Divided Backend Duplication Methodology for Balanced Dual Rail Routing

Karthik Baddam and Mark Zwolinski

Electronics Systems and Devices Group,
School of Electronics and Computer Science,
University of Southampton,
SO17 1BJ, UK
{kb04r,mz}@ecs.soton.ac.uk
http://www.esd.ecs.soton.ac.uk/

Abstract. Dual Rail Precharge circuits offer an effective way to address Differential Power Analysis Attacks, provided routing of differential signals is fully balanced. Fat Wire [1] and Backend Duplication [2] methods address this problem. However they do not consider the effect of coupling capacitance on adjacent differential signals. In this paper we propose a new method, Divided Backend Duplication, which is based on Divided Wave Dynamic Differential Logic [3] and Backend Duplication [2], that effectively addresses balanced routing problem of Dual Rail Precharge circuits. Experimental results on an AES test circuit in 130nm technology show improvements in achieving a balanced dual rail design. Further our method can also be successfully applied to FPGAs. Results from an sbox test circuit implementation on a Xilinx FPGA are presented.

Keywords: Differential Power Analysis, Dual Rail Routing, Dual Rail FPGA Implementation.

1 Introduction

Security is an important and often primary design goal in embedded systems such as smart-cards [4] sidelining other design parameters such as cost, performance and power consumption. Differential Power Analysis Attack (DPA) [5] pose a serious threat to secure embedded systems such as smart-cards. As a result, researchers have developed several DPA countermeasures [3,6,7,8,9,10,11]. Of these, the logic level countermeasures that fall under Dynamic and Differential logic (also referred to as Dual Rail Precharge - DRP) style, theoretically offer more resistance to DPA. The basic principle behind DRP logic is to eliminate any information leaks, by consuming the same amount of power in every clock cycle. DRP circuits have been proved to prevent DPA, provided the routing of differential nets is balanced [12].

Balancing differential nets (balanced Dual Rail routing) is not, however, a trivial task. To address the routing problem, to date the following proposals have been put forward: DWDDL [3], FatWire [1], Backend Duplication [2], Three

Phase Dual Rail [13], Path Switching [9], Double WDDL [14] and an iterative correction flow [15]. Of these, three proposals [1,2,3] impose some constraints on backend implementation flows. Three Phase Dual Rail [13] tries to avoid the routing problem by introducing a third phase, which is an additional overhead. Path Switching [9] offers an improvement to dual rail circuits and only protects registers and buses with high capacitance. Double WDDL, as the name implies, has two separate WDDL implementation thereby increasing the area overheads by four times. Double WDDL was developed mainly for use in FPGAs [14]. The first WDDL part is implemented using normal place & route flow. The second WDDL part is obtained by copying the first WDDL part, including the routing details, and reversing the orginal and complementary logic [14]. Backend correction flow, described in [15], is iterative and can consume a significant amount of time to implement a design.

In this paper we concentrate on the implementation of balanced Dual Rail Precharge logic styles rather than the alternatives. We try to present a simple yet effective solution to improve Dual Rail circuit routing capacitance. In Section 2 we discuss Dual Rail Precharge Logic Styles, give a brief introduction to backend design flow, and discuss existing methods and their shortcomings. In Section 3 we present the inversion problem and discuss its solutions. In Section 4 we present our proposed methodology. In Section 4.1 & Section 4.2 we present ASIC & FPGA implementations respectively and then conclude the paper.

2 Background

2.1 Dual Rail Precharge Logic Styles

Dynamic and Differential Logic (also referred to as Dual Rail Precharge - DRP) [3,7,8] has been proposed to prevent DPA. The idea is to consume the same amount of power for any combination of inputs. This is achieved by using differential logic (two signals instead of one) and by precharging both the differential nets in every clock cycle. In DRP circuits for every logic gate, a complementary gate exists, usually referred to as *false* logic (or *false* part).

Dual Rail Precharge logic styles can be classified into two types based on the way precharge is applied. Sense Amplifier Based Logic (SABL) is a DRP logic based on the principles of domino logic, where a special precharge signal is applied to every gate to force the gate to precharge. Wave Dynamic Dual Rail (WDDL) and Dual Spacer Dual Rail (DSDR) on the other hand propagate the precharge signal from a design's primary inputs and state-elements (flip-flops). WDDL and DSDR have the following differences over SABL: 1) WDDL and DSDR can be constructed using existing CMOS standard cells and 2) that the *true* logic and *false* logic are two different cells. The second point is not true in all cases. WDDL and DSDR both need special inverters, where the *true* and *false* wires are cross connected. As differential logic has both *true* and *false* outputs, an inverter is implemented by exchanging the outputs. Moreover an inverter is an inverting gate, it will stop the precharge wave propagation. Fig. 1 shows the basic building blocks of WDDL with master slave WDDL flip-flops. Although

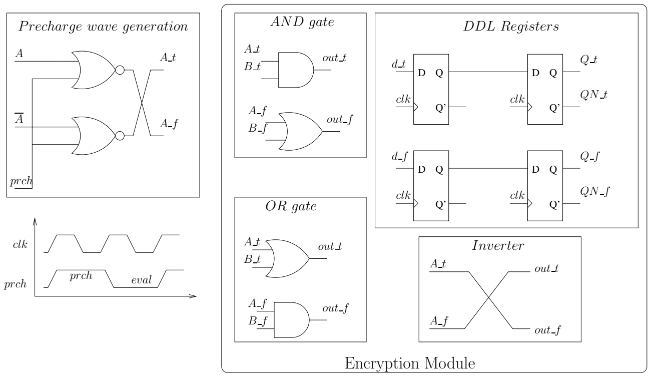


Fig. 1. Building blocks of WDDL,with Master Slave WDDL flip-flops

double the clock frequency is required to get same data rate using master slave flops, these are recommended [3]. All primary inputs are driven by a 'precharge wave generation' block, so that individual gates will propagate the precharge. Note that the inverter is implemented by exchanging the dual rail pairs.

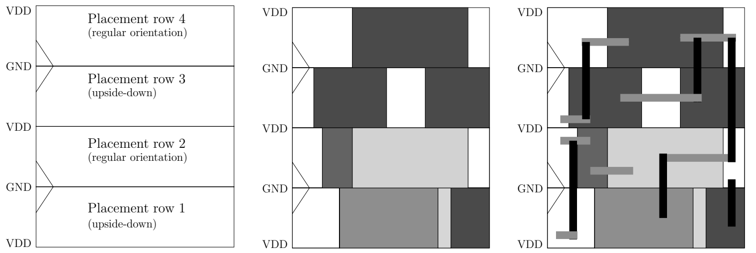
2.2 Backend Design Flow

Most of the digital designs implemented today are based on a standard cell flow. A set of commonly used standard cells are designed and characterized such that CAD tools can be used to automate most of the design flow. Design entry is typically in behavioral HDL and is synthesized and mapped to the target technology's standard cells. After the synthesis, the resulting netlist is placed and routed to get the final design. Backend design is usually referred to the implementation of the design after the synthesis phase and mainly involves floorplanning, placement and routing. A placer partitions the available core area into rows, where the standard cells are placed. In a similar fashion, a router partitions the core area into horizontal and vertical routing grids. Each grid has a minimum size defined by the target technology's wire pitch size.

The place and route flow usually involves the following steps, shown in Fig. 2. First a floorplan is made (Fig. 2(a)). This is where the aspect ratio (or the dimensions) of the chip are determined. Next the standard cells are placed (Fig. 2(b)) and finally the wires are routed (Fig. 2(c)).

2.3 Existing Methods

Divided Wave Dynamic Differential Logic (DWDDL) was proposed by *Tiri and Verbauwhede* [3] to address routing imbalances in DRP logic styles. DWDDL's idea is to place and route a single ended design (the *true* part), copy it and replace the complementary cells (for example 'and' with 'or' and vice versa) to get the *false* part. However, this method assumes that there is no inversion in the single rail design, as an inverting cell would stop the precharge wave propagation. However, in practice it is difficult to have logic without inversion. This is



(a) Normal Backend flow: Create a Floor plan (b) Normal Backend flow: Place the standard cells (c) Normal Backend flow: Route the wires

Fig. 2. Normal Backend flow overview

the only known limitation for DWDDL and no further work has been reported on it.

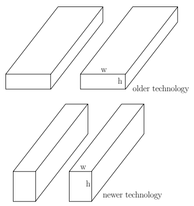
Fat Wire was proposed by *Tiri and Verbauwhede* [1] to address routing imbalances in DRP logic styles. In this methodology a Fat Wire is constructed by two adjacent normal wires. For the Fat Wire method to work, first the dual rail netlist, instantiating dual rail cells, has to be placed. Then instead of routing two differential wires (for the *true* and *false* signals) a single Fat Wire is routed and later decomposed into two normal single wires which will have same wire length.

Backend Duplication was proposed by *Guilley et al.* [2] to address routing imbalances in DRP logic styles. The basic idea of backend duplication is based on placement and routing obstructions (constraints to the CAD tool). The first step of Backend Duplication is to constrain the CAD tool (1) to only use alternate rows for placing cells and routing horizontal routes (2) and to use the alternate routing pitches for routing vertical routes. Thus, when the placer has finished placing the single rail design, a dual rail design can be obtained from copying (and transforming) the single rail into the previously obstructed rows. Note that this operation is a simple shift in coordinates of the placed cells. Duplicating the routes is done in two steps. Once the design is routed, horizontal routes are duplicated in the same way as cells. Vertical routes are duplicated by simple shift in the x-axis of the routing pitch.

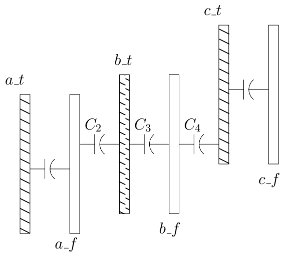
2.4 Shortcomings of the Existing Methods

Coupling capacitance (crosstalk) has become one of the most critical issues in deep sub micron physicaql designs because of 1) interconnect dominated circuit delay and 2) strong coupling effects between intqerconnect wires [16]. As technology scales the wire widths, their height is increased and coupling capacitance between wires increases [16] (Fig. 3(a)).

In the Fat Wire and Backend Duplication methods (vertical routes) dual rail wires end up next to each other, as shown in Fig. 3(b). With coupling capacitances increasing, the effective capacitance seen by a *true* and *false* signal will vary. For



(a) Effect of shrinking wire widths on coupling capacitance



(b) Coupling Capacitance in Dual Rail circuit

Fig. 3. Coupling Capacitance effects

example consider dual rail pairs *b_t* & *b_f*. The coupling capacitance seen by *b_t* is C_2 & C_3 whereas the coupling capacitance seen by *b_f* is C_3 & C_4. Now if the capacitances C_2 & C_4 vary by a huge difference, the resulting design can have unbalanced wire capacitance and can lead to information leaks. Note that this effect becomes more and more dominant as technology scales down. The effect of coupling between differential wires is more significant in the Fat Wire method than in Backend Duplication as the horizontal wires are also next to each other.

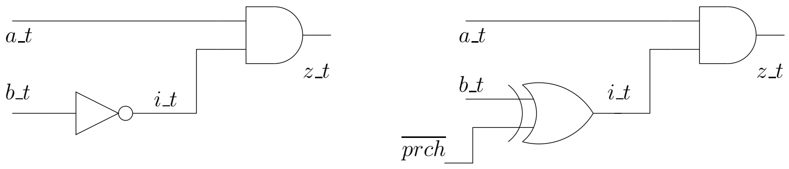
Of course spacing between dual rail wires can always be increased to reduce the coupling capacitance, however such an increase comes at the expense of increased area and reduced routing resources. Of the three methods to address routing problems, DWDDL is the simplest and most effective. However practical designs will always have inversion and hence will not be able to use the DWDDL method.

3 Inversion Problem in DRP Logic

Inversion in Dual Rail Precharge Logic styqles is considered as a free operation, as dual rail signal pairs are coqmplementary; inversion is simply obtained by exchanging the dual rail pairs. On the other hand an inverter cannot exist in a WDDL or DSDR style design as it would stop the precharge wave propagation. In other words, inversion is only possible by exchanging the dual rail pair. This property of WDDL and DSDR logic styles prevents designs from using a DWDDL style of implementation. Of course dual rail pairs can be exchanged after DWDDL implementation, but there is no systematic way of doing this. Moreover the extra wire capacitance from this exchange can add to the critical path delay of a design and can introduce unbalanced wires. This issue of exchanging wires can be worst when the number of unused inverters in a design increases. As an example a 8ns clock period, 128 bit AES had 5,762 inverters from a total gate count of 22,704, excluding buffers used for the clock tree. For this example, we increased the area and delay cost of the original inverter by 10 times so that synthesis tool will use it only when inversion is needed and not for buffering.

3.1 Mitigating the Inversion Problem in DRP Logic

Inverters cannot exists in WDDL and DSDR style designs as they would stop the precharge wave propagation. On the other hand, designing logic without inversion is difficult. It is possible to have a cell that behaves as an inverter and still not prevent the precharge wave propagation. This is possible by using a two input Exclusive-OR (XOR) gate instead of an inverter and connecting the second input of XOR to the negated precharge signal that is used in generating the precharge wave (Fig. 1).



a_t	b_t	i_t	$prch$	z_t
0	0	1	1	0
1	1	0	1	0
0	0	1	0	0
0	1	0	0	0
1	0	1	0	1
1	1	0	0	0

a_t	b_t	i_t	$prch$	z_t
0	0	0	1	0
1	1	1	1	1
0	0	1	0	0
0	1	0	0	0
1	0	1	0	1
1	1	0	0	0

Fig. 4. Using XOR instead of Inverter (Inputs a_t & a_f are driven by Precharge Wave Generation Block shown in Fig. 1)

Consider the example circuit on the left side of Fig. 4, with the truth table shown. When the *prch* signal is high all primary inputs are set to logic 1 (Inputs a_t & a_f are driven by Precharge Wave Generation Block shown in Fig. 1). However intermediate signal i_t (output of the inverter) will not propagate the precharge wave and the output signal z_t will not be precharged. Now consider the circuit on the right of Fig. 4. A two input Exclusive-Or (XOR) gate is used instead of an inverter. The original input and output of the inverter are connected as before to the XOR. The second input of the XOR is connected to the $\overline{prch}$ signal, which is used in precharge wave propagation. When *prch* is high the XOR will act as a buffer allowing the precharge wave to propagate and when *prch* is low XOR will act as an inverter as intended in the original circuit.

It is also possible to use a Domino-style inverter (similar to the one presented in [13]) instead of an XOR gate. As in the case of the XOR, $\overline{prch}$ is used to precharge the domino-inverter. In the case of a domino style inverter, the timing of $\overline{prch}$ is important for the circuit to work. Because of this, we prefer to use an XOR gate and in the rest of this paper we use XOR gates to replace inverters. Note that inverters that are used in clock tree synthesis need not be replaced, as the clock signal is not precharged like normal inputs. Based on this, we now present a method to implement a fully balanced dual rail design.

4 Proposed Method: Divided Backend Duplication

With XOR gates replacing inverters, a dual rail circuit can be implemented as physically separate (without any connections) *true* part (original single-ended part) and *false* part (complementary part). The primary inputs and outputs will still remain common for both the *true* and *false* parts. With this advantage the Divided WDDL implementation, [3], can now be implemented provided that 1) the pins of complementary standard cells should be same, i.e at same location and same metal layer and 2) the size of the complementary standard cells are the same.

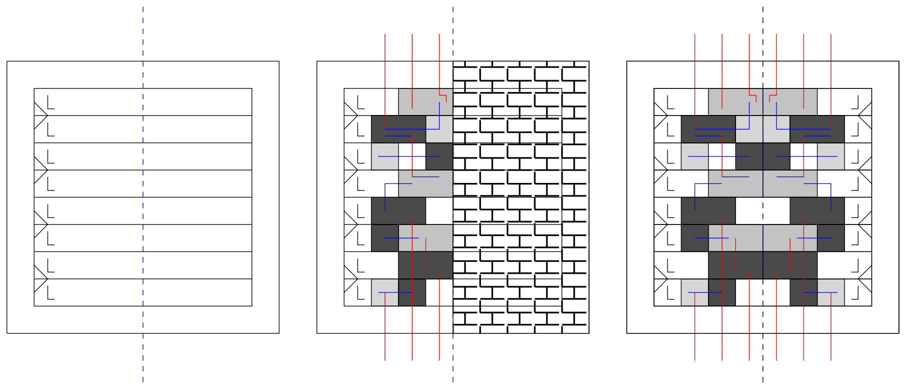

(a) Initial Floor Plan (b) Reserve space for duplicating complimentary logic (c) Flip every object (cells & routes) to right

Fig. 5. Proposed method overview

Fig. 5 shows the overview of our proposed method for balanced dual rail routing. This method is similar to the Backend Duplication method, [2]. A single ended design is used for the initial place and route process and then duplicated to get the final dual rail design. The process can be divided into the following steps (shown in Fig. 6).

1. A WDDL-compliant single rail design is processed to replace the inverter cells with XOR cells (Fig. 4). A program has been written for this conversion, based on OPENACCESS [17]. At this stage the design is still single rail.
2. A floorplan is made for the processed single rail design, with utilization of half the required final utilization. This ensures that there is enough space for duplicating the complementary part (Fig. 5(a)).
3. Half of the floorplan area is reserved (obstructed) for the complementary part (Fig. 5(b)).
4. The Single Rail design is implemented in the usual way, i.e place and route, timing analysis, SI analysis, ECO fixes, etc.

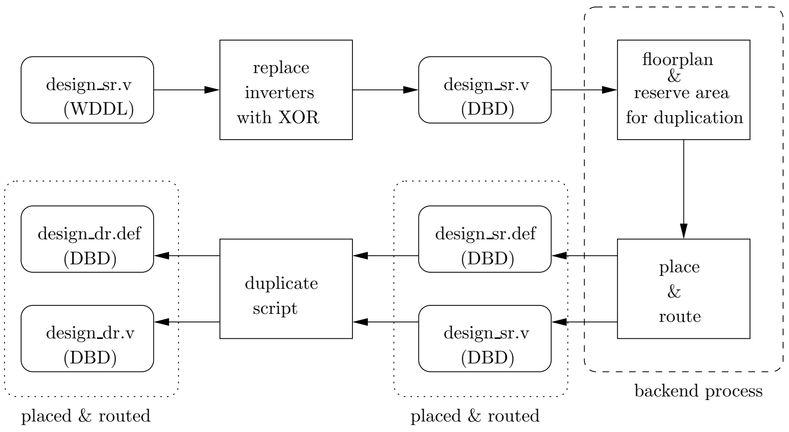


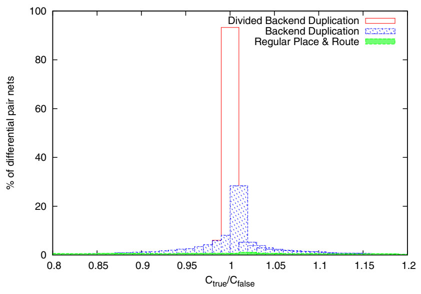
Fig. 6. Divided Backend Duplication implementation overview

5. After the Single Rail design is finalized, the complementary part can be obtained by flipping every object in the single rail design to the right and by replacing the complementary cells, *AND* with *OR* and vice versa, as shown in Fig. 5(c). This step can be done by processing the DEF file and is similar to the process used in Fat Wire [1] and Backend Duplication [2].

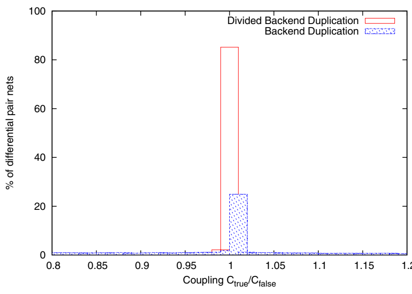
As our proposed method is derived from DWDDL and Backend Duplication, we call it Divided Backend Duplication (DBD). A small variation to the duplication process can be made: 1) Instead of flipping the design objects to right, they can be shifted by half of the core width. 2) Instead of flipping the design objects along the x-axis, this can be done on the y-axis too (flipping to top or bottom).

4.1 ASIC Implementation

To show the effectiveness of Divided Backend Duplication, we implemented an AES test circuit with 20k+ gates in a 130nm process. Three different designs are implemented. All designs have the same constraints and netlist. The difference is in implementation. The first implementation, which we call "regular place & route design", is implemented without any special techniques. The second implementation, which we call "backend duplicated design", is implemented as suggested in [2] and is based on the WDDL logic style [3]. The third design, which we call "divided backend duplicated design", is implemented as suggested in Section 4 and is also based on WDDL logic style [3]. All the designs aspect ratios are set to 1. The row utilization of "regular place & route design" is set to 0.70 while for "backend duplicated design" and "divided backend duplicated design" it is set to 0.35 (half the required utilization, so that enough room is available for duplication). We used Cadence Encounter tools [18] to perform the backend implementation. For parasitic extractions we used Encounter's



(a) Ratio of Total Capacitance of Differential Pair nets



(b) Ratio of Coupling Capacitance of Differential Pair nets

Fig. 7. Ratio of Capacitance of Differential Pair nets

native extractor and set the "detailed" and "coupling" switches to true. After the parasitic extraction, all the parasitic information was exported into a Standard Parasitic Exchange Format (SPEF) file containing the ground capacitance, coupling capacitance and resistance of every wire.

Fig. 7 shows histograms in which the internal interconnect capacitance of the regular place and route design, the Backend Duplicated design and Divided Backend Duplicated (DBD) design are compared. We have not implemented Fat Wire [1] as the effect of coupling on dual rail signal pairs from Fat Wire should be similar to that of the Backend Duplication method [2]. The capacitance per net was extracted from the SPEF file, which in turn was reported from Encounter. Fig. 7(a) shows the distribution of the ratio between the capacitance at the *true* signal net and the capacitance at the corresponding *false* signal net (C_{true}/C_{false}). The ratio C_{true}/C_{false} for regular place & route method is between 0.01 & 10 and for the backend duplication method it is between 0.70 & 1.5. On the other hand, for the divided backend duplication method this ratio is only between 0.90 & 1.1. The percentage of nets that have a ratio of 1 for Divided Backend Duplication is 93.25% when compared to 28.34% for backend duplication.

Fig. 7(b) is similar as Fig. 7(a) except that coupling capacitance is only considered instead of total capacitance. The cumulative coupling capacitance per net was extracted from SPEF file, which in turn was reported from Encounter. Coupling capacitance ratio, $Coupling\ C_{true}/C_{false}$ for regular place & route method are not shown as the ratio for some nets was as high as 70. For the backend duplication method, the ratio $Coupling\ C_{true}/C_{false}$ is between 0.22 & 3.52 while for divided backend duplication is 0.60 & 1.9. The percentage of nets that have a ratio of 1 for Divided Backend Duplication is 85.15% when compared to 24.86% for Backend Duplication. As discussed in Section 2.4, this increase in capacitance ratio for Backend Duplication method is due to unevenly distributed coupling capacitance, whereas the Divided Backend Duplication method shows much less variation.

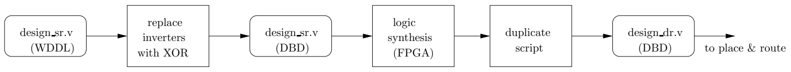
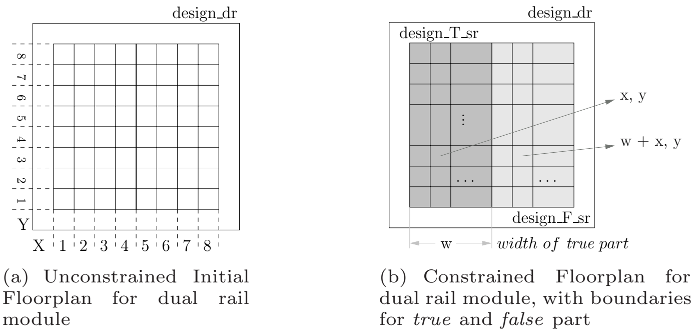


Fig. 8. Divided Backend Duplication Synthesis for FPGAs



(a) Unconstrained Initial Floorplan for dual rail module

(b) Constrained Floorplan for dual rail module, with boundaries for *true* and *false* part

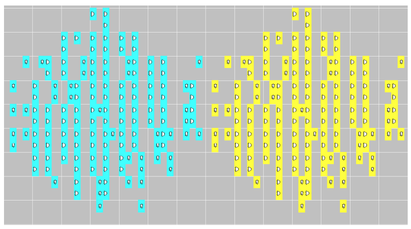
Fig. 9. Floorplanning to implement Divided Backend Duplication Dual Rail design on FPGAs

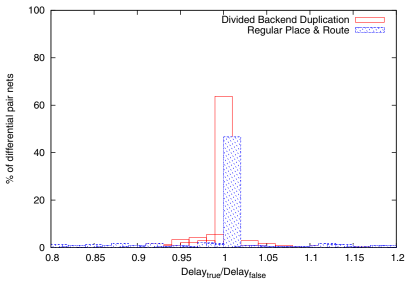
4.2 FPGA Implementation

Differential routing on FPGAs is more difficult than on ASICs as the routing resources are limited. *Tiri and Verbauwhede* [19] have discussed a WDDL implementation on FPGAs and proposed a synthesis flow. However, the differential routing problem in FPGAs has not been addressed to the best of our knowledge. In this section we discuss how the Divided Backend Duplication method can be applied to get balanced differential routing in FPGAs.

Before implementing a design in FPGA, it has to be synthesized to the target FPGA. Synthesizing for a secure dual rail implementation has been discussed in detail in [19]. We adopt the flow presented in [19] to synthesize for Divided Backend Duplication implementation with the modifications shown in Fig. 8. After replacing the inverters with XORs, FPGA synthesis can be done with a commercial CAD tool or "Clustering" technique described in [19]. Care needs to be taken if Commercial CAD tools are used, to preserve the wave dynamic nature of the design. Note that the structural *true* and *false* part are identical for FPGAs, the only difference being the LUT programming value.

FPGAs have highly regular structure as shown in Fig. 9(a). Each box in Fig. 9(a) corresponds to a Configurable Logic Block (CLB) and its associated routing resources. Unlike ASICs, the place & route process of FPGAs is not standardized. This makes it difficult to duplicate the placement and routing information for complementary parts of a dual rail design. Although each FPGA vendor has a specific implementation tool, most of the tools offer procedures to 1) floorplan and 2) constrain a design's instance to a specific location. However, constraining a net to a specific routing resource is not supported. Based on this,


(a) Floorplan view of duplicated design on a Xilinx FPGA



(b) Ratio of Delay of Differential Pair nets

Fig. 10. Divided Backend Duplication implementation results on a Xilinx FPGA

the process to implement a balanced dual rail design in FPGAs can be divided into the following steps.

1. The WDDL-compliant single rail design is processed to replace the inverter cells with XOR cells and to transform the netlist into a FPGA-specific netlist (Fig. 8).
2. The floorplan area is divided into two equal parts (for the *true* and *false* parts), comprising equal number of CLBs, local routing resources and global routing resources (Fig. 9(b)).
3. The top-level dual-rail design is implemented in the usual way, without violating the boundary constraints set above. The implementation steps usually are place & route, timing analysis, ECO fixes, etc.
4. After the top-level dual-rail design is successfully implemented, locations of all the instances of *true* part are saved to a file. Based on the location of a *true* part's instance, the corresponding *false* part's instance is calculated and written to a constraint file.
5. Based on the new constraints, the *false* part is re-implemented.

To see the effectiveness of backend duplication, we implemented a DES sbox on a Xilinx FPGA [20]. Xilinx's XST tool was used for synthesis and ISE was used for implementation. The Xilinx Floorplan editor was used to constrain the floorplan. After the initial place & route Xilinx's Floorplan editor was used to save all the instance locations. The final place & route process was constrained by using Xilinx's UCF file. Fig. 10(a) shows a floorplan view of such a duplicated design. Although FPGA implementation tools do not report detailed parasitic information, they report delays associated with an instance and interconnect in a Standard Delay File (SDF). This SDF file was analyzed and the resulting distribution of the ratio between the delay at the *true* signal net and the delay at the corresponding *false* signal net ($Delay_{true}/Delay_{false}$) is shown in Fig. 10(b). The delay ratio $Delay_{true}/Delay_{false}$ for the regular place & route method is between 0.40 & 2.7 and for the divided backend duplication method it is between 0.8 & 1.2. The percentage of nets that have a ratio of 1 for Divided Backend Duplication is 64.25%

compared to 46.34% for regular place & route. Although we have constrained an instance to be at a specific location, the implementation tool is free to connect the wires and may be the reason for only 64.25% of nets to have a ratio of 1. Note that we are not constraining the FPGA tool to duplicate the routes, as we could not find a way to achieve this. *Yu and Schaumont* have implemented a duplication method for Double WDDL style on Xilinx FPGAs [14] that can be used to completely balance the routing of differential nets on FPGAs.

4.3 Advantages of Divided Backend Duplication

The main advantage of Divided Backend Duplication is that both the *true* and *false* parts see the same environment. The coupling capacitance problem discussed in Section 2.4 is now eliminated. As Divided Backend Duplication is based on standard cells implementation styles such as WDDL and DSDR, it can be adapted to both ASICs and FPGAs.

Divided Backend Duplication will not have a problem with diagonal routing, an upcoming interconnect technology (already available in Xilinx FPGAs and supported by the Cadence X architecture router), whereas Backend Duplication currently cannot handle it. Implementing Divided Backend Duplication process is a straightforward process. Neither specific design rules need to be changed nor specific routing blocks have to be imposed on the design. In our example implementation for ASIC, the run time was 3 times less when compared to Backend Duplication. As the *true* and *false* parts are not interleaved, implementing any Engineering Change Orders (ECOs) is also simple and straightforward.

The only requirements to implement Divided Backend Duplication are that 1) the pins of complementary standard cells should be same, i.e. at same location and same metal layer and 2) the size of complementary standard cells are the same. This is an advantage when compared to the requirements imposed by Fat Wire [1] and Backend Duplication [2].

As Divided Backend Duplication separates the *true* and *false* part, a by-product is that two separate data sets can be processed at the same time, instead of one. Divided Backend Duplication designs can have a *random mode* where one part can process the required data and the other can process random data. Further the entire design can be configured such that the design can randomly switch from *dual rail mode* to *random mode* and back. Divided Backend Duplication designs can even be configured to operate either the *true* or *false* part at a given time to reduce power consumption, when DPA countermeasure is not required. The only requirement to achieve this is to change the input/output interface to the dual rail design.

4.4 Disadvantages of Divided Backend Duplication

The main disadvantage of the Divided Duplication method is the additional area and delay overhead introduced by replacing inverters with XOR gates. The number of XOR cells used depends on the design and cannot be generalized. For our AES test circuit about 25% of cells were XORs. This increased the critical

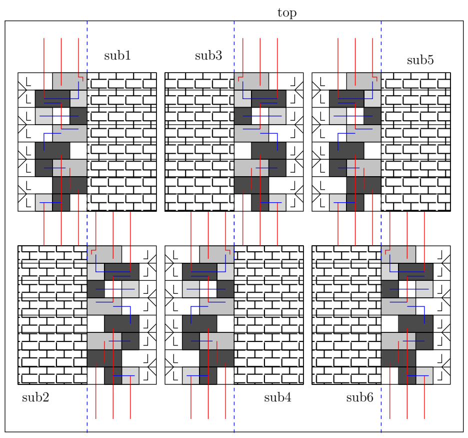


Fig. 11. Hierarchical Divided Backend Duplication

path delay by 1.2 times. The delay and area overhead introduced by XOR can be minimized by using a domino style inverter instead of XOR. Also the *prch* signal needs to be buffered as it drives all the extra XOR cells.

As the *true* and *false* part of the design are physically separated, there may be a concern that EM analysis attacks [21] may be successful, by only observing the *true* or *false* part. Although this may seem unlikely, one may minimize the extent of this concern by taking a hierarchical approach to implementing Divided Backend Duplication compared with that shown in Fig. 6. An example floorplan for a hierarchical Divided Backend Duplication is shown in Fig. 11. Another approach would be to use the Backend Duplication method [2], but with the following difference for duplication: instead of shifting to the right, every object can be flipped to the right.

5 Conclusion

We have shown that coupling capacitance between dual rail nets can cause routing imbalances. To address this, we have proposed a new method, called Divided Backend Duplication. We have shown that the Divided Backend Duplication method can be applied to get a balanced dual rail design in both ASICs and FPGAs and that it offers a significant improvement in balancing routing capacitance compared to previous methods. Divided Backend Duplication is the first method to address routing imbalances in FPGAs. Divided Backend Duplication has an area overhead of around 25% and a delay overhead of around 1.2 times.

Acknowledgments

We are very thankful to Sylvain Guilley for his valuable discussions & suggestions regarding implementation of Backend Duplication and for sharing his Perl script to implement Backend Duplication. We would also like to acknowledge the anonymous referees for their valuable comments & suggestions.

References

1. Tiri, K., Verbauwhede, I.: Place and Route for Secure Standard Cell Design. In: 6th International Conference on Smart Card Research and Advanced Applications (CARDIS 2004), August 2004, pp. 143–158 (2004)
2. Guilley, S., Hoogvorst, P., Mathieu, Y., Pacalet, R.: The Backend Duplication Method. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 383–397. Springer, Heidelberg (2005)
3. Tiri, K., Verbauwhede, I.: A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In: DATE 2004: Proceedings of the conference on Design, automation and test in Europe, pp. 246–251. IEEE Computer Society, Washington (2004)
4. Ravi, S., Raghunathan, A., Kocher, P., Hattangady, S.: Security in embedded systems: Design challenges. Trans. on Embedded Computing Sys. 3(3), 461–491 (2004)
5. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
6. Bucci, M., Guglielmo, M., Luzzi, R., Trifiletti, A.: A power consumption randomization countermeasure for DPA-resistant cryptographic processors. In: Macii, E., Paliouras, V., Koufopavlou, O. (eds.) PATMOS 2004. LNCS, vol. 3254, pp. 481–490. Springer, Heidelberg (2004)
7. Sokolov, D., Murphy, J., Bystrov, A., Yakovlev, A.: Design and Analysis of Dual-Rail Circuits for Security Applications. IEEE Transactions on Computers 54(4), 449–460 (2005)
8. Tiri, K., Verbauwhede, I.: Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 125–136. Springer, Heidelberg (2003)
9. Baddam, K., Zwolinski, M.: Path switching: a technique to tolerate dual rail routing imbalances. Design Automation for Embedded Systems (accepted for publication) (2008), `http://www.springerlink.com/content/32181g28411w2121`, doi:10.1007/s10617-008-9017-z
10. Pramstaller, N., Oswald, E., Mangard, S., Gürkaynak, F.K., Haene, S.: A Masked AES ASIC Implementation. In: Ofner, E., Ley, M. (eds.) Proceedings of Austrochip 2004, Villach, Austria, October 2004, pp. 77–82 (2004)
11. Popp, T., Mangard, S.: Masked Dual-Rail Pre-Charge Logic: DPA-Resistance without Routing Constraints. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 172–186. Springer, Heidelberg (2005)
12. Tiri, K., Verbauwhede, I.: Prototype IC with WDDL and Differential Routing DPA Resistance Assessment. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 354–365. Springer, Heidelberg (2005)
13. Bucci, M., Giancane, L., Luzzi, R., Trifiletti, A.: Three-Phase Dual-Rail Pre-charge Logic. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 232–241. Springer, Heidelberg (2006)
14. Yu, P., Schaumont, P.: Secure FPGA circuits using controlled placement and routing. In: CODES+ISSS 2007: Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis, pp. 45–50. ACM, New York (2007)

15. Bouesse, G.F., Renaudin, M., Dumont, S., Germain, F.: DPA on Quasi Delay Insensitive Asynchronous Circuits: Formalization and Improvement. In: DATE 2005: Proceedings of the conference on Design, Automation and Test in Europe, pp. 424–429. IEEE Computer Society, Washington (2005)
16. Weste, N., Harris, D.: CMOS VLSI Design A Circuits and Systems Perspective, 3rd edn. Addison-Wesley, Reading (2004)
17. Si2.org: OpenAccess Coalition (April 2007), `http://openeda.si2.org/`
18. Cadence Design Systems: ENCOUNTER DIGITAL IC DESIGN PLATFORM (April 2007), `http://www.cadence.com/products/digital_ic/index.aspx?lid=dic`
19. Tiri, K., Verbauwhede, I.: Synthesis of Secure FPGA Implementations. In: International Workshop on Logic and Synthesis (IWLS 2004), June 2004, pp. 224–231 (2004)
20. Xilinx Inc: Xilinx Inc. (April 2007), `http://www.xilinx.com/`
21. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic Analysis: Concrete Results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)