PLACEMENT AND ROUTING ALGORITHMS FOR ASYNCHRONOUS LOGIC CIRCUITS

Pham Quoc Cuong, Nguyen Vu Thien Nga, Dinh Duc Anh Vu, Pham Hoang Anh
University of Technology, VNU-HCM

1. INTRODUCTION

Asynchronous design is more and more developed due to its remarkable benefits compared with synchronous design ([1, 8]):

- No clock skew,
- Lower power,
- Average-case instead of worst-case performance,
- Easing of global timing issues,
- Better technology migration potential,
- Automatic adaptation to physical properties,
- Robust mutual exclusion and external input handling.

With all potential advantages of asynchronous circuits, an efficient design methodology from a high-level description language and powerful CAD tools for asynchronous circuits become an essential demand. A visualization tool supporting automatically placement and routing for logic circuits is one of most important tools. Placement and routing are recognized as NP-Complete problems. Many heuristic algorithms are introduced to deal with these problems but most of them are suitable for physical layout only. Logical layout has its own constraints as below:

- Nets are not allowed to cross the area of cells (1).
- Nets must be not very zigzag (2).
- The number of intersection between nets is as small as possible (3).

These features make the logical layout in routing problem different from the physical one. Consequently, a new routing algorithm is necessary and an appropriate approach is chosen for the previous phase – placement.

For placement problem, the first thing to be considered is partition. It is an important step to achieve the good result in placement phase. Several partitioning algorithms have been proposed such as Simulated Annealing [3], Simulated Evolution [3] and Kernighan-Lin [2]. Placement algorithms are also introduced in literature such as Simulated Annealing [3], Simulated Evolution [3], Breuer [3], Cluster Growth [2] and Advanced Genetic algorithm [4, 5]. Among these algorithms, the approach cooperating Kernighan-Lin and Breuer can give a good result. The implementation of this approach is quite simple and is detailed in section 2.

For routing problem, a lot of algorithms are proposed such as Lee's [2, 6], Hadlock's [3], Soukup's [3] algorithms. However, these algorithms are not entirely suitable to above specific characteristics of logical layout. A new algorithm is introduced in this paper to meet the constraints of logical layout. This algorithm is based on the idea of Lee's algorithm – the wave propagating strategy. The detail of this algorithm is presented in section 3.

2. PLACEMENT

In this section, a new placement algorithm is proposed. This algorithm is a combination of Kernighan-Lin algorithm and Breuer algorithm. The idea is to partition the logic circuit into different parts. These parts of the circuit are then suitably placed on display regions. This process is recursively repeated for each part on its assigned display region.

The proposed algorithm contains a loop which consists of three main steps:

- Partitioning the logic circuit into different parts.
- Dividing the display window in different regions.
- Suitably mapping circuit parts on display regions.

2.1 Partitioning the logic circuit

Partitioning algorithm used in this work is based on Kernighan-Lin algorithm. Kernighan-Lin algorithm divides the logic circuit into hierarchical partitions. A logic circuit consists of components (cells) linked by connections. A set of logic components is hierarchically partitioned into two equal sets so that the number of connections between components in two sets is smallest. This partitioning is terminated when sets contain only one component.

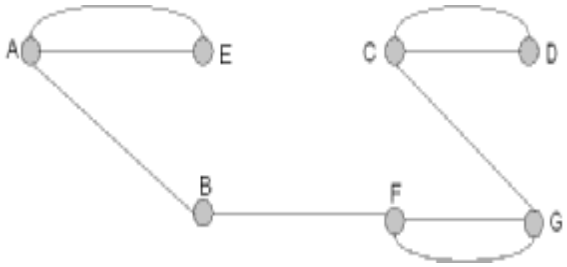

Figure 1 . An example of logical layout

These partitions can be represented by a binary tree. Figure illustrates the binary tree of the example represented Figure .

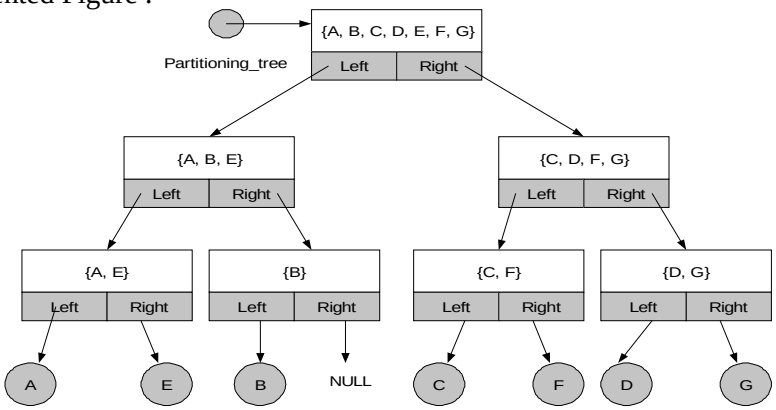

Figure 2. The binary tree created by Kernighan-Lin algorithm

In our work, in order to be suitable to number of display regions a logic circuit is partitioned into four parts. A new step is added to convert the Kernighan-Lin's binary tree into a 4-ary tree. Figure represents the 4-ary tree of the example in Figure .

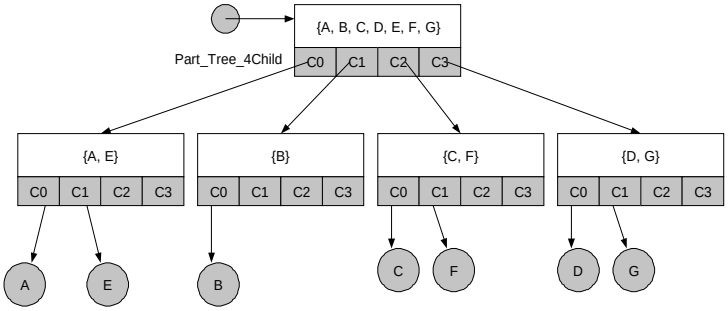

Figure 3. 4-ary tree of the example in Figure 1

2.2. Partitioning the display window

The display window is divided by using the quadrature procedure of Breuer algorithm. The number of dividing step must correspond to the level of 4-ary tree in the previous partition. Figure 4 represents 4 sub-regions of a display region.

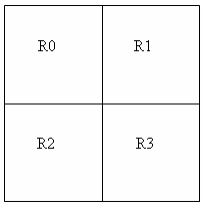

Figure 4. A display region is divided into 4 sub-regions

2.3 Mapping circuit partitions into display regions

Due to the characteristic that signals are mostly one-direction (input or output), the mapping strategy selected in our work is the following

The circuit partition which has the largest number of connections to the left partition of higher level in the hierarchy is placed on region R0.

The circuit partition which has the largest number of connections to the right partition of higher level in the hierarchy is placed on region R1.

The circuit partition which has the largest number of connections to the partition placed on R0 is placed on region R2.

The circuit partition which has the largest number of connections to the partition placed on R1 is placed on region R3.

3. ROUTING

Almost existing routing algorithms in literature, including Lee's algorithm [2], only support two-terminal nets. In this paper, we propose a new routing algorithm which can be applied for a logic circuit with n-terminal nets.

3.1 Proposed routing algorithm

The proposed routing algorithm for a n-terminal net is presented in pseudo-code as follow:

```
Algorithm ROUTING (terminal_list, return_path)
begin
    target_list = f ;
    t = get (terminal_list);
    insert (t, target_list);
    return_path = f ;
    while terminal_list <> f do
        s = get (terminal_list);
        P1 = SUBROUTING (s, target_list);
        L1 = point_of (P1);
        P = P + P1;
        target_list = target_list + L1;
end
```

The algorithm Routing finds a net (path), this net crosses all terminals in terminal_list. In this pseudo-code, SUBROUTING is a routing algorithm from s to any point in the target_list. This algorithm is showed in the next section.

3.2 SUBROUTING algorithm

Our problem concerns with an algorithm that could provide a path from a point to a set of points. There are many routing algorithms in literature that could be applied for searching a path from a source point to a target point ([2, 3]). Among them, Lee's algorithm is the most suitable one that is capable to be modified to meet our goal.

The exploration phase of Lee's algorithm is an improved version of breadth-first search [9, 10].The search can be visualized as a wave propagating from the source:

- The source is labeled '0' and the wave front propagates to all the unblocked vertices adjacent to the source.
- Every unblocked and unlabeled vertex adjacent to the source is marked with a label '1'.
- Every unblocked and unlabeled vertex adjacent to the vertices with a label '1' is marked with a label '2' and so on.
- This process continues until any vertex in list target is reached.

A vertex is blocked if all paths are not allowed to cross it. Otherwise, it is unblocked.

Hence the result of Lee's algorithm is the shortest path from the source point to a destination point. This path may contain a lot of intersection points and therefore may not the best one in our criteria (as described in section 1). Thus, Lee's algorithm is not suitable for logical layout.

Inspired from Lee algorithm, we introduce a new algorithm which is based on minimax-search strategy [9, 10]. First, a path P0 is obtained by using Lee's algorithm. Using the wave propagating strategy similarly to Lee's to consider other paths as below:

- The source is labeled '0' and the wave front propagates to all the satisfied vertices adjacent to the source.
- Every satisfied vertex adjacent to the source is marked with a label '1'.
- Every satisfied vertex adjacent to the vertices with a label '1' is marked with a label '2' and so on.
- This process continues until any vertex in list target is reached. At that time, the path P0 is replaced by the new path.

A vertex is satisfied if it is unblocked, unlabeled and cost of the path from the source to it is better than the cost of P0.

However, Lee's algorithm is the grid-based algorithm so it occupies large memory and long search time for large grid. In order to reduce the memory space and the search time, a solution is proposed. The placement and routing procedures are actually implemented in two different phases. The first phase, placement algorithm is used to place components on display region. Each net is then assigned in the local area of the partition to which it belongs. Thus, in routing phase, almost nets are routed in local grid corresponding with their local area. Because local grids are much smaller than the global grid (corresponding with the global area of the entire netlist), the time for routing phase is significantly reduced.

The complexity of Lee's algorithms is O(m.n) where m.n is grid size. In second phase of routing, other paths are found by using wave propagating and applying minimax-search strategy. The numbers of paths which are found in this step are not larger than m.n. The complexity for routing a net is O(m.n.(m.n +1)). Therefore the total time for routing of large circuit can be acceptable.

3.3 Cost function

To evaluate one path, some added information about this path is needed. This information is represented in following data structure:

```
typedef struct __COST_DATA_STRUCT     NLCostData;
struct __COST_DATA_STRUCT
{
    int Cut;     /* number of intersection points
                 between the path and other nets */
    int Turn;    // number the path turns
    int Len; // length of the path
};
```

of the path

};

Suppose that CD1 and CD2 are respectively cost data structures of two paths. To decide which path is better, the following features need to be considered: (1) the path length needs at least 2 additional units and (2) the turn number needs at least 4 additional units to avoid a blocked point. Figure represents these features.

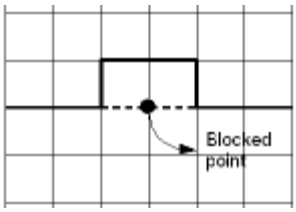

Figure 5. Illustration of avoiding blocked point

So, the cost function is described as follow:

Consider two paths, called P1 and P2, have the cost data CD1 and CD2. The path P1 is the best path before finding P2. It is better than P2 if

```
CD1.Cut < CD2.Cut
Otherwise, P1 is better than P2 if and only if
CD1.Len < CD2.Len – 2(CD1.Cut – CD2.Cut) and
CD1.Turn < CD2.Turn – 4(CD1.Cut – CD2.Cut)
```

3.4 Improvement of the SUBROUTING algorithm

At a given point, the next points are its neighbors [9]. These points are assumed to be visited in clockwise order – top, right, bottom and then left. So, there exist some paths that the algorithm never reaches. In Figure , wave from source s propagates to A, B, C and D. From A, wave propagates to E, F, G. From B, wave propagates to only H because F is already reached from A. Therefore, from S to F, only path SAF is considered while SBF is never considered. This feature may lead to one thing: the result is not the best one. While considering paths searched by Lee's algorithm, many paths are omitted and the best one may be one of these paths.

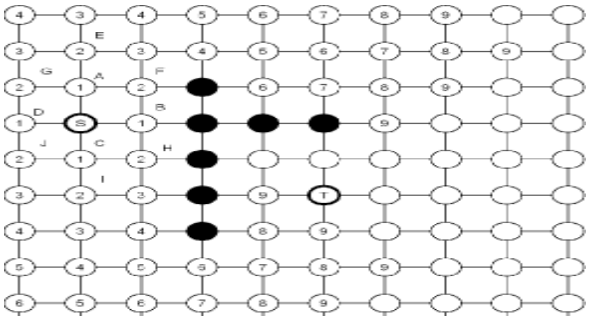

Figure 6.Illustration of the wave propagation

To improve this drawback, the following routing algorithm is used:

- Step 1: Find a path P0 using Lee's algorithm
- Step 2: Create a list of intersection points between P0 and other nets
- Step 3: For each point in this list
- Step 3.1: Block the point
- Step 3.2: Search for better path than P0 by wave propagating strategy
- Step 3.2.1: If the better path exists, replace P0 with the new path and then unblock this point. Return to step 2
- Step 3.2.2: If the better path does not exist, unblock this point and then return to step 3

4. NETLIST VISUALIZATION TOOL

In the design framework for asynchronous circuits developed in the department of Computer Engineering, a netlist visualization tool is implemented (Figure 7). This tool is based on the proposed placement and routing algorithms. It is showed that the correctness of the proposed method is guaranteed. Table 1 shows the total time for placement and routing of some real circuits. Experiments show that these algorithms give a good result in an acceptable execution time.

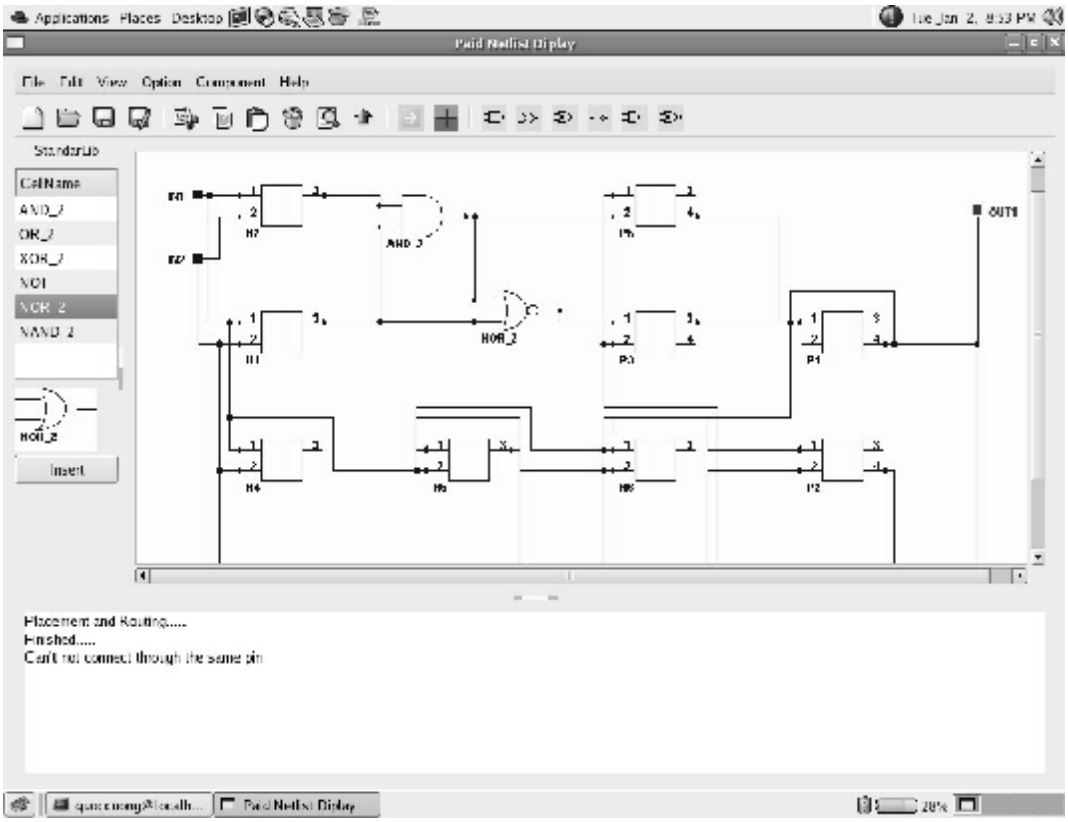

Figure 7.Netlist visualization tool

Table 1. Results of placement and routing algorithms

Test circuit	Num. of components	Num. of nets	Grid size	Total time (ms)
Design 1	4	9	56 x 38	570
Design 2	11	5	92 x 54	3160
Design 3	21	8	164 x 102	66840

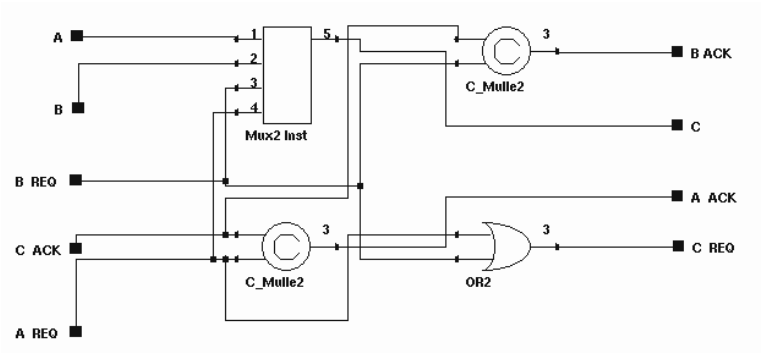

Figure 8. A 4-phase bundled-data of merge component

Figure shows result of placement and routing for design 1, a 4-phase bundled-data of merge component. It consists of an asynchronous control input and a multiplexer controlled by the input requests. The request signals on input channels are mutually exclusive and may simply be ORed

together to produce the request signal on output channel. For each input channel, a C-element procedures an acknowledge signal in response to an acknowledge signal on the output channel provided that the input channel has valid data. A C-element which does not exist in synchronous circuits is a state-holding element much like an asynchronous set-reset latch. When both inputs are 0 the output is set to 0, and when both inputs are 1 the output is set to 1. For other input combinations the output does not change. Design 2 and design 3 are combinational circuits to test correctness and time execution of the proposed algorithm.

5.CONCLUSION

In this paper, automatic placement and routing algorithms for asynchronous circuits are proposed. The combination between Kernighan-Lin and Breuer algorithms gives a good placement for the next phase – routing. A new routing algorithm which is based on wave propagating strategy is also introduced. This algorithm allows obtaining good paths in an acceptable time of executing. Although the proposed solution is for logical layout placement and routing of asynchronous circuits, they can be also expanded to cover synchronous circuits. Experiments are done to show that these algorithms give a good result in an acceptable execution time. These algorithms are implemented and integrated in a design framework for asynchronous circuits (PAiD).

REFERENCES

[1]. Scott Hauck: *Asynchronous Design Methodologies.* In Proc. of the IEEE, Vol. 83, No. 1, pp. 69-93, (1995).

[2]. Sabih H. Gerez, *Algorithms for VLSI Design Automation,* ISBN 0-471-98489-2, June (2000).

[3]. Naveed A. Sherwani: *Algorithms for VLSI Physical Design Automation,* ISBN 0-7923-8392-1 Third Edition, (1999).

[4]. Roony Krashinsky, *GRAPE: Genetic Routing and Placement Engine*, In MIT Laboratory for Computer Science, Cambridge, MA 02139 Embodied Intelligent (6.836) Term Projec, Oct 2000

[5]. Josef Schwarz: *Partitioning-oriented Placement Based on Advanced Genetic Algorithm BOA,* Technical University of Brno Faculty of Engineering and Computer Science Department of Computer Science and Engineering CZ - 61266 Brno, Bozetechova 2. In Procceedings of the Mendel '2000 Conference, (2000).

[6]. Lee: *An algorithm for path connection and its application*. In IRE Trans. Electronic Computer, EC-10, 1961.

[7]. Mohammad S. Munshey and Usman A. Zahidi: *A Rectilinear Shortest-path Routing Algorithm for A Two Dimensional Grid,* In Proceedings 2000 IEEE National Multi Topic Conference, pp 174-178, (2000).

[8]. Jen Sparso and Steve Furber, *Principles of Asynchronous Circuit Design – A System Perspective,* Kluwer Academic Publishers ISBN-10: 0792376137, Jan (2002).

[9]. Stuart Russell & Peter Norvig: *Artificial Intelligence: Modern Approach.* Prentice Hall; 2st edition, (2002).

[10]. Elaine Rich & Kevin Knight: *Artificial Intelligent.* McGraw-Hill; 2nd edition (December 1, 1990) ISBN-10: 0070522634.