Efficient Realization of Strongly Indicating Function Blocks

P. Balasubramanian, D.A. Edwards
School of Computer Science, The University of Manchester,
Oxford Road, Manchester M13 9PL, United Kingdom.
E-mail: (padmanab, doug)@cs.man.ac.uk

Abstract

This paper presents a technique for efficient gate-level realization of strongly indicating function blocks. For the function block implementing the desired logic, the input state space explodes as it expands exponentially for even a gradual increase in the number of inputs. In this context, a novel design methodology for realizing non-regenerative logic as a function block, under the discipline of quasi-delay-insensitivity with four-phase handshaking and dual-rail encoding, which adheres to the strongly indicating timing regime has been discussed. Approximately 3 times reduction in transistor cost has been achieved by the proposed method in comparison with a recent work, based on analysis with benchmarks and widely used digital circuit functionality; in particular cases the savings are remarkable.

1. Introduction

Digital logic design has been dominated by synchronous solutions for the past several decades. The renewal of interest in and requirement of asynchronous design is largely motivated by the need to overcome the problems associated with clocking. Future deep sub-micron technologies would be characterized by irregular parameter variations (voltage and temperature variations, process changes and noise), which could make asynchronous design an attractive solution. It is expected that parametric variance of device delay might reach 35% by 2020 [1]. Since asynchronous circuits are self-timed, they tend to absorb the deviations of device characteristics.

Unlike their synchronous counterparts, asynchronous circuits dispense with the need for clock synchronization and so they must operate correctly independent of any delays in circuit elements. The vast majority of existing design automation flows target synchronous circuits. Even when asynchronous designs leverage existing tool flows, they introduce large area overheads. Hence asynchronous function block implementation requires special techniques compared to the synthesis of non-regenerative (combinational) logic for synchronous digital circuits.

2. Function block – Description

A function block is the asynchronous equivalent of a digital combinatorial circuit [2]: it computes one or more outputs from a set of input signals. Apart from computing the desired function outputs based on the function inputs, a function block must also be transparent to handshaking that is implemented by its surrounding latches. Function blocks can be strongly indicating (SI) or weakly indicating (WI). In this work, we confine ourselves to the SI timing regime. A strong indication function block waits for all of its inputs to become valid/empty, before it starts to compute and produce valid/empty outputs. Besides it should possess a valid combinatorial structure implying that it should not have dangling inputs/outputs and feed-back paths.

Circuits designed following the four-phase protocol dual-rail (DR) approach are generally quasi-delay-insensitive (QDI), since the class of delay-insensitive (DI) circuits is rather small [3]. QDI is as robust as the DI class to variable operating conditions and transistor variations [4]. A circuit is QDI if and only if the production rule set describing it is stable and non-interfering [5]. It is also an attractive design style mainly for the simple timing closure and analysis that it permits. QDI circuit design assumes that both operators and wires can take an arbitrary time (finite and positive) to switch, except for certain wires that form isochronic forks [6] (weakest compromise to delay insensitivity). The isochronic fork assumption has been defined by Martin in [6] as: “In an isochronic fork, when a transition on one output is acknowledged and thus completed, the transitions on all outputs are acknowledged and thus completed”.

3. Related work

Various techniques proposed earlier are found to be suitable for strongly indicating function block design employing four-phase handshaking with DR encoding [7] [8] [9] [10] [11] [12]. [7] and [8] require the generation of all minterms (standard product terms), which is O[2^n] for '*n*' inputs, resulting in an input space explosion. For [8], decomposition of the multiple input C-element is necessitated. This may result in unacknowledged transitions (orphans) occurring within the circuit making it Q^nDI, leading to violation of speed-independence conditions.

In [9], a self-timed function block design was proposed, but decomposition procedures for the monotonic implementation of the dual-rail combinatorial network were not considered, without which gate orphans cannot be avoided. This restricts the scalability of this approach.

Five different techniques for implementation of a function block were discussed in [10]. However, many of them correspond to the lowest level of design abstraction (transistor level), though one of them is suitable for implementation with standard cells. But without specialized logic optimization techniques, possibility exists for unacknowledged transitions, which could pose potential hazards and probably lead to circuit malfunction.

A recent work [11] [12] dealing with the synthesis of QDI circuits (using 2-input C-elements and 2-input OR gates) described using DR or *m*-of-*n* codes, encompasses a decomposition technique incorporating elements of both conventional rectangle covering based multilevel logic synthesis and speed-independent synthesis. Though it is a versatile method, it also suffers from the problem of input space explosion as the entire input space is being covered with no room for accommodating don't care function set(s). However, it has solved the difficulty faced by synthesis tools such as Petrify [13] in implementing non-regenerative logic functions by asynchronous style due to the general purpose nature of the algorithms employed therein and also owing to the large size and concurrency inherent in combinational logic designs.

4. SI Function block design

The main issue addressed in this paper is to efficiently realize any arbitrary combinational logic as a QDI function block adhering to the SI timing regime within the discipline of 4-phase handshaking with DR encoding using standard library cells (including 2-input and 3-input C-elements), avoiding the possibility of orphan(s) creation while simultaneously satisfying the monotonic cover constraint (MCC) [14]. This poses a significant challenge for asynchronous logic design, as in general, the area overhead is considerable in comparison with conventional logic synthesis solutions.

4.1. Dependency set, D(C) of a Boolean cube C and Cubes Dependency Intersection set, CDI

The dependency set of a Boolean cube entails enumeration of all the literals in their actual form that the cube depends upon for its evaluation to a logic '1'. For a cube C specified by *ef'g'h*, its D(C) = {*e*,*f'*,*g'*,*h*}.

The intersection of a cube dependency set with another is characterized by the literals that are common to both the cubes and is referred to by an intersection set, CDI. For example, with D(C_1) and D(C_2) described by {*abc'd'*} and {*ab'c'g*} respectively, the intersection of their corresponding dependency sets is, CDI[D(C_1), D(C_2)] = {*a*,*c'*} and its cardinality is 2.

4.2. Synthesis issues and Decomposition constraints

- Obtain a two-level minimized solution for all the multiple outputs (in positive phase). Similarly obtain a two-level reduced solution for all the multiple outputs (in negative phase). While the former corresponds to the expressions for true outputs, the latter corresponds to the equations for false outputs, after DR encoding.
- Consider the whole functionality as a global combinatorial network with DR inputs and outputs.
- Transform the cover cubes of both the true and false function output(s) expressions into disjoint cover cubes. This is accomplished through redundancy insertion using identity law and also by recursive application of absorption and distributive axioms of Boolean algebra.
- If any |CDI[D(C_1), D(C_2)]| = 0 and D(C_1)/D(C_2) is not singleton; redundancy insertion through identity axiom is followed by application of distributive and absorption laws. Additional cube(s) get introduced.
- For *k* cubes in the network, perform cube dependency intersection with each of the remaining (*k*-1) cubes. Isolate the common literals by an intermediate node and substitute into its parent nodes.
- However, this is permissible only when HD(C_1, C_2) = 1 to preserve speed-independence, where HD signifies the Hamming distance between the cubes.
- A cube in the transformed Boolean network is to be chosen only once, also an intermediate node (cube) resulting from the network should be chosen once, whether for substitution or for covering.

• The covering cube(s) absorbs the covered cube. If a cube covers more than one cube in the network, then priority for a cube to be covered would be based upon a maximal value of cover extent for the covered cube.

• A covered cube can be the covering cube for another cube and a covered cube can be substituted into as many covering cubes as possible apart from its actual covering cube provided cover extent for other covering cubes with respect to this cube is maximal.

• Uncovered smaller cubes with small |D(C)| (bound defined) might exist in the final decomposed network.

• For cases otherwise, the entire input space has to be explored to facilitate an optimal QDI realization.

5. Results, Discussion and an Illustration

Analysis has been carried out by considering some IWLS benchmarks and few widely used digital circuits.

Table 1. Comparison in terms of 2-input gate count and transistor cost

Circuit and its specification	Realization style	2-input gate count	Device count
c17 (5 I/p, 2 O/p)	[9]	229	1262
	[12]	224	672
	Proposed	123	320
newcwp (4 I/p, 5 O/p)	[9]	210	1078
	[12]	166	660
	Proposed	175	524
newtag (8 I/p, 1 O/p)	[9]	3750	22590
	[12]	466	1396
	Proposed	146	356
wim (4 I/p, 7 O/p)	[9]	234	1166
	[12]	194	828
	Proposed	178	542
newbyte (5 I/p, 8 O/p)	[9]	149	614
	[12]	177	670
	Proposed	237	706
con1 (7 I/p, 2 O/p)	[9]	1697	10042
	[12]	486	1460
	Proposed	191	534
newtpla1 (10 I/p, 2 O/p)	[9]	240	1258
	[12]	342	1100
	Proposed	175	500
3:8 Decoder with high Enable (4 I/p, 8 O/p)	[9]	128	502
	[12]	139	554
	Proposed	160	490
4:16 Decoder with high Enable (5 I/p, 16 O/p)	[9]	277	1158
	[12]	396	1844
	Proposed	333	1058
5:32 Decoder with high Enable (6 I/p, 32 O/p)	[9]	602	2646
	[12]	1285	6702
	Proposed	690	2290
4-bit Priority Encoder (4 I/p, 4 O/p)	[9]	160	806
	[12]	152	576
	Proposed	79	228
8-bit Priority Encoder (8 I/p, 8 O/p)	[9]	4164	24662
	[12]	3440	15712
	Proposed	184	562

Comparison has been carried out on the basis of two-input gate equivalent count [9] and transistor cost [11] [12] for logic realization and they are listed in Table 1. On the basis of [11], the cost of a 3-input C-element is equated to 12. However, an n-input OR gate is decomposed into (n-1) OR_2 gates to maintain a common ground in theoretical estimation. On the whole, for the circuit functionalities listed in Table 1, the proposed realization enables reduction in 2-input gate count and transistor cost in comparison with [9] by 3.4× and 7.4× respectively; 1.8× and 3× with respect to [12]. In case of *newbyte*, the device reduction for other methods is attributed to a singleton ON-set for all the function outputs and no sharing, which is rare. For the 8-bit priority encoder, the reduction in transistor cost is huge for the proposed method (43× over [9] and 27× over [12] respectively). Nevertheless, this excludes shared OR-logic extraction and substitution.

The QDI implementation of a strong indication 2-bit by 2-bit unsigned multiplier with inputs (b1, b0), (a1, a0) and outputs (p3, p2, p1, p0) is portrayed by figure 1, as an illustration of the proposed method. Synchronization of the DR encoded inputs is achieved through Block I1 (shown in figure 1). Decomposed multi-level realization of the actual logic which strictly satisfies the MCC is represented by Block I2. Block I3 ensures that the strong indication criterion is satisfied.

Table 2. Comparison of different methods for a SI 2-bit unsigned multiplier implementation

Realization style	2-input gate count	Device count	Critical path elements
[9]	124	596	AND_4, OR_4, OR_3, CE_2
[12]	152	576	2 CE_2, 4 OR_2
Proposed	162	382	CE_3, 2 CE_2, OR_4, OR_2

6. Conclusion and Further work

This paper highlights a novel strategy for SI function block design within the ambit of QDI with DR encoding and 4-phase handshaking. The decomposition technique is complex and decomposes several gates in the entire implementation space simultaneously. Enhancing the efficacy of the proposed procedure, power/performance evaluation and focusing on WI function block realization constitute further work.

7. Acknowledgment

The authors acknowledge the support of EPSRC, UK for the SEDATE project grant EP/D052238/1.

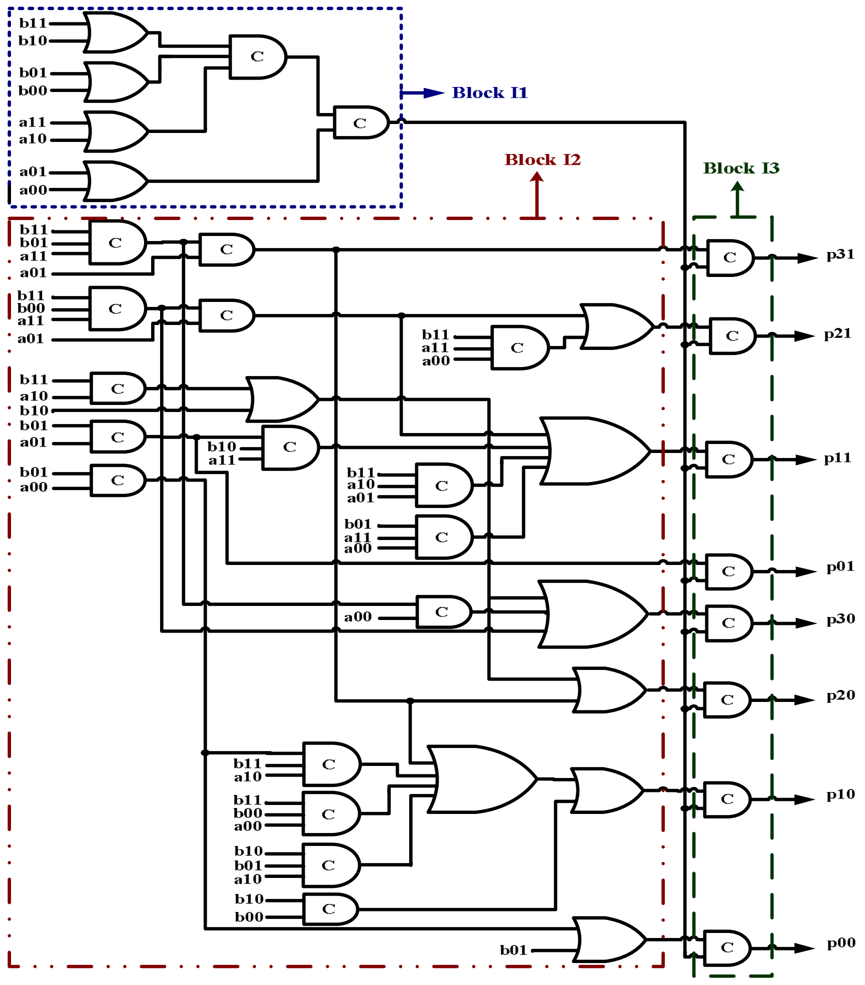

Figure 1. Proposed QDI implementation of the SI 2-bit by 2-bit unsigned multiplier

8. References

[1] SIA's ITRS Report 2005, http://www.itrs.net

[2] J. Sparso and S. Furber (Eds.), *Principles of Asynchronous Circuit Design*, Kluwer Academic, 2001.

[3] A.J. Martin, "The limitations to delay-insensitivity in asynchronous circuits", *Proc. 6th MIT Conference*, pp. 263-278, MIT Press, 1980.

[4] T.M. Mak, "Is CMOS more reliable with scaling", *Proc. IEEE International On-Line Testing Workshop*, July 2002.

[5] R. Manohar and A.J. Martin, "Quasi-delay-insensitive circuits are Turing-complete," *Caltech CS Technical Report*, CS-TR-95-11, 1995.

[6] A.J. Martin, "Compiling communicating processes into delay-insensitive VLSI circuits", *Distributed Computing*, vol. 1, no. 4, pp. 226-234, 1986.

[7] C.L. Seitz, in *Introduction to VLSI Systems*, *Chapter 7 – System Timing*, Addison-Wesley, 1990.

[8] J. Sparso and J. Staunstrup, "Delay-insensitive multi-ring structures", *Integration, the VLSI journal,* vol. 15, no. 3, pp. 313-340, October 1993.

[9] I. David, et al., "An efficient implementation of Boolean functions as self-timed circuits", *IEEE Trans. on Computers*, vol. 41, no. 1, pp. 2-11, January 1992.

[10] C.D. Nielsen, "Evaluation of function blocks for asynchronous design", *Proc. ACM European DAC*, pp. 454-459, 1994.

[11] W.B. Toms and D.A. Edwards, "Efficient synthesis of speed independent combinational logic circuits", *Proc. Asia South-Pacific DAC*, pp. 1022-1026, 2005.

[12] W.B. Toms, "Synthesis of Quasi-Delay-Insensitive Datapath Circuits", *PhD thesis*, Univ. of Manchester, 2006.

[13] J. Cortadella, et al., "Petrify: A tool for manipulating concurrent specifications and synthesis of asynchronous controllers", *IEICE Trans. on Information and Systems*, vol. E-80D, no. 3, pp. 315-325, 1997.

[14] A. Kondratyev, et al., "Hazard-free implementation of speed-independent circuits", *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 17, no. 9, pp. 749-771, September 1998.