph isomorphism problem. In: Proc. FPL2002. LNCS2438, Springer (2002) 1068–1071
5. Ichikawa, S., Yamamoto, S.: Data dependent circuit for subgraph isomorphism problem. IEICE Trans. Information and Systems **E86-D(5)** (2003) 796–802