An Optimal Design Method for De-synchronous Circuit Based on Control Graph*

Gang Jin, Lei Wang, Zhiying Wang, and Kui Dai

School of Computer Science,National University of Defense Technology, Changsha, 410073, China
jingang@nudt.edu.cn

Abstract. De-synchronous is a very useful method to design asynchronous circuit automatically from synchronous description of circuits. This paper introduces an optimal design method based on Control Graph which is an abstract model of the de-synchronous circuit. The main purpose of this optimal design method is to reduce the extra overhead in the area of the de-synchronous circuit. The optimization algorithm takes the performance evaluation function based on the Control Graph of the de-synchronous circuit as its heuristic function. The performance evaluation function presented in this paper is a linear programming problem. In the end of this paper, the optimal method is applied to a set of benchmark circuits. The number of the local controllers in these circuits is markedly reduced by 54%, and the number of C-elements that is required to construct the handshake circuitry between local controllers is also reduced by 76.3%. So the entire area of the circuit is sharply reduced. Because this design method is directed by the performance evaluation function of the circuit, there is no penalty in performance of the de-synchronous circuit.

Keywords: de-synchronous, asynchronous, performance evaluation, algorithm, control graph, Petri-net.

1 Introduction

Along with the scale of chip is getting larger and larger, the clock skew becomes more and more serious. In order to solve the clock skew problem, a large balance clock tree needs to be constructed, involving much area and consuming much energy of the circuit. Compared with synchronous circuits, the power dissipation of asynchronous ones are really small. As the different parts of the circuit operate at different speed and switching activity, the Electro-Magnetic Compatibility has increased. With their interfaces free from global constraints such as operating frequency, asynchronous circuit provides inherent modularity, which is the major advantage of the asynchronous design methodology. However, there are some disadvantages related to asynchronous circuit. On the one hand, there are no

* Supported by the National Natural Science Foundation of China under Grant No. 90407022.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 70–79, 2007.

good CAD tools for asynchronous circuit design; on the other hand, the design of asynchronous circuit is really complex since there is no global control signal in the circuit.

In order to avoid the disadvantages of the asynchronous design style, the concept of de-synchronous has been brought up [1], whose essential idea is that the design starts from a standard synchronous synthesized circuit, and then the global clock network is directly replaced by a set of local controllers. All steps of this design flow can be implemented within standard CAD tools. This method works efficiently in dealing with the difficulties which will be encountered when adopting a pure asynchronous design style.

When adopting de-synchronous design methodology, a local handshake circuitry should be inserted into the circuit in order to take place the global clock. So an extra overhead, i.e. area, will be introduced to the circuit. The major purpose of this paper is to reduce this area overhead. A abstract model is developed to represent the control path of the circuit, based on which an algorithm is introduced to reduce the area of the de-synchronous circuit.

In Section 2, the related works of this paper are introduced. The concept and design flow of the de-synchronous design style are explained in Section 3. In Section 4 we use an abstract model—Control Graph based on Petri-net to model the de-synchronous circuit. In Section 5, based on Control Graph of de-synchronous circuit, an optimization algorithm is developed to combine the local controllers in order to reduce the entire area of the circuit, which is directed by the performance evaluation function. In Section 6, the algorithm is applied to a number of benchmark circuits. Section 7 is the conclusion part.

2 Related Work

The idea of generating local control signals for a synchronous latch-based circuit is proposed by Sutherland in his Turing award lecture [2]. The micorpipeline theory has been adopted in several designs [3] and CAD tools [4,5,6].

Theseus Logic has developed a design method [7] that uses the commercial EDA tools to synthesize and optimize the datapath, and directly translates the control path into an asynchronous implementation. But this approach suffers from high overhead and requires the non-standard HDL style.

Liner and Harden have introduced a method that replaces each logic gate with an equivalent sequential handshake asynchronous circuit, where the synchronization information is encoded into the code of data using an LEDR delay-insensitive code [8]. This approach also have an expensive overhead.

The similar work as this paper is presented by A. Davare in [9]. He also directly replaces the global clock network with local controllers. But in that paper, he only introduces a simple method to optimize the circuit without concerning the circuit performance.

Our group has also done a lot of works on de-synchronous circuit design. We have presented a de-synchronous circuit design flow [10,11] based on macrocell. This design flow tries to compatible with current EDA tools for synchronous

design, which makes it easy to design asynchronous circuits. Based on this design methodoloy, we have designed a 32-bits asynchronous multiplier in $0.35\mu m$ process. Compared with the synchronous partner, our design of multiplier has smaller area, lower power dissipation and higher performance.

3 Design Step for De-synchronous Circuit

Generally speaking, the design based on flip-flop will need more complex control circuitry, which will lead to an extra area overhead. In this paper, we translate each flip-flop to a pair of master-slave latches, because latch-based design will be smaller and faster. In [1], a design flow of de-synchronous circuit has been introduced. All steps of this flow starting from a flip-flop-based synchronous circuit that can be implemented with standard CAD tools. The de-synchronization method proceeds in the following three steps:

1. Splitting each flip-flop into a master-slave latch pair.
2. Generating matched delay for combinational logic.
   Serving as a completion detector for the corresponding combinational block, each matched delay must be greater than or equal to the delay of the critical path of the corresponding combinational block.
3. Implementing the local controller corresponding to each latch.
   For each latch of the latch-based synchronous circuit, a local controller will be inserted into the circuit for generating the control signal. The local controllers communicate with each other over handshake. Request signals from predecessors are delayed by the matched delay generated in the previous step.

4 The Model of De-synchronous Circuit

In fact, the data path of the de-synchronous circuit presented in this paper has no difference with its synchronous partner, so the major design concern should be paid to the design of the control path of the circuit. Based on the work in [12] , this paper introduces an abstract model of the control path of the de-synchronous circuit—Control Graph. The Control Graph takes a weighted directed graph to represent the local controllers corresponding to the latches in the circuit and the handshake circuitry between the local controllers. The definition of the Control Graph is:

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,* $W : F \mapsto R$ *is a weighted function.*

In the directed graph $< V, F >$, V is a set of all vertices in this graph, and each vertex in V represents a local controller in the de-synchronous circuit; F is a set of all edges in the graph, and each edge in F represents a connecting relationship of two local controllers, in other words, these two local controllers

should synchronize with each other. In a real circuit, an edge also indicates that there is a combination logic path between the two latches. The polarity function P assigns a polarity to each vertex of $< V, F >$ according to the type of corresponding latch, such as master or slave. The weighted function assigns a real number to each edge of $< V, F >$, which indicates that the worst case delay associates with corresponding combination logic path of this edge.

The construction process of the Control Graph is as follows:

1. Using logical synthesis tools to synthesize the circuit, getting the gate-level net-list of the circuit.
2. Inserting a new vertex to the Control Graph for every latch in the gate level net-list.
3. Determining the connecting relationship between the vertices in the Control Graph, i.e. determining the predecessors and successors of every vertex in the 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, $pre(v)$. The successor set $post(v)$ can be determined in same way.
4. Determining the polarity of each vertex in the Control Graph. For vertex v, if the latch corresponding to v is a master latch, define $P(v) = even$, else define $P(v) = odd$.
5. Determining the weight of 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 STA tools. This delay is assigned to the edge as a weight.

An example circuit and its Control Graph are illustrated in Fig 1.

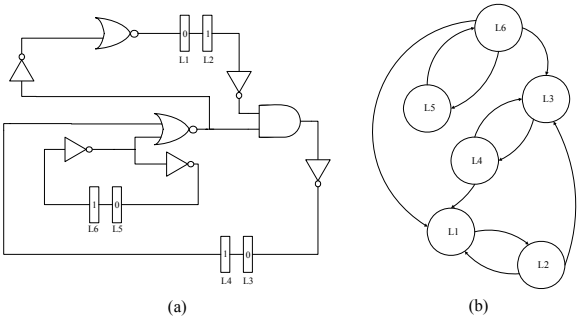

Fig. 1. An example circuit and its Control Graph

The circuitry of control path can be automatically derived from the Control Graph of the de-synchronous circuit. For each local controller v, there is a pair of req_{in}, ack_{in} signals, and a pair of req_{out}, ack_{out} signals, where req_{in}, ack_{in} are the handshake signals with the local controllers in set $pre(v)$; req_{out}, ack_{out} are the handshake signals with the local controllers in set $post(v)$. All req_{out} of the

local controllers in $pre(v)$ are combined by a multi-input C-element to generate the req_{in} of the local controller v; All ack_{out} of the local controller in $pre(v)$ are directly connected to the ack_{in} of the local controller v.

5 Optimization of the Control Path

Based on the Control Graph of the de-synchronous circuit, this paper develops an optimization algorithm to reduce the area of the de-synchronous circuit.

The control path of the de-synchronous circuit is made up by the local controllers and the handshake circuitry between them. The de-synchronous circuit which avoids the area overhead of the global clock introduces the new area overhead of local controllers and the handshake circuitry between them. As the data path of the de-synchronous circuit is the same as its synchronous partner, reducing the area of the control path of the de-synchronous circuit will be important to reduce the entire area of the whole circuit.

We can naturally combine the control signals of a set of latches with a single control signal. One local controller generates the latch signal of a set of latches, by which the number of the local controllers in the control path will be decreased and in turn reducing the whole area of the circuit. According to this concept, we develop an optimal method based on the combining control signals of latches.

5.1 Combining Control Signals of Latches

The meaning of combining control signals of latches is that the control signals of several latches are generated by a single local controller, by which the area of the circuit will be reduced.

In fact, combining control signals of latches is to combine the vertices in the control graph. Because the mapping between the vertex in the control graph and the latch controller in the circuit is one to one, we can use the combination of the vertices in the control graph to represent combining control signals of latches. In this way, we can combine the vertices in the control graph to achieve the purpose of combining the local controllers in the circuit. When two vertices in the control graph are combined, the following rules must be followed:

- Only when two vertices have the same polarity, they can be combined, i.e. $P(u) = P(v)$; 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 of each edge remains unchanged.

5.2 The Performance Evaluation Function Based on Control Graph

Definition 2. *Average Cycle Time[13]*

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

The average cycle time is a performance evaluation parameter of the asynchronous circuit. De-synchronous circuit is a kind of asynchronous circuit, so this parameter can also be the 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 transition,* $M_0 \subseteq P$ *is the initial marking of the Petri-net.*

The timed Petri-net of a de-synchronous circuit can be derived from the Control Graph of the circuit. The procedure is:

1. Every vertex in the Control Graph has been extended to a substructure in timed Petri-net. This substructure is constructed by two transitions and one place; one transition is called input transition which can have several inputs and only one output connecting to the input of the place; the other is called output transition which can have several output and only one input connecting to the output of the place.
2. Each edge in the Control Graph become a place in the timed Petri-net, whose input is connected to the output transition of the extended substructure derived from the source vertex of this edge; the output is connected to the input transition of the extended substructure derived from the target vertex of this edge.
3. The place in the substructure produced in step 1 can present the vertex in Control Graph. Each place corresponding to the odd vertex in the Control Graph should be included in initial marking M_0.
4. For each transition t_i, the corresponding transition execution time θ_i is the maximum delay of the edges which input to t_i.

It is easy to be confirmed that the timed Petri-net derived from above procedure is live and safe.

[14] has pointed that the bottom bound of average cycle is:

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

where $C^- = [c_{ij}^-]_{m \times n}$, c_{ij}^- is the weight of directed arc from transition j to place i; D is the diagonal matrix constructed by θ_{ii}, which is the execution time of the transitions t_i in timed Petri-net; M_0 is an array having the same number of elements as the place set of timed Petri-net, which contains the initial number of tokens kept in the corresponding place.

If the Petri-net is a marked graph (a subclass of Petri-nets[15] that can model decision-free concurrent systems), the maximum average cycle time can be gained

by solving the below Linear Programming problem.

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

This method is very fast, so it works in dealing with large scale problems. For a de-synchronous circuit whose Control Graph is C, we take the average cycle time $T(C)$ calculated in this method as the performance evaluation function of this de-synchronous circuit.

5.3 The Optimization Algorithm

According to the polarity of the vertex 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, it will result in a circuit with only two local controllers, one for the master latches and the other for the slave latches, which behaviors just like the synchronous partner and eliminates the benefit of the asynchronous one. In fact, the purpose of combination of latches is to find a best partition of these two subsets to achieve the best balance between the area and the benefit of asynchronism.

In order to find the exact optimization result, it is necessary to traverse every partition of the two subsets of the vertices, which is unacceptable for it is a NP-hard problem. Therefore, a polynomial time algorithm to find an approximate optimal solution is introduced in this paper.

To prevent the benefit of asynchronism lost, a threshold is introduced to the optimization procedure. The threshold means the up-bound of the number of latches controlled by a single local controller. The larger value the threshold assigned, the more local controller can be combined, the more benefit of asynchronous is lost, vice versa. The algorithm for combining control signals of latches is as follows:

Algorithm 1 (combining control signals of latches). *Given a Control Graph* $C =< V, F, W, P >$. $V_{deleted}$ *is a set that keeps the vertices deleted during the optimization. Set* $n = |V|$. *It maintains an array* $(\theta_1, \theta_2, ..., \theta_n)$, *where* θ_i *present the times which* v_i *has been combined.* Θ *is the threshold assigned to algorithm. The algorithm is as follows:*

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. If we chose the Karmarker's algorithm to solve the linear programming problem, the time complexity is $O(n^{3.5})$ [16], where n is the number of variables in this problem, i.e. the place of timed petri-net corresponding to the circuit. The iterative depth of our optimization algorithm is 2, so the time complexity of the algorithm is $O(n^{5.5})$. The time complexity of this algorithm is polynomial time.

6 Experiment Result

In order to evaluate the effectiveness of the algorithm presented in this paper, several experiments are conducted on some benchmark circuits and describing in the following part.

We choose a subset of the ISCAS'89 benchmark circuit sets, 9 circuits of which are chosen. To determine the effect of the threshold, we choose different threshold values to run the algorithm on these set of benchmark circuits. The reduction of the number of the local controllers is illustrated in Table 1. Since the control path is made up by these local controllers and the handshake circuitry between them, and the major part of the handshake circuitry is C-elements, the total area of the control path is mainly composed by the area of the local controllers and the area of the C-elements. For this reason, we also illustrate the number of C-elements needed by the circuit. In the experiment, we investigate the result when threshold is $2(\Theta = 2)$ and $3(\Theta = 3)$.

Table 1. The experiment result of combining control signals of latches

	Original			Optimized($\Theta = 2$)			Optimized($\Theta = 3$)		
Circuit	Vertex	Edge	C-element	Vertex	Edge	C-element	Vertex	Edge	C-element
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

From the experiment result presented above, we can see that the number of vertices and edges are both decreased, and the C-elements required by the circuit are also dramatically reduced. We can also see that when $\Theta = 2$, the number of local controllers in the circuit is totally reduced by 37.9%, the number of

C-elements is totally reduced by 66.6%, and when $\Theta = 3$, the number of local controllers in the circuit is totally reduced by 54%, the number of C-elements is totally reduced by 76.3%.

Along with the reduction of the number of the local controllers in the circuit, the fan-in and fan-out of a single local controller may be increased, which may cause some extra overhead of area introduced into the circuit. In Table 2, we illustrate the average fan-in and fan-out of these circuits before and after optimization.

Table 2. The average fan-in and fan-out of the benchmark circuits before and after optimization

	Original	Optimized	
Circuit	Avg. fan-in/out	Avg. fan-in/out($\Theta = 2$)	Avg. 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

From the result above, we observe that the change of the average fan-in and fan-out is relatively small compared with the notable reduction in the number of the local controllers and the C-elements. In some cases, because of the great reduction of the number of handshakes in the circuit, the average fan-in and fan-out may even be decreased.

7 Conclusions

In this paper, an optimization method to balance the penalties and benefits of de-synchronous circuit is introduced. It is allowed that the control signals of several latches to be combined into a single signal generated by one local controller. In this way, the overhead of the local controllers can be sharply reduced. From the experiment results, we can see that nearly half of the local controllers can be eliminated and nearly 2/3 of the C-elements required to construct the control path can also be eliminated.

The de-synchronous design methodology can improve EMI, and markedly shorten the design cycle of asynchronous circuit. Our optimization method for de-synchronism can conquer the problem that de-synchronism may bring some overhead into circuit. We believe that de-synchronism with our optimization algorithm is a very useful method to design asynchronous circuit before the pure asynchronous design methodology be widely used.

References

1. Cortadella, J., Kondratyev, A., Lavagno, L., Sotiriou, C.: A concurrent model for desynchronization. In: IWLS 2003 (2003)
2. Sutherland, I.E.: Micropipelines. Communications of the ACM 32 (1989)
3. Furber, S.B., Garside, J.D., Gilbert, D.A.: Amulet3: A high-performance self-timed arm microprocessor. In: Proc. International Conf. Computer Design(ICCD) (October 1998)
4. Bardsley, A., Edwards, D.: Coompiling the language Balsa to delay-insensitive hardware (1997)
5. van Berkel, K.: Handshake Circuits: an Asynchronous Architecture for VLSI Programming. Cambridge University Press, Cambridge (2001)
6. Blunno, I., Lavagno, L.: Automated synthesis of micro-pipelines from behavioral verilog hdl. In: Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 84–92. IEEE Computer Society Press, Los Alamitos (2000)
7. Ligthart, M., Fant, K., Smith, R., Taubin, A., Kondratyev, A.: Asynchronous dsign using commercial hdl synthesis tools. In: Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 114–125. IEEE Computer Society Press, Los Alamitos (2000)
8. Linder, D.H., Harden, J.C.: Phased logic: Supporting the synchronous design paradigm with delay-insensitive circuitry. IEEE Transactions on Computers 45, 1031–1044 (1996)
9. Davare, A., Lwin, K., Kondratyev, A., Sangiovanni-Vincentelli, A.: The best of both worlds: The efficient asynchronous implementation of synchronous specifications. In: Design Automation Conference (DAC), ACM/IEEE (June 2004)
10. Yong, L., Lei, W., Rui, G., Kui, D., Zhi-ying, W.: Research and implementation of a 32-bits asynchronous multiplier. Journal of Computer Research and Development 43 (November 2006)
11. Gong, R., Wang, L., Li, Y., Dai, K.: A de-synchronous circuit design flow using hybrid cell library. In: ICSICT 2006. Proc. of 8th International Conference on Solid-State and Integrated-Circuit Technology, Madrid, Spain, pp. 149–158. IEEE Computer Society Press, Los Alamitos (2004)
12. Blunno, I., Cortadella, J., Kondratyev, A., Lavagno, L., Lwin, K., Sotiriou, C.: Handshake protocols for de-synchronization. In: Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, Shanghai, China
13. Wang, L., Zhi-ying, W., Dai, K.: Cycle period analysis and optimization of asynchronous timed circuits. In: Jesshope, C., Egan, C. (eds.) ACSAC 2006. LNCS, vol. 4186, pp. 502–508. Springer, Heidelberg (2006)
14. Wang, L.: Design and Anslysis Techniques of Asynchronous Embedded Microprocessors. Ph.d. thesis, National University of Defence technology, Changsha (September 2006)
15. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 541–580 (April 1989)
16. Karmarkar, N.: A new polynomial-time algorithm for linear programming. In: Proceedings of the 16th Annual ACM Symposium on Theory of Computing, pp. 302–311 (April 1984)