Matrix Description Specifics for Synthesis of Self-Checking Micro-Programmed Control Units

Serge N. Demidenko*, Eugene M. Levine**
*Institute of Information Science & Technology, Massey University, New Zealand
**Rainbow Technologies Inc., Belarus

Abstract

The paper describes general approach to the synthesis of concurrently self-checking control units with pre-defined level of the fault coverage, and discusses some specific features of the matrices used in the synthesis. The micro-program control is considered though the results can easily be extended to other control architectures.

1. Introduction

Micro-programmed Control Units (MPCUs) have been widely used in electronic systems. MPCU is designed around a memory containing micro-instructions. Each micro-instruction consists of the control code and next address code. The sequence of the codes read out of the memory is determined by the sequence on the output of the next address generator. The generator produces a new address depending on the code in the address field of the current micro-instruction, on the external control signals (start, stop, set, etc.), and on the values of some logic flags coming from the controlled device (these flags are used to organize conditional jumps in the program). The architecture provides ease in programming, and high flexibility of the control. It is also characterized by simplicity of the hardware realization.

Two major approaches to MPCU on-line testing are usually employed in order to satisfy high dependability criteria [1]. The first one is related to checking the control outputs of the MPCU by employing the techniques developed for the highly dependable data path architectures [2]. The other approach includes techniques ranging from MPCU state description based on error-detecting codes to control-flow monitoring by means of on-chip or off-chip watchdog machines [1, 3-5]. The approaches differ in hardware overhead, performance degradation, and error detection latency. Consequently, they can be efficient in some application areas, and inefficient in the others.

This paper deals with the MPCU self-checking based on the technique employing the use of the special check tokens (keys) incorporated into the body of the micro-program. A key (binary word) is put into correspondence to each micro-instruction or to a set of micro-instructions. The keys are then interleaved with micro-instruction during the program execution. This allows checking the control flow by comparing the actual sequence of the check keys with a stored (or calculated) reference sequence corresponding to fault-free operation of MPCU. Different generation and data compression techniques can be used to process sequences of reference and actual keys leading to a wide variety of possible architectural realizations of self-checked MPCUs [4, 5].

2. On-line program flow monitoring

MPCU can be described with the use of a flow graph representing the operation in terms of succession of generated micro-instructions. Every vertex corresponds to one micro-instruction. Check keys are used to code the operator vertices of the flow graph. Generally, the graph can be represented as set of k linear disjoint sections ($k \in \{1,2,3,...,N\}$, N is the total number of vertices) with some i-th section having l_i vertices ($l_i \in \{1,2,3,...,N\}$, $i \in \{1,2,3,...,k\}$). Thus j-th vertex of i-th linear section can be denoted as $Q_{i,j}$. Faults in the MPCU micro-instruction fetch sequence correspond to faulty transitions between vertices, denoted, for instance, as $Q_{i\,j} \to Q_{yz}$, or $Q_i \to Q_j$ if just a linear flow graph is used.

Probabilistic approach is used for quality estimation of the flow graph checking. The fault detection probability P_d is the quality quantitative of the main interest. Expression $P_d(Q_{i\,j} \to Q_{yz})$ denotes the probability that a single faulty transition from vertex Q_{ij} to vertex Q_{yz} is detected by means of on-line checking facilities. The probability value depends on: flow graph itself, type and distribution of the check keys, checking strategy, and check key processing methods.

Obviously, in every particular case the probability $P_d(Q_{ij} \rightarrow Q_{yz})$ is either equal to one (when the faulty transition from Q_{ij} to Q_{yz} is detected), or otherwise equal to zero (when the faulty transition remains undetected). These notations can be used to get a general description for the fault detection in the MPCU program flow [6].

Complex event of detection of the faulty transition $Q_{ij} \rightarrow Q_{yz}$ includes the following sub-events: *(a)* fault occurrence in some vertex Q_{ij}, *(b)* occurrence of the transition from vertex Q_{ij} to vertex Q_{yz}, *(c)* detection of the faulty transition by means of diagnostic tools.

The probability that a fault is occurred in some vertex of a flow graph depends on the reliability characteristics of the MPCU hardware, on the flow graph pattern, on the ranges of the input data as well as on the distribution of the data over these ranges, and on the probabilities of generation of various flags used in conditional jump operations (case statements, "if", "while", etc.). The numerical evaluations for the probabilities can be obtained theoretically from analytical computations or empirically from observations on actual MPCU devices. The values can be summarized a vector ***F*** (*Vector of Fault Occurrence*) of dimension *N* (where *N* is the number of flow graph operator vertices). Each element of ***F*** corresponds to a vertex of the flow graph and is equal to the probability that a fault occurs there.

Probabilities that the faulty transitions occur between two specific vertices can be presented by the square matrix ***T*** (let us call it *Transition Matrix*) of dimension *N x N*. Each element of it is equal to the probability of a faulty transition between the two vertices. The probability evaluations can be found empirically, analytically or from computer simulation.

Probabilities of detecting faulty transitions by means of on-line checking tools can be given by a square matrix ***D*** (*Detection Matrix*) of dimension *N x N*. Assuming that only a single transition error occurs, and the checking is performed on every cycle of MPCU operation, an element of such a matrix is equal to *1* when a faulty transition is detected; otherwise it is equal to zero. If the checking is organized in the interval-by-interval mode and an occurrence of multiple errors is possible [5], the probabilities may have values between zero and one.

Matrix expression for the probability of detection of a faulty transition can be presented as: $\boldsymbol{P} = \boldsymbol{F} \times \boldsymbol{T} * \boldsymbol{D}$, where "$*$" denotes the element-by-element (not the conventional matrix-by-matrix) multiplication. Elements of the matrix ***P*** are the probabilities of detection of faulty transitions between the corresponding vertices of the flow graph. By summing matrix elements over different ranges, it is possible to obtain probability estimates for partial (on some flow-graph interval) and total checking coverage.

3. Synthesis of self-checking MPCU

General synthesis procedure for self-checking MPCU with specified fault coverage includes the following steps.

1. By analyzing the micro-program flow-graph, the architecture of MPCU, and other relevant data (probability of external control signals and conditional jumps in the program initiated by the flag signals, etc.) the elements of vector ***F*** and matrix ***T*** are found
2. General attainability of the fault detection is tested by calculating the maximal possible fault coverage level for given ***V*** and ***T***. At this stage it is assumed that all the elements of matrix ***D*** are ones, that is all possible faulty transitions can be detected. If it is found that the required level of the fault coverage can not be reached, then the algorithm, instruction set, data range, etc., must be modified, while the MPCU architecture must be re-configured, as necessary. Return to the step (1). Once the required coverage is attainable proceed to step (3).
3. Elements of the matrix ***D*** are defined from the required fault coverage and from the values of elements of the vector ***V*** and matrix ***T***.
4. Checking strategy is selected, and the values of the check keys are assigned to the flow graph.
5. Corresponding changes are introduced into the MPCU hardware and software to realize the checking with the use of the assigned keys and selected strategy [6]

4. Conclusion

The above material is just a brief introduction into the approach of synthesis of MPCU providing pre-defined fault coverage. The properties of the matrixes ***V***, ***T***, and ***D***, as well as validity of employing different types of data compression will be discussed at the workshop. Our research is in its way, and we plan that the full-size presentation and publication on the topic will appear in one of the forthcoming conferences this year.

References

1. S. Hellebrand, H.-J. Wunderlich, A. Hertwig: Synthesizing Fast, Online-Testable Control Units, IEEE Design & Test on Computers, Vol. 15, No. 4, 1998, pp. 36-41
2. D. K. Pradhan: Fault-tolerant computer system design, Prentice-Hall, 1996
3. R. A. Parekhji, G. Venkatesh, S. D. Sherlekar: Concurrent error detection using monitoring machines, IEEE design & Test of Computers, Vol. 12, No. 3, 1995, pp. 24-31
4. A. Mahmood, E. J. McCluskey: Concurrent error detection using watchdog processors – a survey", IEEE Trans. on Computers, v. 37, pp. 160-174, 1988
5. S. N. Demidenko et.al.: Concurrent self-checking for microprogrammed control units: an analytical survey, IEE Proc, v.138, pp. 377-388
6. S. N. Demidenko et. al.: New approach to synthesis of self-checking microprogrammed control units with specified fault-detection probabilities, IEE Proc, v.138, pp. 389-396