A Study on Hardware Implementation of Ullmann's Algorithm

Graduate Advisor: Shuichi Ichikawa Registration: 963715 Name: Hidemitsu Saito

1 Introduction

Ullmann's algorithm[1] is one of the popular algorithms for detecting subgraph isomorphism. In this paper, we discuss hardware cost and performance of various hardware implementations of Ullmann's algorithm.

2 Ullmann's Method

Let us define two graphs: $G_\alpha = (V_\alpha, E_\alpha)$, $G_\beta = (V_\beta, E_\beta)$. V_α and V_β are vertex sets of G_α and G_β, respectively. E_α and E_β are edge sets of G_α and G_β, as well. Let p_α be $|V_\alpha|$, and p_β be $|V_\beta|$. Adjacency matrices are defined as follows: $A = [a_{ij}]$ $(1 \le i, j \le p_\alpha)$, $B = [b_{ij}]$ $(1 \le i, j \le p_\beta)$. Let us define matrix $M = [m_{ij}]$ $(1 \le i \le p_\alpha,\ 1 \le j \le p_\beta)$ by $m_{ij} = 1$ when the mapping from $v_{\alpha_i} \in V_\alpha$ to $v_{\beta_j} \in V_\beta$ can lead to a subgraph isomorphism, otherwise $m_{ij} = 0$.

Ullmann[1] showed a method (Refinement Procedure) to judge possibility of isomorphism by recursively applying the following formulae to calculate m_{ij}. Here, r_{xj} are temporary variables $(1 \le x \le p_\alpha, 1 \le j \le p_\beta)$.

$$r_{xj} = (\exists y)(m_{xy} \cdot b_{yj}) \tag{1}$$

$$m_{ij} = m_{ij} \cdot (\forall x)(\bar{a}_{ix} \vee r_{xj}) \tag{2}$$

As each element m_{ij} can be updated independently, Refinement Procedure can be implemented by $p_\alpha p_\beta$ units of the circuit shown in Figure 1 (sub_comb). We call this method as **comb** below.

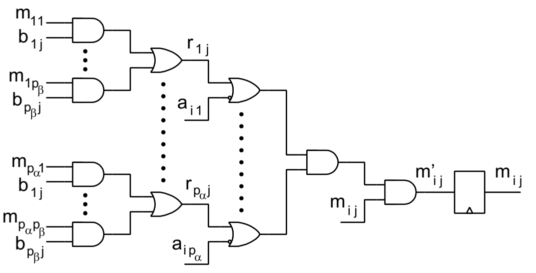

Figure 1: Elemental Circuit for Refinement Procedure[1]

3 Sequential Implementations

The implementation proposed by Ullmann (combinational circuit) achieves short execution time, but it incurs impractical volume of hardware resource. The partial use of sequential circuit can mitigate resource usage in exchange for additional execution time.

Refinement Procedure can terminate if all elements of any row of M are revealed to be '0'. Sequential implementation can utilize this fact when such exceptional condition is found before finishing all iterations to shorten execution time.

(a) Limit the Numbers of sub_comb

Hardware cost can be cut down by reducing the numbers of sub_comb. Execution time gets longer, because the number of m_{ij} that is simultaneously updated is reduced. In this research, the following three methods are examined.

seq_i This method updates M row by row with p_β units of sub_comb. The iteration count is p_α.

seq_i_j This method updates each element of M at a time with a single unit of sub_comb. The iteration count is $p_\alpha p_\beta$.

seq_j This method updates M column by column with p_α units of sub_comb. The iteration count is p_β.

(b) Sequential processing of AND

The n-input AND gate can be implemented by sequential circuit, which processes $(\forall x)$ in formula (2). Hardware resource can be reduced by time-share r_{xj} circuit. Also, iteration can be terminated whenever the output of AND is determined to be '0', by utilizing the fact that the value of n-input AND is zero if any of its inputs is zero.

seq_x This method performs sequential process on x and updates all elements of M at the same time. The iteration count is p_α.

seq_i_x This method performs sequential process on x and updates M row by row. The iteration count is ${p_\alpha}^2$.

seq_j_x This method performs sequential process on x and updates M column by column. The iteration count is $p_\alpha p_\beta$.

4 Estimation Results

For each method, a circuit that can handle up to (p_α, p_β) =(15, 15) is designed. Then, technology mapping to OR2C series FPGA is performed to estimate required hardware resource (the number of PFU) and operating frequency.

The number of clocks for subgraph isomorphism problem is then measured by the hardware simulator written in C language. With this clock count and the operating frequency, the execution time for isomorphism judgement can be estimated. Execution time largely depends on the input graph set. Therefore, the average execution time for 100 random generated connected graphs are used here, where the edge density ed_α, ed_β is both 0.4.

The AT product is calculated here for a measure of evaluation. AT product is the product of the circuit area and the execution time. The circuit is regarded more cost-effective when it gives smaller AT product. Table 1 shows the number of PFU, operating frequency, the sum of average execution time, and the AT product of each method. Figure 2 shows the relation between execution time and circuit area.

Table 1: Estimation Results

Method	PFU	Freq. [MHz]	Time [sec]	AT
comb	2754	22.47	3.89×10^{-2}	1.07×10^{2}
seq_i	1770	27.62	1.04×10^{-1}	1.84×10^{2}
seq_i_j	467	34.22	8.88×10^{-1}	4.15×10^{2}
seq_j	754	28.38	1.37×10^{-1}	1.04×10^{2}
seq_x	671	23.08	1.16×10^{-1}	7.78×10^{1}
seq_i_x	529	33.98	6.31×10^{-1}	3.34×10^{2}
seq_j_x	555	33.98	9.37×10^{-1}	5.20×10^{2}

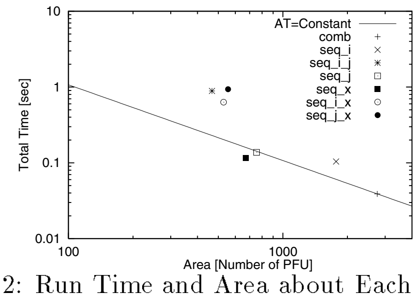

Figure 2: Run Time and Area about Each Method

Slanting line in Figure 2 designates the same AT product as the combinational circuit (comb). The methods seq_j and seq_x are below this line, and regarded better than comb.

5 Conclusion

It is impractical to implement a combinatorial circuit as Ullmann proposed, because it incurs too much hardware cost. On the other hand, sequential implementations of Ullmann's algorithm can achieve practical hardware resource and better AT product than the original Ullmann's proposal.

References

[1] J. R. Ullmann, "An Algorithm for Subgraph Isomorphism," *J. ACM*, Vol. 23, No. 1, pp. 31–42 (1976).