Automatic Design of Reliable Systems Based on the Multiple-choice Knapsack Problem

Jakub Lojda, Jakub Podivinsky, Ondrej Cekan, Richard Panek, Martin Krcma, Zdenek Kotasek
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Bozetechova 2, 612 66 Brno, Czech Republic
Email: {ilojda, ipodivinsky, icekan, ipanek, ikrcma, kotasek}@fit.vutbr.cz

***Abstract*—This paper evaluates the practical usage of the Multiple-choice Knapsack Problem (MCKP) solver to automatically select the proper fault mitigation method for each component to maximize the overall fault tolerance of the whole system. The usage of the MCKP is placed into the context with our fault tolerance automation toolkit, the goal of which is to completely automate the process of fault-tolerant system design on a very general level. To achieve our goal, we present our research on Field Programmable Gate Arrays (FPGAs) for which we have developed the specific components in order to support their fault-tolerant design automation. In our particular case study, the MCKP method on the partitioned system was able to find the solution with 18% less critical bits compared to our previous approach, while even lowering the circuit size. The results indicate that by splitting the system into smaller components and applying the MCKP method, considerably better results in terms of critical bits representation can be achieved.**

***Keywords*—Fault-Tolerant System Design, Electronic Design Automation, Multiple-choice Knapsack Problem, Fault Tolerance Property Estimation, Verification, High-Level Synthesis.**

I. Introduction

As electronic systems penetrate into areas with increased reliability demand, the pressure on designers to make such systems reliable arises. Generally, two main approaches to reliable system design exist: 1) *Fault Avoidance* (FA) [1], which is based on the better selection of proper and more reliable components and does not change the structure or interconnections of the system; and 2) *Fault Tolerance* (FT) [2], which accepts unreliable components as a fact and tries to solve the problem of higher reliability with modification of the system structure. Our research is based on the FT approach. Also, growing complexity of electronic systems led to strategies and design flows that maintain the *Time To Market* (TTM) on a reasonable level. *High-level Synthesis* (HLS) is a good example of such a design flow. Generally, HLS allows a designer to utilize the description written in one of the higher-level programming languages and transform it to its *Register Transfer Level* (RTL) implementation in VHDL or Verilog. In our research we focus on FT design automation and also on its combination with HLS, because our concept of FT design automation is general, and thus is able to cover different design flows.

So far, our research has been targeting SRAM-based FPGAs. SRAM-based FPGAs store their configuration bitstream in an SRAM memory, and thus are prone to the so-called *Single Event Upset* (SEU) bit-flips. A benefit of using FPGAs is also their good usability in the process of FT design testing, because the concepts can be easily tested on a real HW with the usage of the so-called *fault injection*. During the test of an FT circuit, the approach of fault injection is usually combined with the so-called *functional verification* in order to detect the failure of the tested unit. The bits that cause a discrepancy on the output pins during the verification are called *critical bits*, sometimes in the literature also referred to as *sensitive bits*. The percentage of critical bits in an FPGA design is usually understood as a quantified measurement of FT of the design.

The usual step in the process of FT design is to assign a suitable combination of FT techniques to harden individual blocks against faults that would lead to higher FT of the whole circuit. Solutions to these reliability allocation problems can also be found in literature. The *Improved Surrogate Constraint* (ISC) method was applied to the system reliability allocation problems with a mix of components in paper [3]. The authors of [4] proposed the penalty guided artificial bee colony algorithm. The paper [5] presents the use of the variable neighborhood search meta-heuristic method. The use of dynamic self-adaptive multi-objective particle swarm optimization method is proposed in [6]. The usage of the genetic algorithm was examined in [7] and [8], where the use of *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) was presented. The experiments show that the NSGA-II can find a number of promising solutions of the reliability allocation problem. Most of the presented work is a separate solution to this problem without a broader concept. In our research, we bring the integration into the complex tool which targets the automation of FT design process. As the fundamental algorithm for our experiments, we decided to transform the problem of redundancy allocation to the MCKP [9].

This paper is organized as follows: Section II introduces the concept of our FT design automation toolkit. The use of MCKP for FT strategy selection is proposed in Section III. The case study and experimental results are presented and discussed in Section IV. Section V concludes the paper and presents plans for our future research.

II. Fault-Tolerant Design Automation

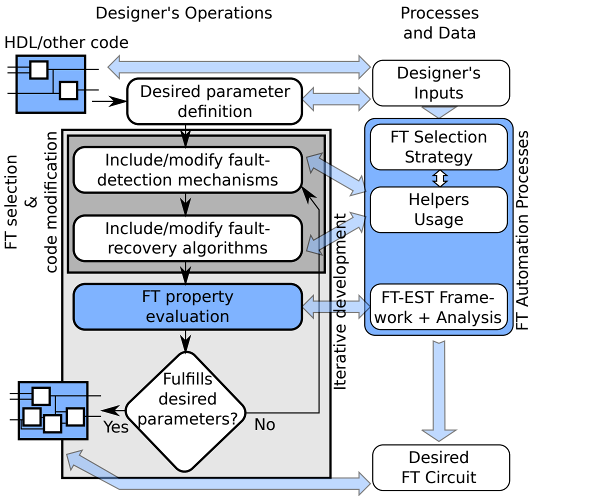
In our research, by FT design automation, we mean the transformation of the so-called *unhardened system* description to its FT version. Our aim is to research FT design methods that allow automatic FT selection and insertion while keeping the methods as much general as possible, with the ability to specialize on particular design flow through addition of special modules. The structure of our platform comes from the ordinary process of FT system creation, that is: 1) specification of the required parameters, and 2) iterative modification and evaluation until the specific requirements are met.

Our FT design automation platform consists of various components and each component targets particular phase or task during the transformation of the system. The first is the component implementing the FT strategy selection, which is the main topic of this paper. The FT strategy selection decides which type of FT method to use for which part of the system. The so-called *helpers* are used during this process. The helpers include libraries or possibly modification

978-1-7281-9938-2/20/$31.00 ©2020 IEEE

scripts, that are able to incorporate redundancy into a particular component of the system. So far, we have been developing specific helpers for the usage with HLS, which we call the *Redundant Data Types* (RDTs) [10]. The RDTs act as new data types in the algorithm description and *decorate* the resulting system to decrease its critical bit representation. After each iteration, the draft of the system is evaluated. The evaluation is performed on the real FPGA HW. We use the *Fault Tolerance ESTimation* (FT-EST) framework [11] to automatically build test-benches. The data obtained through the FT-EST framework can be further examined by the FPGA bitstream-specific analysis [12] to obtain numeric quantification of FT indicators.

The general overview of the traditional approach with a designer making manual operations and the connections to our automatic flow utilizing processes, can be seen in Figure 1.



Designer's Operations
Processes and Data
HDL/other code
Designer's Inputs
Desired parameter definition
FT selection & code modification
Include/modify fault-detection mechanisms
Include/modify fault-recovery algorithms
FT Selection Strategy
Helpers Usage
FT Automation Processes
Iterative development
FT property evaluation
FT-EST Frame-work + Analysis
Fulfills desired parameters?
Yes
No
Desired FT Circuit

Figure 1: Design of FT system from an unhardened system with the designer's operations mapped to the processes of our automation platform.

III. Fault Tolerance Strategy Selection

For our testing, we decided to map the problem of redundancy selection to the MCKP. The MCKP is a special type of the Knapsack Problem (KP). KP and its variations belong to the so-called combinatorial optimization problems [13]. The KP as an optimization is an NP-hard problem. Let's suppose we have a *knapsack* of a particular load *capacity* and a set of objects. Each object has attached the values of the so-called *profit* and *weight*. Then, the KP objective is to solve the problem of the best selection of the objects to achieve the best value (i.e. profit) in the knapsack, while keeping the load below its maximum capacity [14]. The MCKP extends the original KP by distinguishing the objects into classes. In addition, in MCKP, exactly one object from each class must be selected. The general objective is equivalent – to maximize the profit while keeping the load below the given capacity [9]. The MCKP can be mapped to the problem of redundancy selection. However, it is important to note that for our purposes, we slightly modify the MCKP problem to prefer the units with the *smallest handicap*, instead of the highest profit. The handicap is actually the number of critical bits, in our terms and the capacity is the chip area available (e.g. bits of the bitstream). We decided to use critical bits as the metrics, as their representation determines the SEU resistance of a design. Other metrics such as time required to compute a result may be also used for a component, in such case, however, components in the system would have to use a communication protocol because of timing differences. Also the final throughput of the complete system would have to by analyzed separately. Graphical illustration of the usage of MCKP for redundancy selection can be seen in Figure 2.

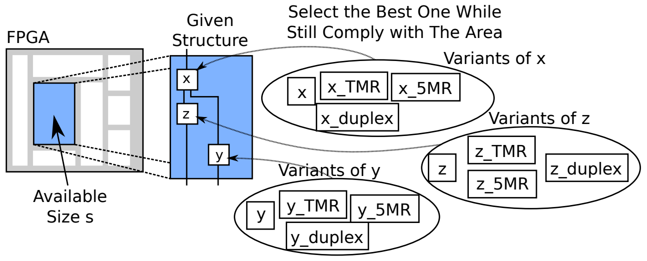


Figure 2: The graphical representation of FT system and the selection strategy based on MCKP.

In our research we chose to evaluate the weights and profits of each component, and then solve the MCKP fully in SW. After that, create the composed unit according to the results obtained from the MCKP solver. This is the reason why our FT strategy has actually just one iteration on the level of the FT automation toolkit because the state-space exploration is performed in the MCKP fully in SW. For this method to work, it is necessary to evaluate each variant of each component to obtain the input parameters for the MCKP solver.

IV. The Case Study and Experimental Results

In our case study, we designed an electronic system that is suitable for our demonstration purposes. The system literally computes the number of 1 bits (i.e. high bits) in the sum of three numbers, of which one is a static constant. The system also computes a CRC-8 checksum of data obtained after the first and second additions. The system can be partitioned into four components: 1) addition; 2) addition of a constant; 3) number of high bits computation; and 4) CRC-8 checksum computation. The block diagram can be seen in Figure 3.

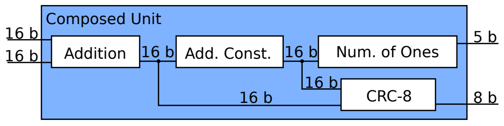


Figure 3: The schematic of the system with its components.

A. Components of Benchmark System and Their Implementation Properties

The component *addition* sums two 16-bit unsigned numbers and provides the result also on 16 bits. The component *Add. Constant* which will be referred to as *addconst* further in the text, adds a constant number to its one 16-bit input. The component *crc8* computes the *Cyclic Redundancy Check* (CRC) out of the 32 bits wide vector. The output is 8 bits wide. The last component, *Number of Ones*, provides an unsigned 5-bit number representing the quantity of high bits in its 16 bits wide input vector. This component will be referred to as *numones*.

For the implementation of the components, we used a traditional HLS design flow utilizing the C++ language. Each component was implemented in four variations incorporating different amount of redundancy in their data-path level using the RDT approach: 1) *simple* (no redundancy); 2) *triple* (data-path triplicated); 3) *quadruple*, (four data-paths); and 4) *quintuple* (five data-paths). Component descriptions were synthesized using the Mentor Graphics Catapult C HLS tool [15].

For each of these circuits, the FT-EST test bench was created. Test benches were synthesized using the Xilinx Integrated Synthesis Environment (ISE) 14.7 [16]. After that, each bit of utilized LUT contents was exhaustively tested on the ML506 evaluation board using Virtex 5 technology. The overview of the synthesized components can be seen in Table I. As can be seen, each RDT (i.e. each FT insertion method) has different efficiency on a component. This property actually suggests that each unique component suits different FT method that achieves the best parameters.

TABLE I: Component Variations and Their Properties

Component Name	Number of Inputs [*b*]	Num. of Outputs [*b*]	Used LUT bits [*b*]	Critical bits [*b*]/ KP Handicap [-]	Critical bits Repr. [%]
addition_simple	16 b; 16 b	16 b	4288	162	3.78
addition_triple	3x 16 b; 3x 16 b	3x 16 b	8320	163	1.96
addition_quadruple	4x 16 b; 4x 16 b	4x 16 b	10304	195	1.89
addition_quintuple	5x 16 b; 5x 16 b	5x 16 b	14528	232	1.6
addconst_simple	16 b	16 b	4224	104	2.46
addconst_triple	3x 16b	3x 16 b	8096	130	1.61
addconst_quadruple	4x 16b	4x 16 b	9952	171	1.72
addconst_quintuple	5x 16b	5x 16 b	14144	198	1.4
crc8_simple	32 b	8 b	4800	977	20.35
crc8_triple	3x 32 b	3x 8 b	6592	819	12.42
crc8_quadruple	4x 32 b	4x 8 b	6976	712	10.21
crc8_quintuple	5x 32 b	5x 8 b	7360	971	13.19
numones_simple	16 b	5 b	4096	380	9.28
numones_triple	3x 16 b	3x 5 b	4800	122	2.54
numones_quadruple	4x 16 b	4x 5 b	5312	125	2.35
numones_quintuple	5x 16 b	5x 5 b	5184	120	2.31

To connect components of various redundancy levels, VHDL bit-based voter was utilized. If there is a requirement to ensure that no component is shared on the system, *pblocks* in the Xilinx PlanAhead software [17] can be used.

B. MCKP Reaction to Area Available on the FPGA

In the first phase of our experiments, the behavior of the MCKP for various chip area settings was tested. We selected the number of LUT bits to represent the chip area occupied. Empirically, the interval from 17500 to 21100 with the step of 100 bits was selected. For each chip-area threshold, the MCKP solver was executed and the resulting configuration was observed. As the granularity of the MCKP reaction to different thresholds is dependent on the size of available components, we obtained six different FT configurations. The results including the actual configurations are shown in Figure 4. As can be seen, the MCKP tries to completely utilize the given area. Another important observation is that with the increasing size of the system, the absolute number of critical bits decreases, indicating the MCKP strategy targets the optimal configurations. Especially important observation is that not only the critical bit representation is lowering but also the absolute value of critical bits decreases while the circuit increases in size.

Furthermore, as can be seen, the MCKP selects to harden only the *crc8* and *numones* components. As you can see in Table I, this is caused by inefficiency of the selected type of redundancy methods for the first two components (i.e. the *addition* and *addconst*).

C. Real Implementation Results

From the previous step, six system configurations were obtained. These systems will be referred to by the name

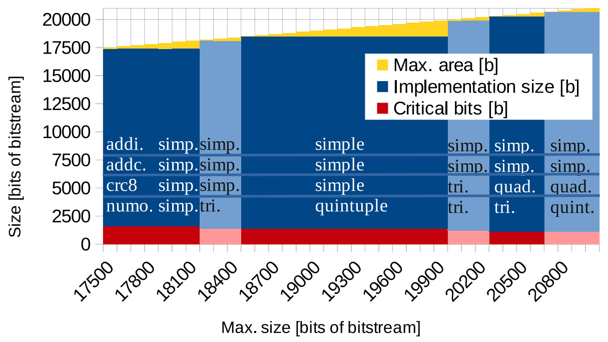


Figure 4: The theoretical computed values for the system configurations obtained by changing the available area (i.e. size) on the chip.

autocomposed_1 to *6*. We further added four reference systems that were composed without the partitioning, just by using equivalent redundancy method on the complete system. These units will be further referred to as *composed_simple*, *composed_triple*, *composed_quadruple* and *composed_quintuple*. In fact, the *autocomposed_1* system is equivalent to the *composed_simple* system, as for the smallest chip area setting, only *simple* units are selected by the MCKP solver. As a result, 9 unique systems were obtained and evaluated with the usage of the FT-EST framework on the exhaustive test of all LUT bits. The results are shown in Table II. As can be seen, after the synthesis, the autocomposed units utilize less LUT bits than indicated by the estimated values provided by a sum of components. Obviously, the synthesis is able to optimize better for larger systems, compared to the much smaller components. The numbers of critical bits, however, follow the general trend and are nearly equivalent to the values determined by the MCKP-based method.

TABLE II: Parameters of the Synthesized Systems

System Name	Num. of Inputs [*b*]	Num. of Outputs [*b*]	Used LUT bits [*b*]	Critical bits [*b*]/ KP Handicap [-]	Critical bits Repr. [%]
autocomposed_1	16 b; 16 b	5 b; 8 b	9120	1518	16.64
autocomposed_2			9856	1255	12.73
autocomposed_3			10240	1147	11.2
autocomposed_4			11684	1044	8.96
autocomposed_5			12000	968	8.07
autocomposed_6			12416	860	6.93
composed_simple			9120	1518	16.64
composed_triple			19684	1049	5.34
composed_quadruple			24448	1060	4.34
composed_quintuple			33056	1261	3.81

As can be seen, the largest autocomposed solution is still 37% smaller than the composition in which the entire system was hardened with the usage of the triple RDT. For this system, the number of critical bits is yet 18% smaller. In addition, the autocomposed units are built of the optimal components that decrease the number of critical bits with the growing size. This is not always the case for the unpartitioned units, where for example the *composed_quintuple* system contains more critical bits than its smaller counterparts. Important properties of the autocomposed and naively composed systems after their synthesis are shown in Figure 5.

Although a significant difference in the estimated and real sizes can be seen, the important fact for us is that the estimation of critical bits (i.e. the overall KP handicap) remained nearly equivalent after the systems were composed and synthesized.

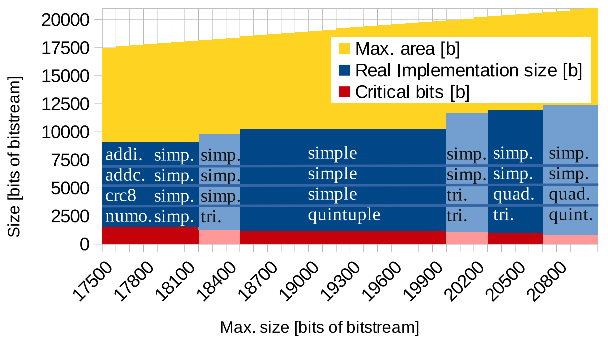


Figure 5: Properties of the synthesized configurations obtained by changing the available area (i.e. size) on the chip.

D. Comparison of Partitioned vs. Homogeneous FT Selection

As can be observed in Figure 6, systems on which equivalent redundancy method was applied to each component (i.e. left part of the chart) have significant steps (i.e. low granularity) in their implementation sizes according to utilized redundancy method. This increases the overhead of the solution, as for a given space on an FPGA, implementation that is smaller than the given space must be selected, resulting in an unused FPGA area. On the other hand, systems on which each component can be hardened using different redundancy method (i.e. right part of the chart) have better granularity among the resulting sizes of the system. Furthermore, the automatic method based on the MCKP solver follows the trend of lowering the absolute number of critical bits, which is not always the case with the *composed* units. Also, the assumption, that each component must be hardened using its most suitable redundancy method can be also confirmed from the chart. It is obvious, that the *autocomposed* units achieve equivalent or better results while occupying significantly smaller area.

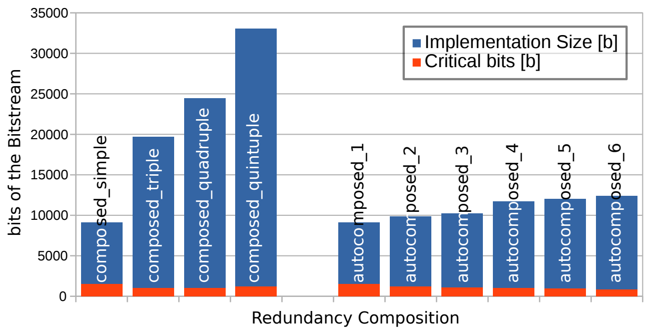


Figure 6: Comparison of partitioned (i.e. autocomposed) systems with systems hardened using equivalent redundancy method (i.e. manually composed systems).

V. Conclusions and Future Research

In this paper, the possibility to utilize the MCKP solver to assign redundancy types to components of a system was presented. The usability was put into the context of the main goal of our research which is the automation of FT design process. In our case study, the method was presented on an artificially created system which was built of four components. The MCKP solver in combination with our FT design automation toolkit was able to optimize the selection and its results contained 18% less critical bits compared to our previous naive selection. In addition, the circuits assembled according to the MCKP solver were much smaller because the solver dismisses the sub-optimal configurations. Furthermore, the results indicate that dividing the system into smaller components and searching the best solution for each component leads to considerably smaller circuit while achieving even better results.

In future, the function blocks of the automation toolkit will have to be programmably interconnected into a platform because, at the moment, a designer must execute them separately. The integration would result in the complete platform, whose user interactions would be in fact minimal, while still allowing to replace and use its blocks as in a toolkit.

Acknowledgements

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science – LQ1602, the Brno University of Technology under number FIT-S-20-6309 and the JU ECSEL Project SECREDAS (Product Security for Cross Domain Reliable Dependable Automated Systems), Grant agreement No. 783119.

References

[1] J.-C. Geffroy and G. Motet, *Design of Dependable Computing Systems*. Kluwer Academic Publishers, 2002.

[2] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[3] J. Onishi, S. Kimura, R. J. James, and Y. Nakagawa, "Solving the Redundancy Allocation Problem with a Mix of Components Using the Improved Surrogate Constraint Method," *IEEE Transactions on Reliability*, vol. 56, no. 1, pp. 94–101, 2007.

[4] W.-C. Yeh and T.-J. Hsieh, "Solving Reliability Redundancy Allocation Problems Using an Artificial Bee Colony Algorithm," *Computers & Operations Research*, vol. 38, no. 11, pp. 1465–1473, 2011.

[5] Y.-C. Liang and Y.-C. Chen, "Redundancy Allocation of Series-parallel Systems Using a Variable Neighborhood Search Algorithm," *Reliability Engineering & System Safety*, vol. 92, no. 3, pp. 323–331, 2007.

[6] K. Khalili-Damghani, A.-R. Abtahi, and M. Tavana, "A New Multi-objective Particle Swarm Optimization Method for Solving Reliability Redundancy Allocation Problems," *Reliability Engineering & System Safety*, vol. 111, pp. 58–75, 2013.

[7] G. Kanagaraj, S. Ponnambalam, and N. Jawahar, "A Hybrid Cuckoo Search and Genetic Algorithm for Reliability–Redundancy Allocation Problems," *Computers & Industrial Engineering*, vol. 66, no. 4, pp. 1115–1124, 2013.

[8] Z. Wang, T. Chen, K. Tang, and X. Yao, "A Multi-objective Approach to Redundancy Allocation Problem in Parallel-series Systems," in *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 582–589.

[9] H. Kellerer, U. Pferschy, and D. Pisinger, "The Multiple-choice Knapsack Problem," in *Knapsack Problems*. Springer, 2004, pp. 317–347.

[10] J. Lojda, J. Podivinsky, Z. Kotasek, and M. Krcma, "Data Types and Operations Modifications: A Practical Approach to Fault Tolerance in HLS," in *2017 IEEE East-West Design Test Symposium (EWDTS)*, Sept 2017, pp. 1–6.

[11] J. Lojda, J. Podivinsky, O. Cekan, R. Panek, and Z. Kotasek, "FT-EST Framework: Reliability Estimation for the Purposes of Fault-Tolerant System Design Automation," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, Aug 2018, pp. 244–251.

[12] J. Lojda, J. Podivinsky, and Z. Kotasek, "Reliability Indicators for Automatic Design and Analysis of Fault-Tolerant FPGA Systems," in *2019 IEEE Latin American Test Symposium (LATS)*, March 2019, pp. 1–4.

[13] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, ser. Algorithms and Combinatorics. Springer Berlin Heidelberg, 2007.

[14] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, ser. Wiley-Interscience series in discrete mathematics and optimization. J. Wiley & Sons, 1990.

[15] M. Graphics, "Catapult HLS," https://www.mentor.com/hls-lp/catapult-high-level-synthesis/, 2017, accessed: 2017-07-07.

[16] Xilinx Inc., "ISE Design Suite," https://www.xilinx.com/products/design-tools/ise-design-suite.html, 2017, accessed: 2017-07-07.

[17] N. Dorairaj, E. Shiflet, and M. Goosman, "PlanAhead Software as a Platform for Partial Reconfiguration," vol. 55, no. 84, 2005, pp. 68–71.