Secure Logic Synthesis

Kris Tiri[1] and Ingrid Verbauwhede[1,2]

[1]UC Los Angeles, [2]K.U.Leuven
{tiri,ingrid}@ee.ucla.edu

Abstract. This paper describes the synthesis of dynamic differential logic to increase the resistance of FPGAs against Differential Power Analysis. Compared with an existing technique, it saves more than a factor 2 in slice utilization. Experimental results indicate that a secure version of the AES algorithm can now be implemented with a mere doubling of the slice utilization when compared with a normal non-secure single ended implementation.

1 Introduction

Side-channel attacks (SCAs) have been identified as an important open issue related to the general security of cryptographic applications on FPGAs [1]. These attacks find the secret key with information associated with the physical implementation of the device, such as time delay and power consumption. Much effort has already gone into setting up the Differential Power Analysis (DPA) on FPGAs [2].

We have previously presented a logic level design methodology to implement a secure DPA resistant crypto processor on FPGA [3]. In this manuscript, we study the synthesis aspects in order to reduce area consumption and time delay. The next section briefly introduces Wave Dynamic Differential Logic (WDDL), the cornerstone of the logic level design methodology. Section 3 discusses a technique to combine several WDDL gates into 1 slice. This reduces the area consumption and time delay. Section 4 describes the clustering procedure of the synthesis methodology. In section 5, the performance is evaluated. Finally, a conclusion is formulated.

2 Wave Dynamic Differential Logic

To address power attacks, we have introduced a family of secure compound standard cells, referred to as Wave Dynamic Differential Logic [3]. WDDL can be constructed from regular standard cells and is applicable to FPGA. WDDL achieves its resistance by charging in every cycle a constant load capacitance. It is dual rail with precharge logic in which a pre-discharge wave travels over the circuit. In the precharge phase, the inputs to the WDDL gate are set at 0. This puts the output of the gate at 0 and the precharge wave travels over to the next gate.

The set of logic gates is restricted to the WDDL AND- and OR-gates in order to assure that every compound standard cell has exactly 1 output transition per cycle [3]. In addition, it is essential for input independent power consumption that the gate

This work was supported in part by NSF grant CCR-0098361.

always charges ideally the same load capacitance. The capacitances at the differential in- and output signals are alike [4]. There is however a difference in the interconnect capacitance due to routing variations. Placing the 2 LUTs of a compound standard cell adjacent and in the same slice minimizes this effect. Then, the differential signals need to travel the same distance.

The basic building block of a Virtex-II FPGA is known as a slice and consists of two 4-input, 1-output look up tables (LUTs), some multiplexers and registers. A WDDL AND-gate (OR-gate) occupies 1 slice, in which the G-LUT functions as an and-operator (or-operator) on the true inputs, while the F-LUT functions as an or-operator (and-operator) on the false inputs.

3 Slice Compaction

Currently, 2 LUTs are used to build a compound WDDL gate. Each LUT is generates one output of the differential output pair. It is however possible to add more functionality into each LUT. A cluster formed by an arbitrary collection of G-LUTs and the cluster formed by the corresponding F-LUTs behave as a compound WDDL-gate. The 2 clusters (1) are differential; (2) transmit the precharge value; and (3) have a 100% switching factor. A LUT on the Virtex-II platform has 4 inputs and 1 output. In this case, the clusters can have at most 4 inputs and 1 output.

Fig. 1 shows an example. Fig. 1A depicts the single ended logic function to be implemented. The WDDL implementation that results from our original methodology is shown in Fig. 1B. Each gate is replaced by its corresponding WDDL gate. In total, 6 slices are occupied. The logic depth is 3. Fig. 1C shows the implementation after clustering. This implementation occupies only 2 slices and has a logical depth of 2. The clustering algorithm to obtain such compact, side-channel resistant implementations of WDDL based circuits is the topic of this paper.

4 Logic Synthesis

The kernel of DPA-proof logic synthesis is a clustering algorithm. Given a DPA-proof implementation consisting solely out of secure compound WDDL AND- and OR-gates, it partitions the design into groups of LUTs with 4 or less distinct inputs and 1 output. Each group will form together with their corresponding dual group a secure compound gate and will be mapped onto adjacent LUTs within the same slice. A group of LUTs can be divided into many partitions. Various factors, such as graph traversal order and redundancy introduction, influence the compaction. The remainder of this section describes an alternative, yet efficient clustering procedure.

4.1 Clustering through Transformation

Fig. 1C could also have been obtained through a transformation of the synthesized single ended design, shown in Fig. 1D. It is a parallel combination of this design and its dual. To implement an arbitrary logic function however, several inversions may be present. Inversions prohibit a direct transformation.

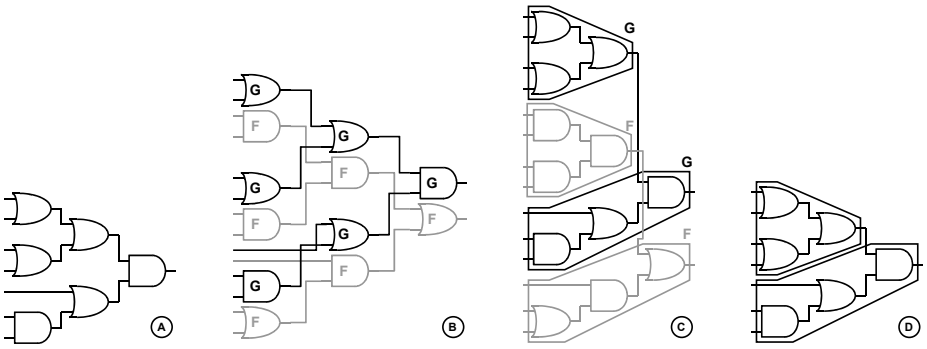

Fig. 1. Original single ended logic function (A); WDDL implementation (B); clustered WDDL implementation (C); and synthesized single ended implementation (D).

This is best seen with an example. Fig. 2A shows a logic function implemented with and2, or2, and inverter gates. The synthesized single ended implementation is shown in Fig. 2B. Note that inside one LUT, there is an inversion. This is not a good partitioning. The precharge 0 at the input of the inverter is propagated as a 1 and consequently at least 1 of the 2 dual LUTs will have a 1 at the output during the precharge phase. Hence, the precharge 0-wave is halted. The WDDL implementation obtained through to the original design methodology is shown in Fig. 2C. Here the inverters have been removed. The outputs of the secure compound gate that precedes the inverter have been exchanged and as a result there is no inverter anymore to halt the precharge wave. This procedure however, interconnects the G- and F-LUTs.

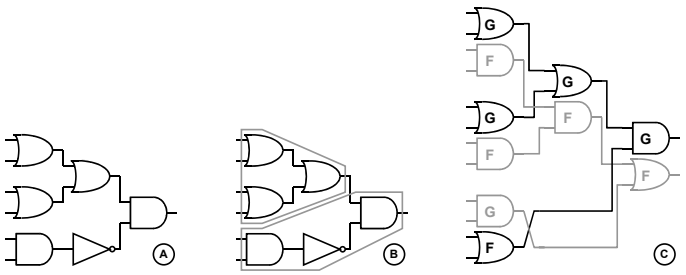

Fig. 2. Inversion mixes G-LUTs and F-LUTs: arbitrary logic function with inversion (A); synthesized single ended design (B); and original WDDL implementation (C).

4.2 Practical Design Flow

The examples of above, lead to a first design flow:

1. The design is synthesized with a limited standard cell library (and2, or2, inverter).
2. The inverters are removed from the result of step 1. The input of each inverter becomes a global output; the output of each inverter a global input.
3. The result of step 2 is synthesized for FPGA implementation.

4. Each LUT of step 3 is implemented in a G-LUT, its dual in the adjacent F-LUT. The in- and outputs created in step 2 are reconnected. The inversions are established by switching the differential connections.

A detailed discussion of this design flow is available [5].

Performing 2 synthesis procedures (in step 1 and 3) is inconvenient and seems redundant. Furthermore, the methodology is only suitable for area optimization. In step 4, the disconnected paths, which have been created in step 2 through stripping of the inverters, are connected. As a result, the delays are summed and may be larger than the critical path of step 3. In the next section, a compressed design flow is presented that ignores steps 2 and 3 and that can minimize the critical path.

4.3 Compressed Design Flow

Since a cluster formed by an arbitrary collection of and2 and or2 gates and its dual will behave as a WDDL gate, the synthesis library can be expanded to include all functions in which 4 or fewer inputs are combined with the and2 and or2 operator. Additionally, since all signals will eventually be differential, the input signals may be inverted and the output signals may be inverted. Or in other words instead of having a secure AND and OR gate, we synthesize directly with the complete selection of secure gates that can be implemented in a slice. We can now skip step 2 and 3 of the practical design flow. The resulting secure digital design flow to implement DPA resistant FPGAs is shown in Fig. 3. The gray colored blocks are the stages of the previous design flow that have been expanded or excluded.

The script transforms the single ended gates in their WDDL counterparts. Each gate declaration is replaced with a primitive module of the FPGA and the dual of this primitive module. Mapping directives are added to implement both in adjacent LUTs.

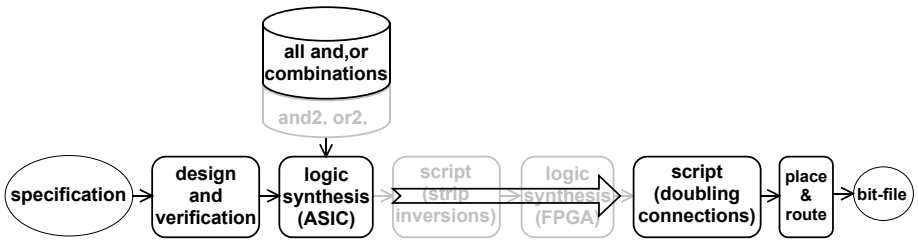

Fig. 3. Secure digital design flow for FPGAs.

5 Experimental Results

We implemented substitution-boxes of Kasumi, DES and AES. For each substitution-box, 4 designs have been implemented: (1) *original WDDL*, the result from the original AND-OR design methodology (Fig. 1B and 2C); (2) *differential*, the result from a regular insecure synthesis of the differential netlist of the original WDDL description; (3) *compacted WDDL*, the result from the compressed design flow of section 4.3 (Fig. 1C); and (4) *single ended*, the result from a synthesis of a normal

insecure single ended design (Fig. 1D and 2B). The differential implementation serves as benchmark because the synthesis tool only has behavioral information and is free to map the functionality onto the LUTs. We have used DesignAnalyzer for the original and the compacted WDDL implementation and the synthesis tool in the XST Verilog design flow for the differential and the single ended implementation. The programming files have been generated for a Virtex2 xc2v1000-6bg575 with the same pin locations for each implementation. Synthesis and Place & Route have been done with the default settings of the tools.

Table 1 presents the slice utilization. On average, there is a factor 2.25 reduction between the original and the compacted WDDL implementations. There is also an important difference, up to 37%, between the benchmark implementation and the compacted WDDL implementation. The compacted WDDL designs of DES and Kasumi are on average a factor 4.42 larger than the single ended designs. The secure AES design however, only requires 1.95 times the slices of the single ended design.

Table 1. Slice utilization.

	DES								Kasumi		AES
	S1	S2	S3	S4	S5	S6	S7	S8	S7	S9	Sbox
original WDDL	138	139	137	143	136	142	137	128	249	303	797
differential	73	81	87	78	86	85	76	73	142	157	357
compacted WDDL	67	65	64	57	70	68	59	64	110	123	340
single ended	14	15	14	8	13	15	15	13	30	32	174

6 Conclusions

We have presented a design methodology to synthesize secure DPA resistant logic. Compared with the original WDDL, slice compaction offers more than a factor 2 reduction in slice utilization. The methodology seems perfect for the AES algorithm. Compared with a single ended design, the overhead in slice utilization is restricted to a factor 2. The experiments have also shown that the synthesis methodology achieves smaller utilization factor than a conventional FPGA synthesis tool.

7 References

1. T. Wollinger and C. Paar, "How Secure Are FPGAs in Cryptographic Applications?" in Proc. of FPL 2003, LNCS 2778, pp. 91–100, Sept. 2003.
2. B. Örs, E. Oswald and B. Preneel, "Power-Analysis Attacks on an FPGA – First Experimental Results," in Proc. of CHES 2003, LNCS 2779, pp. 35–50, Sept. 2003.
3. K. Tiri and I. Verbauwhede, "A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation" in Proc. of DATE 2004, pp. 246–251, Feb. 2004.
4. L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic Power Consumption in Virtex-II FPGA Family," in Proc. of FPGA 2002, pp. 157–164, Feb. 2002.
5. K. Tiri and I. Verbauwhede, "Synthesis of Secure FPGA Implementations," UCLA Internal report, available as report 2004/068 from the IACR Cryptology ePrint Archive, Feb. 2004.