Performance-driven Dual-rail Insertion for Chip-level Pre-fabricated Design

Fu-Wei Chen
Dept. of Computer Science
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.
d9762812@oz.nthu.edu.tw

Yi-Yu Liu
Dept. of Computer Science and Engineering
Yuan Ze University
Chungli, Taiwan, R.O.C.
yyliu@saturn.yzu.edu.tw

Abstract

In recent years, pre-fabricated design styles grow up rapidly to amortize the mask cost. However, the interconnection delay of the pre-fabricated design styles slows down the circuit performance due to the high capacitive load. In this paper, we propose a technique to insert dual-rail wires for pre-fabricated design styles. Furthermore, we propose an effective dual-rail insertion algorithm to reduce the routing area overheads caused by the inserted dual-rail wires. Taking the wire criticality*, the* delay significance*, and the* wire congestion *into consideration, our proposed algorithm is capable of trading additional routing area overheads for the interconnection performance improvement. The experimental results demonstrate that our proposed algorithm reduces the interconnection delay by 11.4% with 5.8% routing area overheads.*

I. Introduction

With the advances of VLSI design technology, mask cost increases dramatically due to the lithographic difficulties. The one-time-use mask cost is no longer affordable for small and medium volume ASIC designs. This results in a higher threshold for conventional standard cell ASIC designs. To amortize the mask cost, some pre-fabricated design styles are then proposed. Structured ASIC is a semi-chip-level pre-fabricated design style. Most of the masks are fixed except the contact-mask and some via-masks. The functionalities and the interconnections are then specified by properly assigning the contacts and the vias. It is also known as the via patterned gate array (VPGA) [1] [2]. Field programmable gate array (FPGA) is a full-chip-level pre-fabricated design style. Both the devices and the wires have been pre-fabricated in FPGA. CAD tools synthesize a circuit into a series of control bits to configure both the functionalities and the interconnections [3] [4].

In this paper, we are going to discuss the chip-level (full-chip-level and semi-chip-level) pre-fabrications. Both design styles have been intensively studied with the advancement of the VLSI technology. FPGA provides the field programmability with very low non-recurring engineering (NRE) cost and medium performance; while structured ASIC provides higher performance at the expense of some NRE cost. The metal wires of both design styles have been pre-fabricated. One major design difficulty of the pre-fabricated wires is to accurately predict required routing resource. Since different designs require different routing resource, we need to over-design the pre-fabricated wires to meet a wide variety of routing resource requirements. However, once a design is mapped, the over-designed routing resource is then wasted. The pre-fabricated routing architectures suffer high capacitive load consequently. With the boost impact of interconnection delay on circuit performance in modern VLSI design, the capacitive issues in pre-fabricated routing architectures should be carefully taken into account.

Uniform dual-rail routing architecture is proposed as a wire tapering alternative to improve the interconnection performance. By exploiting negligible coupling capacitance between two adjacent wires with the same signal source, dual-rail routing architecture provides a signal propagation channel with similar propagation delay, less crosstalk noise, and less power consumption to the wire tapering architecture [5]. Hence, we apply the dual-rail architecture in pre-fabricated design styles to improve the interconnection performance. Furthermore, we propose an effective dual-rail insertion algorithm to reduce the routing area overheads caused by the inserted dual-rail wires. To the best of our knowledge, there is no previous work which exploits the anti-Miller effect in pre-fabricated design styles to trade moderate routing area overheads for interconnection performance.

The rest of this paper is organized as follows. The preliminary background knowledge is given in Section II. Section III presents our proposed dual-rail wire insertion technique for pre-fabricated routing architecture and our dual-rail insertion algorithm. The experimental results are drawn in Section IV. Section V concludes this paper.

II. Preliminary

A. Chip-level Pre-fabricated Design Style

The island-style pre-fabrication is one of the well-known chip-level pre-fabricated design styles [3] [4]. There are two major components in an island-style pre-fabrication, the *logic component* and the *routing component*. The *logic component* is composed of the configurable logic blocks (CLB). Each CLB contains a set of lookup tables and flip-flops to realize both the combinational and sequential logic functions. The *routing*

This work is supported in part by the National Science Council of Taiwan under Grant NSC-96-2221-E-155-070 and NSC-97-2221-E-155-071-MY2.

978-3-9810801-5-5/DATE09 © 2009 EDAA

component consists of the routing channels, the connection-blocks, and the switch-blocks. There are two types of routing channels, the horizontal channels and the vertical channels. The horizontal and vertical channels are made up of several horizontal and vertical tracks, respectively. The connection-blocks are used to determine the connectivities between the I/O of CLB and the horizontal/vertical tracks. The switch-blocks are used to determine the connectivities between different tracks in different routing channels. Hence, with tracks, connection-blocks, and switch-blocks, *routing components* are capable of interconnecting *logic components*.

By properly configuring both *logic components* and *routing components*, various VLSI circuits can be implemented in the island-style pre-fabrication. However, the major problem for designing a general purpose pre-fabricated design style is to properly allocate both *logic components* and *routing components* for various target designs. Many previous researches focus on related issues. Tom and Lemieux propose a mapping technique for module blocks under a given channel-width constraint [6]. Their depopulation technique satisfies the given channel-width constraints at the expense of larger FPGA size and longer wire length. Chang and Wong compares a set of universal switch-block structures to improve the routability [7]. Fan et al. further improve the switch-block design to maintain the routability with lower area cost [8]. Betz and Rose investigate the channel widths of both horizontal channels and vertical channels for better routing area utilization [9]. They further propose heterogeneous wire segmentation and a mixture of pass-transistors and tri-state buffers for signal propagation delay reduction [10]. Those proposed methods improve either the flexibility or the performance of the pre-fabricated design style. However, these techniques inevitably require to over-design either the *logic components* or the *routing components* for a general purpose pre-fabricated design style. Once a design has been mapped into a pre-fabricated design style, the over-designed resource is wasted.

B. Dual-rail Routing Architecture

We have proposed a uniform dual-rail routing architecture, which merges two adjacent wires into one signal propagation channel [5]. Due to the anti-Miller effect, the coupling capacitance between these two merged wires can be greatly reduced. Hence, the wire resistance of this new routing channel is reduced by two without increasing the coupling capacitance. Figure 1(a) draws the schematic of a dual-rail architecture. According to the schematic, we construct a distributed π-model *SPICE* netlist for circuit simulation in Figure 1(b). In Figure 1(b), R, C_c, C_g, L, and M, represent the resistance, coupling capacitance, ground capacitance, self inductance, and mutual inductance of the wire, respectively. The middle two wires represent the *victim*, whereas the upper/lower wires represent the *aggressor*. According to the *SPICE* simulation results, the interconnection delay is reduced by 10% to 15% as compared with the original routing architecture.

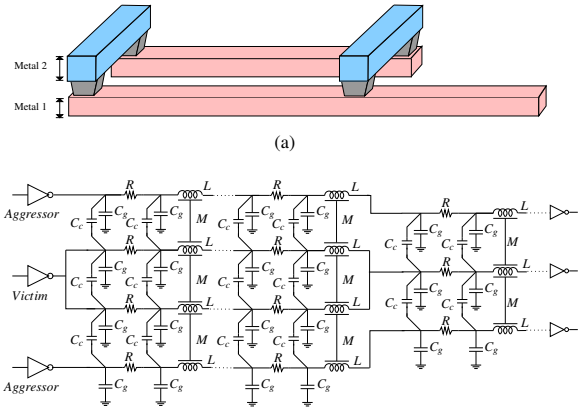

Fig. 1. Schematic and *SPICE* netlist of dual-rail routing architecture

III. Dual-rail for Pre-fabricated Routing Architecture

A. Dual-rail Wire Insertion

According to the aforementioned characteristics of pre-fabricated design style, the routing resource utilization is low. This is a good opportunity for dual-rail wire insertion. To enable the dual-rail routing, we need to insert a dual-rail wire next to the original wire. After that, we need to modify the corresponding connection-blocks and switch-blocks to complete the dual-rail routing. For simplicity, we take the crossbar switch-block as an example to illustrate dual-rail insertion for pre-fabricated design styles. Notice that the dual-rail wire insertion can also be implemented in different switch-block structures, such as the universal switch-block [1].

Figure 2 shows the schematic of modified connection-block. We may either turn on/off the programmable pass-transistor for FPGA or place/remove a mask-programmable via for structured ASIC in the connection-block to enable/disable the dual-rail routing. Figure 3 shows the schematic of modified switch-block. We may either turn on/off the programmable pass-transistor for FPGA or place/remove a mask-programmable via for structured ASIC in the switch-block to enable/disable the dual-rail routing. By enabling dual-rail insertion capabilities for both connection-blocks and switch-blocks, we are capable of improving the interconnection performance of pre-fabricated design styles. Nevertheless, the inserted dual-rail wire may double routing track requirement and hence incurs huge routing area overheads. Therefore, we need to effectively select a small set of wires for dual-rail insertion.

B. Dual-rail Insertion Algorithm

We propose a dual-rail insertion algorithm, which inserts necessary dual-rail wires for those critical and near critical paths. Before illustrating our dual-rail insertion algorithm, we define the following notations.

The *circuit delay* is the maximum delay of a combinational path among four path groups, paths from primary input to

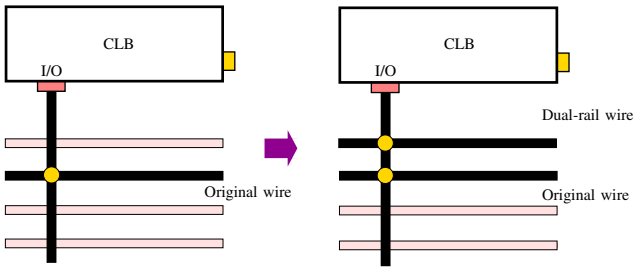

Fig. 2. Modified connection-block for dual-rail routing

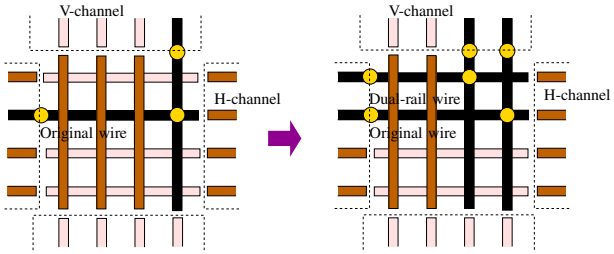

Fig. 3. Modified switch-block for dual-rail routing

primary output, paths from primary input to flip-flop, paths from flip-flop to flip-flop, and paths from flip-flop to primary output. After performing the static timing analysis on the circuit, we can compute the arrival time as well as the required time. The *wire slack* is the required time minus the arrival time. The *maximum slack* is the maximum *wire slack* among all wires.

Definition 3.1: Wire criticality: The *wire criticality* of wire i, denoted as $Crit(i)$, can be formulated as

$$Crit(i) \equiv 1 - \frac{wire\ slack\ i}{maximum\ slack}$$

The *wire criticality* is a fractional number between 0 and 1. High *wire criticality* represents that the wire is on a critical or near-critical path.

Definition 3.2: Delay significance: The *delay significance* of wire i, denoted as $Sig(i)$, can be formulated as

$$Sig(i) \equiv min(\frac{\text{delay of wire } i}{W \times\ circuit\ delay}, 1),$$

where W is a user-specified constant to weight each wire according to the wire delay. In this work, we set $W = 15\%$. The *delay significance* is a fractional number larger than 0 and less than or equal to 1. High *delay significance* represents that the wire contributes a large amount of delay.

Definition 3.3: Channel congestion: The *channel congestion* of wire i in routing channel x, denoted as c_i^x, can be formulated as

$$c_i^x \equiv \frac{\text{number of routed wires}}{\text{width of the channel}}$$

The *channel congestion* of wire i in routing channel x is a fractional number larger than 0. High *channel congestion* represents that the channel is congested.

Definition 3.4: Wire congestion: The *wire congestion* of a wire i, denoted as $Cong(i)$, can be formulated as

$$Cong(i) \equiv MAX(c_i^x), \forall \text{ routing channel } x \text{ of wire } i$$

Algorithm *Dual-rail_Insertion(2-pin wires)*
Begin
perform static timing analysis
foreach wire i
if $slack(i) \leq 15\% \times circuit\_delay$
wire i is a dual-rail candidate
end if
end foreach

foreach candidate wire i
if $Crit(i) \times Sig(i) \geq Cong(i) - k$
insert dual-rail
update channel congestion
end if
end foreach
update wire delay
End

Fig. 4. The *Dual-rail Insertion* algorithm

The *wire congestion* of wire i is the maximum *channel congestion* of wire i for all routed channels. High *wire congestion* represents that the wire pass through some congested channels.

Based on the aforementioned notations, we can illustrate the flow of our dual-rail insertion algorithm. First, we perform original wire routing. Once the routing is completed, we use Elmore delay model to compute the delay of each 2-pin wire. Static timing analysis is then performed to compute the arrival time, the required time, and the *wire slack*. Since we can have 10% to 15% of wire delay improvement after dual-rail insertion, we select wires with slack less than and equal to 15% of *circuit delay* as candidate wires for dual-rail insertion. After that, we take three aforementioned factors, the *wire criticality*, the *delay significance*, and the *wire congestion*, to reduce the wire overheads caused by the inserted dual-rail wires. The following formula is used to decide whether a dual-rail wire could be inserted or not.

$$Crit(i) \times Sig(i) \geq Cong(i) - k, \quad (1)$$

where k is a user-specified constant. The dual-rail wire is inserted if the condition of Equation (1) satisfies. The reason behind this formula is quite intuitive. The product of *wire criticality* and *delay significance* implies the potential gains from dual-rail insertion. The *wire congestion* minus k implies the tolerance of routing area overheads. Hence, the overall formulation mimics that, a dual-rail wire could be inserted only if both the original wire with large interconnection delay and the wire is on a critical or near-critical path under a certain of congestion constraint. Finally, static timing analysis is performed again to compute the exact timing of the design. The algorithm of dual-rail insertion is drawn in Figure 4.

IV. Experimental Results

We use a FPGA physical design automation environment, *VPR* [3], as our dual-rail insertion platform. Both the crossbar switch-block and the dual-rail insertion algorithm proposed in

TABLE I

RESULTS OF DUAL-RAIL WIRE INSERTION

Circuit	*Profile*		*Original*			*VPR+D*			*VPDR*		
	CLB	*Net*	$Track_h$	$Track_v$	*Delay*	Ov_h (%)	Ov_v (%)	*Timing* (%)	Ov_h (%)	Ov_v (%)	*Timing* (%)
apex4	1262	1271	33	33	1.19E-08	15.15	15.15	15.37	6.06	3.03	12.99
bigkey	1707	1936	30	30	1.77E-08	100.00	16.67	11.37	13.33	6.67	8.96
C6288	2416	2448	30	30	1.58E-08	40.00	23.33	3.41	3.33	3.33	2.72
des	1591	1847	35	35	1.97E-08	60.00	17.14	20.27	2.86	5.71	20.27
diffeq	1497	1561	37	37	8.24E-09	16.22	8.11	11.60	8.11	2.70	11.60
dsip	1370	1599	37	37	1.24E-08	16.22	8.11	11.60	2.70	2.70	6.29
elliptic	3604	3735	34	34	1.74E-08	61.76	20.59	7.13	5.88	5.88	7.30
ex1010	4598	4608	46	46	2.03E-08	8.70	8.70	7.84	2.17	2.17	7.95
ex5p	1064	1072	28	28	2.34E-08	28.57	21.43	20.86	3.57	3.57	18.52
frisc	3556	3576	40	40	1.86E-08	32.50	20.00	12.60	7.50	5.00	12.56
misex3	1397	1411	28	28	2.22E-08	42.86	28.57	16.87	7.14	7.14	15.52
s298	1931	1935	28	28	1.79E-08	100.00	35.71	8.50	7.14	3.57	5.11
seq	1750	1791	35	35	9.26E-09	20.00	11.43	11.23	2.86	2.86	6.35
spla	3299	3315	31	31	1.79E-08	12.90	6.45	13.55	6.45	3.23	13.44
tseng	1047	1099	26	26	7.11E-09	26.92	15.38	24.89	7.69	7.69	21.09
average						38.79	17.12	13.14	5.79	4.35	11.38

Section IV are integrated into *VPR* and denoted as *VPDR*. The experiments are performed on a 2.4 GHz Linux workstation with 2 GB memory. We use large circuits (more than 20 tracks in a channel) from the *MCNC* benchmark suite. In our experiment, all benchmark circuits are optimized by *SIS* using *rugged* script [11]. Then, we use *FlowMap* [12] to perform technology mapping on the optimized circuits. All the circuits are mapped into 4-input LUTs and flip-flops. After that, we use *T-Vpack* to pack 4-LUTs and flip-flops into CLBs. Finally, we perform placement and dual-rail routing by our *VPDR*. The experimental result is reported in Table I.

In Table I, we compare the performance improvement and the area overheads of different approaches. Column *Original* represents the placement and routing results from *VPR*. We use the results of *Original* as a baseline. Column *VPR+D* represents the results of dual-rail wire insertion for those with slack less than and equal to 15% of *circuit delay*. That is we perform dual-rail insertion for all candidate wires described in Figure 4. Column *VPDR* represents the results of our proposed algorithm, where we use $k = 0.155$ in our experiment. Columns $Track_h$ and $Track_v$ are the number of tracks in horizontal and vertical channels, respectively. Column *Delay* is the wire delay. Columns Ov_h, Ov_v, and *Timing* are the horizontal and vertical routing area overheads and timing improvement as compared with the baseline, respectively. From Table I, we can see that our proposed algorithm achieves 11.38% of timing improvement with routing area overheads 5.79% in horizontal channel and 4.35% in vertical channel.

V. CONCLUSION

Dual-rail routing was proposed to improve the circuit performance of a uniform routing architecture in modern VLSI design. In this paper, we present a technique to insert dual-rail wires for pre-fabricated design styles. Furthermore, we propose an effective dual-rail insertion algorithm to reduce the routing area overheads caused by the inserted dual-rail wires. Taking *wire criticality*, *delay significance*, and *wire congestion* into consideration, our proposed algorithm is capable of trading routing area overheads for the interconnection performance improvement. The experimental results demonstrate that our proposed algorithm reduce the interconnection delay by 11.4% with about 5.8% routing area overheads.

REFERENCES

[1] C. Patel, A. Cozzie, H. Schmit, L. Pileggi, "An architectural exploration of via patterned gate arrays", *in Proceedings of International Symposium on Physical Design*, pp.184-189, April 2003.

[2] Y. Ran, M. Marek-Sadowska, "Via-configurable routing architectures and fast design mappability estimation for regular fabrics", *in Proceedings of International Conference on Computer-Aided Design*, pp.25-32, May 2005.

[3] V. Betz, J. Rose, A. Marquardt, "Architecture and CAD for deep-submicron FPGAs", Kluwer Academic Publishers, 1999.

[4] D. Chen, J. Cong, P. Pan, "FPGA design automation", Now Publishers, 2006.

[5] F. W. Chen, Y. Y. Liu, "Wire sizing alternative - an uniform dual-rail routing architecture", *in Proceedings of Design, Automation and Test in Europe*, pp.796-799, March 2008.

[6] M. Tom, G. Lemieux, "Logic block clustering of large designs for channel-width constrained FPGAs", *in Proceedings of Design Automation Conference* pp.726-731, June 2005.

[7] Y. W. Chang, D. F. Wong, C. K. Wong, "Universal switch modules for FPGA design", *in ACM Transactions on Design Automation of Electronic Systems* pp.80-101, vol.1, i.1, Jan 1996.

[8] H. Fan, J. Liu, Y. L. Wu, C. C. Cheung, "On optimum switch box designs for 2-D FPGAs", *in Proceedings of Design Automation Conference* pp.203-208, 2001.

[9] V. Betz, J. Rose, "Directional bias and non-uniformity in FPGA global routing architectures" *in Proceeding of International Conference on Computer-Aided Design*, pp.652-659, 1996.

[10] V. Betz, J. Rose, "FPGA routing architecture: segmentation and buffering to optimize speed and density", *in Proceedings of International Symposium on Field Programmable Gate Arrays* pp.59-68, Feb 1999.

[11] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis", *Electronics Research Laboratory*, Memorandum No. UCB/ERL M92/41, 4 May 1992.

[12] J. Cong, Y. Ding, "Flowmap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs", *in IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, pp.1-12, vol.13, i.1, Jan. 1994.