Balancing WDDL Dual-Rail Logic in a Tree-based FPGA to Enhance Physical Security

Emna Amouri, Shivam Bhasin, Yves Mathieu, Tarik Graba, Jean-Luc Danger
Institut TELECOM / TELECOM ParisTech, CNRS LTCI (UMR 5141)
Departement COMELEC, 46 rue Barrault, Paris, France
firstname.lastname@telecom-paristech.fr

Habib Mehrez
LIP6, Universite Pierre et Marie Curie
4, Place Jussieu, Paris, France
firstname.lastname@lip6.fr

Abstract—The Tree-based FPGA offers better density and timing determinism than traditional mesh-based FPGA. Moreover, thanks to its multilevel structure, it offers greater easiness to balance dual signals in terms of routing resources number. In this paper, we study the use of the Wave Dynamic Differential Logic (WDDL) on a custom tree-based FPGA of 2048 cells. The WDDL technique offers an effective way to withstand Differential Power Attacks (DPA). However, the effectiveness of this countermeasure is guaranteed provided a symmetry is maintained between the routing of the both the direct and complementary paths, which is very hard to achieve in FPGA. Thus, balancing aware Computer-Aided Design (CAD) tools must be developed. In this work, we propose first adjacent placement and balancing-aware routing techniques for tree-based FPGA to counter the routing unbalance. Then side channel analyses are performed on FPGA circuit implementing PRESENT crypto-processor. Experimental results show that the balancing methods enhance the design security against side channel attacks.

Index Terms—placement, routing, balance, Tree-based FPGA, Differential Power Analysis, WDDL, security.

I. INTRODUCTION

Side Channel Attacks (SCAs) present a serious threat to implementations of cryptographic algorithms. They rely on the data-dependent power consumption variation in order to extract the secret key.

During the last years, many countermeasures have been proposed to protect cryptographic devices against SCA. One approach, called Dual-Rail Precharge Logic (DPL), appears to be one of the most promising techniques. The principle of hiding consists in consuming the same amount of power consumption regardless of data inputs. This is achieved by using differential logic (signals are encoded as two complementary wires), and pre-charging the differential signals in every clock cycle. It is also called Dual-Rail Precharge logic (DPL). Several implementations of secure dual rail cells have been proposed, specifically for ASICs, such as SABL [3], WDDL [1], STTL [4], MDPL [2] and BCDL [5].

The Wave Dynamic Dual Rail Logic (WDDL) technique, developed by K. Tiri [1], is the most popular DPL countermeasure. In WDDL design, the netlist is duplicated into a true and a false part. The components used are limited to positive rail signals, in WDDL design, without adding logic to balance dual networks, but rather control the placement and the routing of the differential design to avoid area increase. The FPGA we are targeting is a multilevel hierarchical FPGA which is a novel Tree-based architecture (TFPGA) presented in [11]. It was shown that this architecture is better in terms of area density as compared to common VPR-style MESH architecture. The architecture interconnect is described in section II. Then, in section III, we propose an adjacent placement, a resources_balance_driven and a timing_balance_driven routing algorithms to target balance improvement of dual nets. Finally, we perform side channel analyses on TFPGA circuit and present the experimental results obtained with Present [12] cryptographic algorithm.
Driven Routing Algorithm

Encounter tool, are inserted in the routing tool. Thus, the algorithm [13] which is an iterative rip-up algorithm.

phase, each cluster is assigned to a position inside its owner connections between partitions. Then, during the placement objective of our partitioning is to minimize the number of until the lowest level of the architecture is reached. The cursive partitioning phase. First, the top level clusters are

with Encounter tool.

timing informations of FPGA routing resources are extracted with Encounter tools and fabricated with ST 65

and arity 8x8x8x4 (2048 LBs), was designed using SOC A prototype of this architecture, with 4 levels of hierarchy and

all LBs from different paths.

As shown in Figure 1, input pads and output pads are con-

ected to US and DS respectively. Thus, input pads can reach all LBs of the architecture, and output pads can be reached by all LBs from different paths.

Figure 1 shows TFPGA architecture with 2 levels of hierarchy and Arity 4x4 (the level2 cluster contains k = 4 sub-clusters and each level1 cluster contains k = 4 LBs). A prototype of this architecture, with 4 levels of hierarchy and arity 8x8x8x4 (2048 LBs), was designed using SOC Encounter tools and fabricated with ST 65 nm process. All timing informations of FPGA routing resources are extracted with Encounter tool.

III. CONFIGURATION FLOW

The TFPGA configuration flow begins by a top-down recursive partitioning phase. First, the top level clusters are

constructed, then each cluster is partitioned into sub-clusters, until the lowest level of the architecture is reached. The objective of our partitioning is to minimize the number of connections between partitions. Then, during the placement phase, each cluster is assigned to a position inside its owner cluster. After that, we route signals using the Pathfinder routing algorithm [13] which is an iterative rip-up algorithm.

FPGA routing resources delays, which are extracted with Encounter tool, are inserted in the routing tool. Thus, the router can compute the timing unbalance between two routed dual signals. To improve the differential signals balance, we propose new placement and routing techniques.

A. Adjacent Placement

Keeping the direct and dual gates in the same cluster of TFPGA can be favorable to obtain routing nets as symmetrical as possible. So, to achieve an adjacent placement, we give to our partitionment tool the direct network only. This network is partitionned into clusters as explained before. The architecture used in this phase is not the entire architecture. It is composed of the half of LUTs in each cluster in the lowest level of TFPGA. In our case, each cluster contains 8 LBs. So, only 4 LBS are used. We keep the other half of LUTs free for the dual network. Then, during the placement phase, the direct network is placed using the half of LUTs in each cluster. After that, we place each dual gate adjacent to the direct gate in the same cluster.

B. Resources_Balance_Driven Routing Algorithm

This router is an extension of the PathFinder algorithm. Its objective is to balance the number of routing resources used to route two dual nets. It is based on congestion and resources-balance negotiation. Interconnect resources are presented by a routing graph with nodes corresponding to wires and CLBs pins and edges presenting multiplexers. Consider the connection to sink j of net i. We define the cost of using a routing resource n as:

\[ \text{Cost}(n) = \text{Cong} \cdot f(\text{Diff_res_nb}(i,n,j)) \]  (1)

where 

\[ f(\text{Diff_res_nb}(i,n,j)) = \max(0.1,|\text{Diff_res_nb}(i,n,j)|). \]

Before describing our routing algorithm, we define the following terms:

1) Cong\_cost(n) is the congestion cost of a routing resource n. It takes into account the number of nets sharing this resource, and the history of congestion on it [14].

2) Diff\_res_nb(i,n,j) is the difference between the estimated total resources number that will be used to reach the sink j of net i from the net driver and using the node n and the number of used interconnect resources in the routing of the dual signal. If Diff\_res_nb(i,n,j) = 0, the resource cost (1) should not be equal to 0, because the congestion can be important. We have experimentally determined that setting the minimal f(Diff\_res_nb(i,n,j)) to 0.1 is the best value.

Based on the aforementioned notations, we can illustrate the flow of our routing algorithm. In the first iteration, one of two complementary nets is routed, based on the congestion cost. The routing does not take into consideration the differential resources number, since the dual net has not yet been routed. Then, during the routing of the dual net, the router computes the Diff\_res_nb(i,n,j). It tries to find the path with the same number of routing resources, taking into account the resource congestion. During the next iteration, the original net will be ripped-up and re-routed taking into consideration the last routing results of the dual net (in the previous iteration), and
vice versa. The PathFinder routing algorithm performs many iterations until all resources conflicts are resolved.

C. Timing_Balance_Driven Routing Algorithm

We noted that the timing unbalance between dual nets is caused not only by the differential routing resources number, but also by the difference between wires lengths in the same hierarchical level in FPGA. Relying on this remark, we propose a routing algorithm which goal is to balance the propagation delays of routing resources used by two dual connections and also to balance the number of routing resources used. The new router is an extension to PathFinder router an it is based on congestion-delay negotiation. Consider the connection to sink \( j \) of net \( i \). We define the cost of using a routing resource \( n \) as:

\[
\text{Cost}(n) = (1 - \text{Delay} \_\text{Crit}(i,j)) \cdot \text{Cong} \_\text{cost}(n) + \text{Delay} \_\text{Crit}(i,j) \cdot \text{Res} \_\text{diff} \_\text{delay}(i,n,j) + \text{Diff} \_\text{res} \_\text{nb}(i,n,j)
\]

(2)

\( \text{Cong} \_\text{cost}(n) \) and \( \text{Diff} \_\text{res} \_\text{nb}(i,n,j) \) are explained in III-B. The other terms are defined as the following.

1) \( \text{Res} \_\text{diff} \_\text{delay}(i,n,j) \) is the difference between the delay of a resource \( n \) used to route a connection to sink \( j \) of net \( i \), and the delay of the resource \( n' \) used by the complementary connection. \( n \) and \( n' \) must be of the same type (downward or upward) and situated at the same hierarchical level in the tree-based architecture.

2) \( \text{Delay} \_\text{Crit}(i,j) \) is the delay unbalance criticality between the connection \( j \) of net \( i \) and the dual connection. It can be formulated as:

\[
\text{Delay} \_\text{Crit}(i,j) = \min(0,9, \frac{\text{Connection} \_\text{diff} \_\text{delay}(i,j)}{\text{Max} \_\text{connection} \_\text{diff} \_\text{delay}})
\]

(3)

where:
- \( \text{Max} \_\text{connection} \_\text{diff} \_\text{delay} \) is the maximum \( \text{Connection} \_\text{diff} \_\text{delay}(i,j) \) among all routed connections in the design.
- \( \text{Connection} \_\text{diff} \_\text{delay}(i,j) \) is the timing unbalance between the connection to sink \( j \) of net \( i \) and the complementary connection to sink \( j' \) of dual net \( i' \).

The connection criticality is a fractional number between 0 and 1. High connection criticality means that the real and dual connections have an important delay unbalance.

The first term in equation 2 is the congestion sensitive term, and the second term is the delay sensitive term. The congestion-delay trade-off of each connection is controlled by how critical it is. The router performs many iterations until all routing resources conflicts are resolved. In a routing iteration, once the original net is routed, the router stores informations about the routing: the type of used resources, the delay of each resource and the number of used resources. Then, during the routing of the dual net, the router computes the delay of each used resource \( n \), the \( \text{Res} \_\text{Diff} \_\text{delay}(i,n,j) \), and the \( \text{Diff} \_\text{res} \_\text{nb}(i,n,j) \). It tries to find the path with the minimum differential delay and the minimum differential routing resources number, considering the resource congestion.

It can be seen in equation (2) that a critical connection will be routed by a most balanced path even if it is congested, while a non critical connection pays less attention to balancing.

D. Routing Results

To evaluate the complementary networks balance, we compute, for all dual connections of all dual nets, the absolute difference \( \Delta \text{delay} = |\text{delay}(\text{true}) - \text{delay}(\text{false})| \), where \( \text{delay}(\text{true}) \) and \( \text{delay}(\text{false}) \) are interconnect latencies of true and false signals respectively. We compute also the number of unbalanced connections in terms of used routing resources number. The table I shows the routing results of WDDL PRESENT design with the different P&R strategies. Unconstrained P&R means that we use the partitioning and placement technique explained in III and the original PathFinder routing algorithm. Both resources_balance_driven and timing_balance_driven routing algorithms are applied after an adjacent placement. We can see that new P&R techniques produce interesting results. Indeed, new routers succeeded to route all dual connections with the same number of routing resources (Res_imbalanced connections nb. = 0).

Resources_balance_driven router achieves 52 % of timing improvement. With timing_balance_driven router, we obtain better results. In fact, timing unbalance is reduced by 86 % compared to results obtained with unconstrained P&R.

IV. SECURITY EVALUATION

A. Side-Channel Analysis

Side-channel attacks or physical attacks target the unintentional analog leakage \( Y \) (power consumption, electromagnetic radiation, ...) from a cryptographic circuit. To mount an attack, an attacker predicts an intermediate leakage value \( L(X) \), for a known part of the plaintext (or ciphertext) \( X \) and key hypothesis \( K \). \( L \) is known as the leakage function which is implementation or platform dependent. We target the state register and thus use the Hamming distance leakage model i.e. number of transition in a register. The attack consists in finding a dependence between the acquired side-channel activity (for ex. power traces) and estimations based on the leakage model. We use Correlation power analysis (CPA [15]), which uses Pearson Correlation Coefficient \( (\rho) \) to find dependence.

B. Experimental Setup and Results

The tree-based FPGA circuit is mounted on a daughter card which is interfaced (mounted) with a SASEBO-W evaluation board [16] using the I/O pins. SASEBO-W contains a Spartan-6 FPGA from XILINX which is used to communicate between
the FPGA circuit and outside world (PC). As explained earlier, we program the tree-based FPGA with several PRESENT [12] crypto-processors, each with a different placement and routing configuration. Side-channel traces are acquired when the PRESENT crypto-processors encrypts a given message. These traces are captured using a EM antenna of the HZ–15 kit from Langer, placed over a power decoupling capacitor, and a 54855 Infinium Agilent Oscilloscope at a sampling rate of 5 GSample/s. We captured 10000 traces for the unprotected single-rail PRESENT crypto-processor and 500,000 traces for each configuration of PRESENT WDDL crypto-processors.

We compute the security gain (SG), i.e. the ratio between the number of Measurements To Disclose (MTD) the secret key of WDDL protected modules to corresponding single-rail unprotected module. Table II summarizes the results of a CPA on PRESENT unprotected module. With the acquired traces, we retrieve all the 16 bytes of the secret key using only 1500 traces only. Next we attacked the traces for each of the three WDDL implementations. We do not recover all the 16 bytes of the key from any WDDL implementation. Table III shows the security gain computed for each SBOX of the three PRESENT WDDL implementations. “−” sign means that the particular byte of the secret key is not disclosed at the end of the attack. The security gain varies from $2^8$ to $2^{182}$. We reduce the number of bytes attacked from 8 to 6 by applying more constraints on the P&$R$. Thus we show that the proposed P&$R$ techniques do enhance the security of dual-rail designs. This is also shown by the the timing results in Table I.

Although the timing unbalance has significantly reduced from unconstrained P. & R. to adjacent P. & timing_balance_driven R. (factor of 7), we do not see the same trend in security evaluation results. This indicates that there are other parameters like process variation, early propagation effect etc. between LUTs in the circuit which add unbalance. Since in our current experiments we could not model the effect of process variation and other unknown sources of unbalance, we cannot directly link the timing and security results. Further research would focus on finding and modeling these unknown sources of unbalance.

V. Conclusion

In this paper, we studied the security of WDDL countermeasures on a hierarchical Tree-based FPGA (TFFPGA). WDDL can resist against SCAs and fault attacks. Although the resistance against fault attacks in WDDL comes by construction, to protect against SCAs proper backend design techniques must be deployed. We tested different configurations of placement and routing for the TFFPGA. We proposed an adjacent placement, a resources_balance_driven and a timing_balance_driven routing algorithms. We showed that with proper P&$R$ we can improve the security of WDDL design. Security evaluations were carried out using $500K$ traces. They indicate that the use of proposed P&$R$ techniques is important for SCA resistance of WDDL design.

We suspect that there exist other sources of unbalance in the FPGA like process variation, early propagation effect etc. In future work, we intend to study and resolve these unbalances.

References

[17] W. He et al., Automatic generation of identical routing pairs for FPGA implemented DPL logic. ReConFig 2012.