2023 Design, Automation & Test in Europe Conference (DATE 2023)

Quo Vadis Signal? Automated Directionality Extraction for Post-Programming Verification of a Transistor-Level Programmable Fabric

Apurva Jain, Thomas Broadfoot, Yiorgos Makris, Carl Sechen
Department of Electrical and Computer Engineering, The University of Texas at Dallas, Richardson, Texas, USA
{apurva.jain, thomas.broadfoot, yiorgos.makris, carl.sechen}@utdallas.edu

***Abstract*—We discuss the challenges related with developing a post-programming verification solution for a TRAnsistor-level Programmable fabric (TRAP). Toward achieving high density, the TRAP architecture employs bidirectionally-operated pass transistors in the implementation of its logic and interconnect network. While it is possible to model such transistors through appropriate primitives of hardware description languages (HDL) to enable simulation-based validation, Logic Equivalence Checking (LEC) methods and tools do not support such primitives. As a result, formally verifying the functionality programmed by a given bit-stream on TRAP is not innately possible. To address this limitation, we introduce a method for automatically determining the signal flow direction through bidirectional pass transistors for a given bit-stream and subsequently converting the HDL describing the programmed fabric to consist only of unidirectional transistors. Thereby, commercial EDA tools can be used to check logic equivalence between the transistor-level HDL describing the programmed fabric and the post-synthesis gate-level netlist.**

I. Introduction

A TRAnsistor-Level Programmable fabric (TRAP) was recently developed [1] and used in the context of integrated circuit redaction [2] to protect hardware intellectual property from an untrusted foundry. Unlike traditional FPGAs, which implement logic functions using Look-Up Tables (LUTs), TRAP pushes granularity of post-fabrication programmability down to the transistor level. TRAP claims significant reduction in area, performance and power consumption overhead over conventional LUT-based solutions, while at the same time presenting much harder obstacles for brute-force or intelligent search-based reverse engineering attacks to overcome [1].

Widespread adoption and utilization of this promising technology, however, requires support by commercial CAD tools for all design-related tasks, including formal verification. This capability is particularly important for unconventional custom fabrics, such as TRAP, to instill confidence regarding circuit implementation correctness. While industry-standard Logic Equivalence Checking (LEC) tools are capable of understanding and handling transistor-level constructs, this capability is, in itself, insufficient for performing formal verification of an entire design. Indeed, describing a design at the switch level results in a tremendous number of nodes that need to be compared and, therefore, prohibitive time. Instead, LEC tools employ a transistor abstraction method that converts switch-level designs into their gate-level equivalent. Such conversion assumes that the signal flow direction in MOS transistors is resolved and known to the LEC tool. However, toward implementing a dense fabric, TRAP employs various bidirectional pass transistors for which it is challenging to determine the actual signal direction, even if the programming bitstream is known.

This paper introduces a method which leverages the information produced by the placer, router and bitstream generator when mapping a design to a TRAP fabric, to generate a signal flow graph and deduce direction across bidirectional pass transistors. This information enables derivation of an HDL model for a *programmed* TRAP fabric, which can be used by LEC tools to mathematically assert that the Boolean expression implemented by a transistor-level netlist on TRAP is the same as the one implemented by the post-synthesis gate-level netlist.

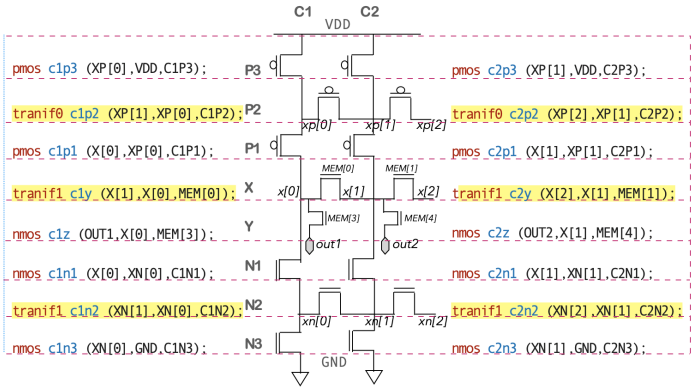

Fig. 1. HDL for an Unprogrammed Transistor Array (TA)

II. TRAP Architecture and HDL Modeling

This section highlights the architecture and HDL of the two core programmable components: (i) a transistor array (TA) and (ii) an interconnect network. These components are hierarchically arranged into a *Unit*, which is then replicated in a 2-dimensional array to produce a continuous fabric. An array of memory cells (*i.e.*, custom SRAM) is inter-weaved with it to store the programming bits.

A. Transistor Array

Fig.1 shows transistor-level HDL code written for two columns of the TA. In HDL, modeling with built-in *pmos/nmos* primitives requires that we define the input and output for a transistor, while a *tranif0/tranif1* construct, which is a bidirectional PMOS/NMOS respectively, has no such requirement. Hence, switches with known signal direction are modeled as *pmos/nmos*, while switches for which signal direction is not established are modeled as *tranif0/tranif1*.

Depending on the logic being implemented on a set of columns in the TA, the so-called horizontal transistors can have a signal flow in either direction (highlighted in yellow in Fig. 1). Transistors P3 and P1 are defined as unidirectional PMOS devices because the signal flow direction through them is known (*i.e.*, one node is clearly at a higher potential than the other). However, the specific signal direction for transistor P2 will depend on how the transistor gates are programmed in this column; therefore, P2 must be defined as *tranif0*. A similar logic is used to define transistors in the pull-down network.

B. Interconnect Network

Fig.2 shows the transistor-level HDL for one column of the interconnect network of TRAP. Switches, shown as dots, are NMOS pass transistors that connect orthogonal metals (*i.e.*, L3 and L4), controlled by the corresponding SRAM bit (MEM[n]) at multiple locations. The signal flow direction in the interconnect network depends on the application mapped on

978-3-9819263-7-8/DATE23/© 2023 EDAA

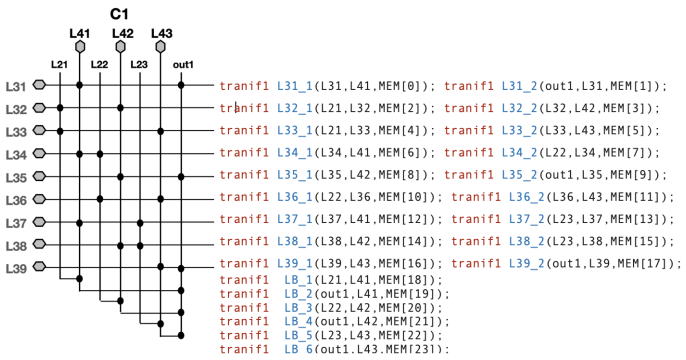

Fig. 2. HDL for an Unprogrammed Interconnect network of TRAP

it. Hence, the bidirectional NMOS construct *tranif1* correctly represent each switch. When switches are turned ON, the two metal layers connected by the switch share the same signal.

III. Automated Directionality Extraction

The HDL model of the *unprogrammed*, TRAP fabric as derived in the previous section, can be used in conjunction with a programming bitstream to perform simulation-based validation. However, the use of the HDL construct *tranif1/tranif0*, which was necessary for modeling the bidirectional pass transistors in the TA and interconnect network of TRAP, is not supported by LEC tools. Indeed, in order to employ such tools, the signal flow direction through these bidirectional pass transistors must first be resolved, and the *tranif0/tranif1* constructs must be converted to their equivalent *pmos* or *nmos* constructs. Such information is not available until after the device is programmed and, even then, it is not trivial to extract.

A. Resolving Signal Direction in the Transistor Array

Depending on how such transistors are programmed (and therefore also connected together), not only is the logic being programmed on the fabric different, but also the direction of how signals flow across each pass transistor is different. Indeed, a key issue encountered in transistor-level programmable fabrics is that, in addition to the transistors implementing the functionality, additional permanently turned ON or OFF transistors are used in order to stitch together the logic functionality and to delineate/isolate logic gates from the rest of the array, respectively. To handle the latter, we developed a method, which not only makes use of the programming bitstream but also leverages two important observations regarding the TRAP architecture. First, for signal integrity purposes, the maximum number of NMOS or PMOS transistors that can be stitched in series is small (in our case, three). Second, the location of the output nodes in each column of the TA is pre-determined. We categorizes horizontal bidirectional pass transistors in the TA based on the signal that drives their gate and, accordingly, performs the following actions:

- **When the gate is OFF:** When bidirectional pass transistors are programmed to be are permanently off, we model the transistor as an open circuit which is equivalent of remove the corresponding *tranif0/tranif1* construct in the HDL code.
- **When the gate is ON:** When bidirectional pass transistors are programmed to be permanently on, the nodes connecting the source and drain are shorted. Therefore, we replace the corresponding *tranif0/tranif1* construct in the HDL implementation with the primitive *tran*, which connects the two nodes mentioned in its arguments, thus effectively implementing a shorted wire.
- **When the gate input is a signal:** In this case, we convert *tranif0/tranif1* to a directional *pmos/nmos* construct by resolving the two remaining ports as source and drain, respectively, based on their proximity to power/output ports to which they are connected through conducting transistors. For example, a node connected to a power port through an ON transistor is classified as source.

B. Resolving Signal Direction in the Interconnect Network

The regular interconnect architecture in TRAP has orthogonal metal lines connected via bidirectional NMOS pass transistors. To resolve their directionality and develop the HDL that describes the programmed interconnect, we rely on the path generated by the router for each net, a list of active switches and a graph of the unprogrammed TRAP interconnect architecture (II-B). To generate the HDL of the interconnect of the programmed fabric, we select one routing path at a time and traverses it starting from origin of the net and until it encounters the first active switch. At that point, it determines that the origin of the net is the driving net for the next segment. It then continues traversing the routing path and every time it encounters an active switch, the previous net becomes the driving net, eventually determining independently the signal flow path through each active pass transistor in each branch of the path. Once the source and drain terminals of each active switch are determined by this traversal. This method generates the HDL for the programmed interconnect by replacing each encountered *tranif1* construct of the unprogrammed fabric with an appropriately directed *nmos*. Finally, we remove all inactive transistors from the interconnect HDL.

IV. Formal Verification of Programmed TRAP

Once the signal directionality through bidirectional pass transistors in the TA and the interconnect network of TRAP is resolved and the *tranif1/tranif0* statements of the unprogrammed HDL description of TRAP are replaced with appropriate constructs, each primary input and output is assigned to the corresponding track on fabric. The resulting HDL describing the programmed TRAP fabric can be readily used by commercial LEC tools. This HDL is verified against the synthesized netlist to ensure equivalence between the desired gate-level implementation and the actual transistor-level implementation on TRAP.

V. Conclusion

Towards maximizing flexibility in implementing logic functions, transistor-level programmable fabrics, such as TRAP, often employ bidirectional pass transistors. While such transistors can be appropriately modeled in an HDL and simulated for validating the functionality of a programmed fabric, the corresponding HDL constructs cannot be handled by LEC tools due to their unknown signal directionality. The methodology presented in this work resolves this limitation by automatically extracting signal directionality of such bidirectional pass transistors in the TRAP architecture and by generating an HDL for the programmed fabric which is accepted by LEC, thereby enabling formal verification between a synthesized gate-level netlist and its actual transistor-level implementation on a fabric.

References

[1] J. Tian *et al.*, "A Field Programmable Transistor Array Featuring Single-Cycle Partial/Full Dynamic Reconfiguration," in *DATE*, 2017.
[2] M. Shihab *et al.*, "Design Obfuscation through Selective Post-Fabrication Transistor-Level Programming," in *DATE*, 2019, pp. 528–533.