Taking Test Programs Connectivity to the Bit-Level

Andrea Calabrese
Dip. automatica ed informatica, Politecnico di Torino
Turin, Italy
andrea.calabrese@polito.it

***Abstract*—As System-on-Chips become more and more complex, a high fault coverage is becoming more difficult to achieve. As fault coverage is the de facto standard metric for testing, we aim for a faster heuristic for test evaluations. In particular, last year a faster heuristic known as *connectivity* was defined in ITC 2022 [1]. We want to improve over this heuristic, trying to improve the precision of the connectivity metric. In particular, we introduce a bit-level connectivity, using the general connectivity as an optimistic value, however taking into account the toggle coverage at a register-level. Our case study is a chip developed by STMicroelectronics of the SP58 family.**

***Index Terms*—Burn-in test, functional test, heuristics, EDA, Automotive.**

I. Introduction

As System-on-Chips (SoCs) increase in scale on every generation, testing is becoming an increasingly difficult task, both in terms of test complexity and in terms of time spent; moreover, many SoCs include now multi-core capabilities and even the interactions between different cores need to be tested [2]. Thus, the probability of physical defects keeps increasing [3]. Standards such as ISO-26262 [4] provide a well-defined metric for safety applications in the automotive field, but means to achieve the desired safety metrics are becoming more and more expensive in terms of time.

In this paper, we will improve over the coverage heuristic [1] to provide more details about the data passed between instructions of a test program. In Section II we will shortly talk about the original coverage metric. In Section III we will propose the methodology to improve over the original coverage metric. In Section IV we will show our experiments on our case study. In the end, we will draw some conclusions.

II. Background

Burn-In (BI) is a well-established methodology for finding latent defects in SoCs [5]. In particular, during the BI analysis, various tests and programs run on the Device under Test (DuT), interleaved with signature checks on intermediate results. Functional testing is a methodology suited for both online and manufacturing testing [6]–[8]. It mainly consists in executing programs, both hand-written or generated by an automatic tool such as microGP [9], during the testing phase.

Fault coverage is the de facto standard in the testing field to evaluate the goodness of a testing program or pattern. However, this process tends to be time-consuming; however, in [1], a connectivity metric emerges as a proposal for a fast evaluation of a test program. Connectivity does not aim to substitute the fault coverage, but it is meant to be a helping metric for the programmer (be it a person or a tool) to improve upon such a program.

In general, given two instructions I_x, I_y and a register R_n, we say those instructions are connected on R_n when, if I_x writes on register R_n and I_y reads from such register, no instruction I_z between I_x and I_y writes on R_n. The connectivity analysis assesses the connections between all of the instructions on the test program on each register.

This way, connectivity aims to detect the usefulness of each instruction of the test program, giving the programmer a choice between improving the program in terms of time consumption (eliminating useless instructions) or in terms of coverage (connecting those instructions to the next ones).

III. Proposed approach

As the connectivity takes into account the flow of the program, it is arguable that the data for the testing is also important. In this paper, we want to propose an enhancement to the connectivity metric, taking into account the propagation of the bits of each register. In particular, we want to produce a report of what bits changed for each register, both involved or not, in the program execution.

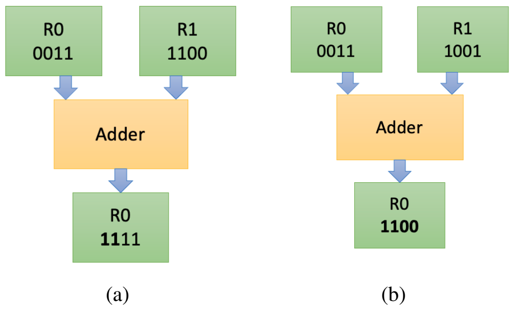

Fig. 1. An adder acting on the same registers carrying different data.

As an example, in Figure 1 we suppose 2 4-bit registers R0 and R1. We suppose that we can control R1 without changing the desired flow of the test program; our objective is to toggle all the bits contained in R0 to improve the toggle activity. In Figure 1(a), we chose a sub-optimal value for R1, as in R0 only 2 bits toggle after the operation. However, in Figure 1(b) we chose R1 in such a way that all the values in R0 toggle in the end.

Thanks to (in alphabetical order): Francesco Angione, Paolo Bernardi, Lorenzo Cardone, Stefano Quer, and the STMicroelectronics' team.

TABLE I

IMPROVEMENTS ON BIT-LEVEL CONNECTIVITY AND THE CONSEQUENCES ON THE FAULT COVERAGE

	Original			**Improved**		
Test program	**Coverage [%]**	**Connectivity [%]**	**Bit-level connectivity [%]**	**Coverage [%]**	**Connectivity [%]**	**Bit-level connectivity [%]**
Adder	92.56	91.38	23.82	93.67	91.38	81.21
Multiplier	92.30	68.96	45.25	93.38	68.96	78.43
Floating Point	90.78	96.4	36.96	91.71	96.4	40.01
Shifter	86.20	92.38	48.14	89.48	95.17	48.29
Count-zeros	86.81	94.49	13.72	87.72	94.49	14.51
Load-Store	91.85	53.28	19.06	93.76	53.28	22.06
Branch Target Buffer	71.16	68.45	30.99	72.14	68.45	35.99
Bit-wise logical	95.00	100.00	30.14	95.95	100.00	31.84

We can extend this idea to every N-bit register and to every operation performed by the test program. Starting from the connectivity data, we perform an additional pass: since only the connected instructions produce fault coverage, we are interested in the bit-level connectivity of those instructions. We compute the cumulative bit-level connectivity percentage on the instruction-to-instruction level per register using Equation 1:

$$C_{i2i}(R_n(t_0), R_n(t_1)) = \frac{Count\_ones(R_n(t_0) \bigoplus R_n(t_1))}{Size\_of(R_n)} \tag{1}$$

Where C_{i2i} is the bit-level connectivity on instruction-to-instruction level, the $\bigoplus$ operation is a xor, and $R_n(t_x)$ is the status of the register n at the time t_x.

However, using a single instruction for a full test on an operation on a register may not be feasible, both because it may affect the program flow and might not take into account different side effects (eg.: overwriting the instruction pointer). In this case, to evaluate the whole program, we can perform a similar operation using the entire original connectivity report, losing granularity over the information while, however, assessing the test at an end-to-end level. To assess this end-to-end bit-level connectivity, we use the Equation 2:

$$C_p = C * \sum_{n=0^N} \frac{\sum_{t=1^T} min(1, C_{i2i}(R_n(t-1), R_n(t)))}{N * Size\_of(R_n)} \tag{2}$$

Where C_p is our desired metric, C is the original connectivity and the rest is the sum of the bit-level single instruction connectivities applied to all registers including all the program, capped at 1. We designed this bit-level connectivity to cap at C for computing the distance between an optimal program in terms of data and the optimal program with respect to the original connectivity.

IV. EXPERIMENTAL RESULTS

We performed our experiments on the SPC58 automotive autocontroller provided by STMicroelectronics.

Table I shows improvements on the fault coverage obtained by increasing the bit-level connectivity on some of our test programs; the only exception is the shifter program, which received improvements both in the original connectivity and in the bit-level connectivity. By looking at the improved coverage column, we can see that even slightly changing the data may have a small improvement on the overall bit-level connectivity. Experimentally, this analysis comes at a slightly higher cost with respect to the original connectivity analysis, in the order of 0.1 seconds. However, the added bit-level analysis provides more in-depth results than the original connectivity, providing a suggestion on what instruction can be improved to reach the full potential of the connectivity metric, therefore increasing the fault coverage.

V. CONCLUSIONS

We propose an improvement on the connectivity metric, developed for fast feedback to the test programmer, providing bit-level precision for each instruction. This bit-level connectivity improves the feedback for the test programmer, showing how many bits were tested for each register on each instruction acting on it. This way, the programmer can focus on the data quality with knowledge of the data flow provided by the original connectivity.

REFERENCES

[1] F. Angione, P. Bernardi, A. Calabrese, L. Cardone, A. Niccoletti, D. Piumatti, S. Quer, D. Appello, V. Tancorre, and R. Ugioli, "An innovative strategy to quickly grade functional test programs," in *2022 IEEE International Test Conference (ITC)*, 2022, pp. 355–364.

[2] F. Angione, P. Bernardi, G. Filipponi, C. Tempesta, M. S. Reorda, D. Appello, V. Tancorre, and R. Ugioli, "Online scheduling of concurrent memory bists execution at real-time operating-system level," in *2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2022, pp. 1–6.

[3] A. Vassighi et al., "Cmos ic technology scaling and its impact on burn-in," *IEEE Transactions on Device and Materials Reliability*, 2004.

[4] "Iso 26262-[1-10], road vehicles – functional safety," 2011.

[5] C. He et al., "Wafer level stress: Enabling zero defect quality for automotive microcontrollers without package burn-in," in *IEEE International Test Conference (ITC)*, 2020.

[6] S. M. Thatte *et al.*, "Test generation for microprocessors," *IEEE Transactions on Computers*, vol. C-29, no. 6, pp. 429–441, 1980.

[7] D. Brahme *et al.*, "Functional testing of microprocessors," *IEEE Transactions on Computers*, vol. C-33, no. 6, pp. 475–485, 1984.

[8] D. Appello et al., "System-level test: State of the art and challenges," in *IEEE IOLTS*, 2021.

[9] G. Squillero, "MicroGP—An Evolutionary Assembly Program Generator," *Genetic Programming and Evolvable Machines)*, 2005.

COVER LETTER for PhD FORUM

First name	Andrea
Last name	Calabrese
Affiliation	Politecnico di Torino
E-mail	andrea.calabrese@polito.it
Phone number	3393944747
Proposed PhD thesis title	New programming paradigms for optimization
Start PhD date	November 2020
Expected End PhD date	February 2024
Type of PhD topic[1]	Applied research
Collaborators[2]	Politecnico di Torino
PhD motivation[3]	I was always passioned about computer. First, I started playing videogames, then I started being interested in the mechanisms behind them. Later, I became interested in parallel algorithms and GPU, on which I worked during the thesis. It felt natural, then, to prosecute the work of my master thesis to the Ph.D., looking for opportunities to both provide an improvement on the current algorithms and provide a concrete usefulness to my research activity.
Scientific results[4]	D. Appello *et al*., "Accelerated Analysis of Simulation Dumps through Parallelization on Multicore Architectures," *2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, Vienna, Austria, 2021, pp. 69-74, doi: 10.1109/DDECS52668.2021.9417048. Quer, S.; Calabrese, A. Graph Reachability on Parallel Many-Core Architectures. *Computation* **2020**, *8*, 103. https://doi.org/10.3390/computation8040103 F. Angione *et al*., "An innovative Strategy to Quickly Grade Functional Test Programs," *2022 IEEE International Test Conference (ITC)*, Anaheim, CA, USA, 2022, pp. 355-364, doi: 10.1109/ITC50671.2022.00044. D. Appello *et al*., "Parallel Multithread Analysis of Extremely Large Simulation Traces," in *IEEE Access*, vol. 10, pp. 56440-56457, 2022, doi: 10.1109/ACCESS.2022.3177613. Calabrese, Andrea, Stefano Quer, and Giovanni Squillero. "Smart Techniques for Flying-probe Testing." *ICSOFT*. 2021.

[1] e.g., Applied research/ R&D, fundamental research, etc.
[2] e.g., university, industry, research centre, etc.
[3] explain why you have decided to pursue a PhD and why in this topic.
[4] provide a list of scientific outputs and innovations (accepted, submitted, expected) including: Patents, Books/ book chapters, Journal papers, Conference papers, Workshop papers etc.