An Abstract Model for De-synchronous Circuit Design and its Area Optimization

Gang Jin*†, Lei Wang*, Zhiying Wang*
*School of Computer Science
National University of Defense Technology, Changsha, China
Email: {jingang, leiwang, zywang}@nudt.edu.cn
†School of Computer Science
University of Manchester, Manchester, UK
Email: jgang@cs.man.ac.uk

***Abstract*—De-synchronism is a useful method to design asynchronous circuit automatically from synchronous circuit. An optimising design method based on control graph is introduced. Control graph is an abstract model of de-synchronous circuit. This optimising design method can reduce the area overhead of control path in de-synchronous circuit. To demonstrate the results of this algorithm, it has been applied to a set of benchmark circuits. The number of local controllers in these circuits is markedly decreased by 54%, and 76.3% of the C-elements required to construct handshake circuit are removed. Moreover, this algorithm introduces no performance penalty.**

I. Introduction

Compared with synchronous circuit, asynchronous circuit consumes less power and shows better electromagnetic compatibility and modularity. However, asynchronous circuit is not properly supported by current commercial CAD tools and the design of asynchronous circuit is complicated due to the lack of global control signals.

To avoid the disadvantages of asynchronous design method, de-synchronous methodology [1] generates asynchronous circuit by replacing the global clock network in the standardly synthesized synchronous circuit with sets of local controllers. All transformations could be done within the standard CAD design framework.

De-synchronous design methodology will introduce area overhead in circuit because the global clock is replaced by the local handshake circuit. The major purpose of this paper is to reduce this area overhead. An abstract model has been developed to represent the control path in the circuit and based on this model, an algorithm is introduced to reduce the area consumption of de-synchronous circuits.

II. Design procedure

In this paper, each flip-flop has been transformed into a master-slave latch pair, because latch-based design will be smaller and faster. In [1], a design flow for de-synchronous circuit has been introduced. The whole design procedure proceeds in the following three steps:

1) Split each flip-flop into a master-slave latch pair.
2) Generate the matched delay unit for each combinational logic path.
3) Implement the local controller corresponding to each latch.

III. The abstract model of de-synchronous circuit

Since the data path of de-synchronous circuit presented in this paper have no difference with its synchronous counterpart, the major design concern lies on the implementation of control path. Based on the work in [2], an abstract model of control path in de-synchronous circuit—Control Graph has been introduced. The control graph utilizes a weighted and directed graph to represent local controllers and the handshake circuits between them.

Definition 1: Control Graph

A control graph is a 4-tuple, $< V, F, W, P >$. $< V, F >$ is a directed graph, $P : V \mapsto \{even, odd\}$ is a polarity function, and $W : F \mapsto R$ is the weighted function.

In the directed graph $< V, F >$, set V includes all vertices, and each vertex in V represents a local controller in de-synchronous circuit. Set F includes all edges in the graph, and each edge in F represents a connection between two local controllers which are synchronized with each other. In the circuit, an edge

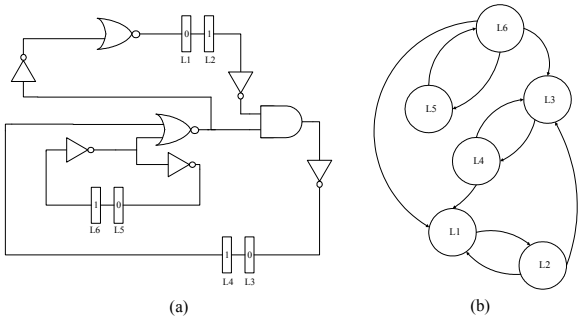

Fig. 1. An example circuit and its control graph

indicates the combinational logic path between two latches. The polarity function P assigns a polarity to each vertex in $< V, F >$ according to the type of the latch. The weighted function W assigns a real number to each edge in $< V, F >$, which indicates the worst case delay associated with the corresponding combinational logic path.

The constructing procedure of control graph is as follow:

1) Get the gate-level net-list of the circuit.
2) Insert a new vertex to the control graph for each latch in the gate level net-list.
3) Determine the connection relationship between the vertices in the control graph, i.e. determining the predecessors and successors of each vertex in the control graph. For each vertex v, all vertices that are connected to the input ports of the combination block whose output port is connected to the input ports of v construct the predecessor set of v, $pre(v)$, and the successor set $post(v)$ can be determined in similar way.
4) Determine the polarity of each vertex in the control graph. For vertex v, if the corresponding latch is a master latch, $P(v) = even$, else $P(v) = odd$.
5) Determine the weight of each edge in the control graph, i.e. the weighted function W. The worst case delay of each Combinational path corresponding to the edge of the control graph can be calculated by normal STA tools. This delay is assigned to this edge as a weight.

An example circuit and its control graph are illustrated in Fig 1.

IV. The optimising design method for control path

Based on this abstract model, an optimising design method is developed to reduce the overall area consumption of control path.

Noticing that several latches could be driven by a single control signal, it is possible to reduce the area overhead of local controllers by driving a certain set of latches by one combined controller. According to this, an optimising design method is proposed based on the combination of controllers.

A. The combination of controllers

Driving multiple latches by a single local controller, the combination of controller will reduce the area overhead of control path.

The combination of controllers can be described as combining the vertices in control graph. Because the latch controllers in practical circuit are exactly mapped into the corresponding control graph, it is possible to use combining the vertices in control graph to represent the practical latch controllers. Two vertices can be combined only when the following rules are complied with.

- $P(u) = P(v)$. The vertex u and v have the same polarity. After the combination, a new vertex w will be inserted into the control graph.
- $pre(w) = pre(u) \bigcup pre(v)$.
- $post(w) = post(u) \bigcup post(v)$.
- The weight values of the edges remain unchanged.

B. The performance evaluation function

Definition 2: Average Cycle Time[3]

Average cycle time of an asynchronous circuit is the longest average cycle time among all circles in the timed petri-net model corresponding to the circuit.

Average cycle time is a performance evaluation parameter of the de-synchronous circuit.

Definition 3: Timed Petri-net

Timed petri-net is defined as a 5-tuple $N =< P, T, F, \Delta, M_0 >$, where $P = \{p_1, p_2, ..., p_m\}$ is the non-empty and finite set of place, $T = \{t_1, t_2, ..., t_n\}$ is the the non-empty and finite set of transition, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relationship, $\Delta : T \mapsto R$ is the execution time function of transitions, $M_0 \subseteq P$ is the initial marking of the petri-net.

The timed Petri-net of a de-synchronous circuit can be derived from the corresponding control graph.

Pointed out by [4], for particular timed petri-net, the lower bound of average cycle time is:

$$\tau = \max_i \{\frac{y_i^T (C^-)^T D x}{y_i^T M_0}\}$$

where $C^- = [c_{ij}^-]_{m \times n}$ and c_{ij}^- is the weight of the directed arc from the transition j to the place i. D is a diagonal matrix constructed by θ_{ii}, which is the

execution time of the transitions t_i in the timed petri-net. M_0 is an array that has the same number of elements with the places in the timed petri-net and contains the initial number of tokens kept in the corresponding places.

If the petri-net is a marked graph (a subclass of Petri-nets that can model decision-free concurrent systems), the maximum average cycle time can be obtained from this linear programming problem.

$$\begin{aligned} T = \max & Y^T (C^-)^T \theta \\ & C \cdot Y = 0 \\ st. \quad & Y^T \cdot M_0 = 1 \\ & Y \geq 0 \end{aligned}$$

C. The optimising algorithm

According to the polarities of vertices in the control graph, the latches in the circuit can be divided into two subsets. Only vertices with the same polarity can be combined. In the extreme case, a circuit can only have two local controllers, one for the master latches and the other for the slave latches, which behaviors just like a synchronous counterpart and eliminates all the benefits gained from asynchronous design style. The purpose of combination is to find an optimal tradeoff between the overall area and the benefit of asynchronism.

In order to reach the optimal result, it is required to traverse all possible partitions of this two subsets of vertices, which is an unacceptable NP-hard problem. Therefore, a polynomial time algorithm that is able to seek an approximate optimal result is introduced.

To constrain the loss of asynchronism, a threshold is defined to control the maximal number of the latches can driven by one local controller. Both the reduction of area consumption and the loss of asynchronism are proportional to this threshold. The algorithm to combine the controllers is demonstrated as follows:

Algorithm 1 (Combination of control latches): Given a control graph $C =< V, F, W, P >$, $V_{deleted}$ is a set that restores the vertices deleted during the optimising procedure. Set $n = |V|$. An array $(\theta_1, \theta_2, ..., \theta_n)$ is maintained , where θ_i presents the times that v_i has been combined. Θ is the threshold assigned to this algorithm. The algorithm can be described as:

1) Set $V_{deleted} = \emptyset$;
2) Set $i = 1$, $\tau_{min} = \infty$;
3) Set $j = i + 1$;
4) If $v_i \in V_{deleted}$, then jump to 5. If $v_j \in V_{deleted}$ or $\theta_i + \theta_j > \Theta$, then jump to 4. Combine v_i and v_j of C to produce a new control graph C', if $T(C') \leqq T(C)$, then set $\tau_{min} = T(C')$, $i_{min} = i$, $j_{min} = j$. If $j \geqq n$, then jump to 5, otherwise $j = j + 1$ and jump to 4.
5) If $i > n$, then jump to 6, otherwise $i = i + 1$ and jump to 3;
6) If $\tau_{min} < \infty$, then combine $v_{i_{min}}$ and $v_{j_{min}}$ to produce the new control graph C and set $V_{deleted} = V_{deleted} \cup \{v_{j_{min}}\}$.

The core step of this algorithm is step 4, in which $T(C)$ has been calculated. $T(C)$ is a linear programming problem. The time complexity to solve this linear programming problem is $O(n^{3.5})$ [5], where n is the number of variables in this problem, i.e. the number of the places in the timed petri-net corresponding to the circuit. The iterative depth of our optimising algorithm is 2, so the time complexity of the algorithm is $O(n^{5.5})$. The time complexity of this algorithm is polynomial time.

V. Experiment Results

In order to evaluate the effectiveness of the optimising algorithm, several experiments are conducted on some benchmark circuits and the results are discussed in the following part.

The optimising algorithm has been tested on nine benchmark circuits selected from the ISCA'89 benchmark circuit sets. Two different thresholds are set to determine the optimization efficiency. Since the control path is implemented by local controllers and the handshake circuits between them, the total area of control path is mainly composed by the area of local controllers and the area of handshake circuit which could be approximated to the area of the C-elements used to construct control path. In these experiments, the threshold Θ is set to 2 and 3.

TABLE I
THE EXPERIMENT RESULT OF COMBINATION OF CONTROLLERS

	Original			Optimized($\Theta = 2$)			Optimized($\Theta = 3$)		
Circuit	V	E	C	V	E	C	V	E	C
s27	6	10	4	4	6	2	4	6	2
s298	28	83	55	16	38	22	12	25	13
s344	30	93	63	16	40	24	12	35	23
s349	30	93	63	16	40	24	12	35	23
s386	12	42	30	6	12	6	4	6	2
s420	32	152	120	14	44	28	12	30	18
s510	12	42	30	6	12	6	4	6	2
s526	42	165	123	22	77	55	14	48	34
s1448	12	42	30	6	12	6	6	12	6

According the experiment results, it can be seen that the number of vertices and edges are both dramatically decreased, and the number of C-elements required by control path are also dramatically reduced. These results

prove that this optimising algorithm is very effective to reduce the area of control path in de-synchronous circuit. And it can also be seen that when $\Theta = 2$, the number of local controllers in the circuit is totally reduced 37.9%, the number of C-elements is totally reduced 66.6%, and when $\Theta = 3$, the number of local controllers in the circuit is totally reduced 54%, the number of C-elements is totally reduced 76.3%. Because our algorithm is directed by the performance evaluation function of de-synchronous circuit, the optimising algorithm markedly reduces the overall area of circuit but has no penalty on performance of the circuit.

Nevertheless, since the fan-in and fan-out of a single local controller may be increased because of combination, a small area overhead would be introduced by this algorithm.

TABLE II
THE AVERAGE FAN-IN AND FAN-OUT OF THE BENCHMARK CIRCUITS BEFORE AND AFTER OPTIMIZATION

	Original	Optimized	
Circuit	fan-in/out	fan-in/out($\Theta = 2$)	fan-in/out($\Theta = 3$)
s27	2.67	3.00	3.00
s298	3.96	4.15	4.42
s344	4.10	4.38	5.42
s349	4.10	4.38	5.42
s386	4.50	4.00	4.50
s420	5.75	4.75	5.17
s510	4.50	4.00	4.50
s526	4.93	5.41	6.43
s1448	4.50	4.00	4.00

It is observed that the change of the average fan-in and fan-out is relatively small compared with the notable reduction in the number of local controllers and C-elements. In some cases, because of the great reduction of the number of edges in the circuit, the average fan-in and fan-out may even be decreased.

VI. CONCLUSIONS

In this paper, an optimising design method to balance the penalties and benefits of de-synchronous circuit is introduced. The combination of local controllers allows the control signals of multiple latches generated by one local controller. In this way, the overhead of local controllers is sharply reduced. Shown in the experiment results, nearly half of the local controllers are reduced and nearly $2/3$ of the C-elements required to construct the control path are removed.

The de-synchronous design methodology can improve EMI and markedly shorten the design cycle of asynchronous circuit. The optimising design method proposed in this paper can minimize the overhead induced by de-synchronous design. It is believed that the optimising de-synchronous circuit by this algorithm is a practical way to design asynchronous circuit before the pure asynchronous design methodology are widely used.

REFERENCES

[1] J.Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou, "A concurrent model for desynchronization," in *IWLS 2003*, 2003.

[2] I. Blunno, J. Cortadella, A. Kondratyev, L. Lavagno, K. Lwin, and C. Sotiriou, "Handshake protocols for de-synchronization," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Shanghai, China.

[3] L. Wang, Z. ying Wang, and K. Dai, "Cycle period analysis and optimization of asynchronous timed circuits," in *11th Asia-Pacific Conference on Advences in Computer Systems Architecture, LNCS 4186*. Shanghai: Springer-Verlag, sep 2006, pp. 502–508.

[4] L. Wang, "Design and anslysis techniques of asynchronous embedded microprocessors," Ph.D. thesis, National University of Defence technology, Changsha, September 2006.

[5] N. Karmarkar, "A new polynomial-time algorithm for linear programming." *In Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pp. 302–11, Apr 1984.