A SysML Model for Code Correction and Detection Systems

Stefan Stancescu, Lavinia Neagoe, Raluca Marinescu, Eduard Paul Enoiu
University POLITEHNICA of Bucharest, Mälardalen University, Västeräs
stst@elia.pub.ro, rlavinia2002@yahoo.com, {rmu09001, eeu09001}@student.mdh.se

Abstract - The Unified Modeling Language (UML) is a well known approach for specifying and designing software components. UML for hardware designs of embedded systems is also possible in the simulation process, when the hardware is in the software form. The large number of tools for UML, the general adoption of this technology for heterogeneous system design and verification, makes UML a very powerful and robust design instrument. Based on UML, the SysML [1] language has been developed in order to support all the details of system designs. SysML extends UML towards the systems engineering domain. As a good example, a SysML model for hardware components that perform error detection and correction, based on polynomial registers mod p(x), will be presented. The approach is justified as efficient and flexible.

I. INTRODUCTION

We represent complex electronic devices by specifying internal structure (components and links) of each included block through various hardware design languages HDL, as the one presented in [2]. It is common to directly combine primitive building-blocks available, such as logic gates, with larger blocks already defined elsewhere. A vast component library is available in various technological variants, easy to be included in new designs. The problem of choosing the appropriate component for the specific requirement, and inclusion of that component at the appropriate system level, remain a major task in the optimum design process.

Verilog HDL, as a well accepted HDL, allows circuits to be described in two ways, structural (how they are built), or in terms of behavior, (what they do). It is important to remember that, accepting circuits' behavior description, rather than its structure, the synthesis tool has to translate the description into logic. The synthesis tool, although very good at its job, may not always translate the description into the intended logic and may not produce the most efficient circuit possible. For these reasons, it is useful, if not necessary, to achieve early behavior and logic validation of how description will be translated into logic. This level of abstraction allows designers to describe the circuit's functionality in an algorithmic way. In this approach, the designer describes the behavior of the circuit without any consideration to how the circuit transfers to hardware.

SysML[1] is an extension of UML that allows modeling a broad range of complex systems, including SoC (system-on-Chip) designs. The SysML approach introduces the possibility of describing the system as a set of interconnected components both software and hardware.

II. TRANSLATION FROM SYSML TO VERILOG

In order to find how UML specifications from [1] are translated into Verilog code it is necessary to define a template code. Ideally there is a one to one mapping, but in practice, the realization of such mapping is not straightforward.

The mapping between SysML and Verilog is defined as follows:

- SysML parts mapped to Verilog modules;
- SysML flow port mapped to Verilog signals and ports;
- SysML allocations mapped to Verilog process(always, initial);

With this mapping we are able to create a file for each module that must include: module definition, port declarations and process declaration.

Table 1.
SysML parts mapped to Verilog modules.

	SysML	Verilog
Modules	Block	module reed_solomon_encoder() ... endmodule
Ports	Interconnections	input [n:0] d; input clock; output [n:0] q; reg [n:0] out; wire value;

Process	Sequence diagram	always @(posedge clock)
		if(statement)
		begin
		...
		end

III. ECC DECODING

Error Correcting Codes, ECC's are fast hardware mechanisms that trade redundancy for reliability. Cyclic ECC's, based on polynomials, offer a fast and straightforward coding process, with hardware implemented modulo p(x) registers, working on line with the data codes.

The Cyclic ECC's redundant bits are immediately available after the data ones. In the absence of errors, the inconveniences in speed of the few redundancies are negligible in the economics of transmission. It is not this case when errors occur. Then, the decoding process must

analyze the erroneous data and redundant bits and decide where the data is corrupted and what original data must be inserted instead. As the errors are statistically rare events, these inconvenients of long decoding intervals are also rare and, generally, do not contribute significantly to performance degradation. In the view to minimize the above-mentioned decoding interval, complex hardware solutions are conceived, with parallel working components. The code polynomial itself is provided from many simple polynomials, multiplied, in view to permit such a parallel processing.

An appropriate example is Reed Solomon code , with the associate Berlekamp-Massey decoding algorithm. Hardware implementations of fast Cyclic ECC's are complex and represent a difficult task for an optimum design. This is the reason to involve advanced design methods to reduce the complexity burden.

Contemporary applications of Reed Solomon codes include magnetic and magneto-optic digital recording, data communications on various media, spatial communications included. There are many variations on the Reed Solomon error correcting codes theme, due to many different applications requesting for specific polynomials and performances (speed, error correcting capacity, complexity, power limitations, etc.). Useful references are [5,9,10,11] books. Various serial or parallel decoding architectures, with co-design solutions that mixes hardware and software algorithms are known. An accurate, flexible and reliable design procedure is needed in view to quickly fulfill the gap between specifications and appropriate circuit implementation. The UML, with the SysML variant above mentioned, is a solution we choose to optimize the design task.

Decoding ECC implies the extraction of two information entities from the received word. These are the set of Partial Syndromes, and the Error Locator Polynomial. From these two parameters, Error Locations and Error magnitudes are extracted.

We will present an example of an UML simplified model for a universal Reed-Solomon decoder, to whom we applied the SysML block description diagram.

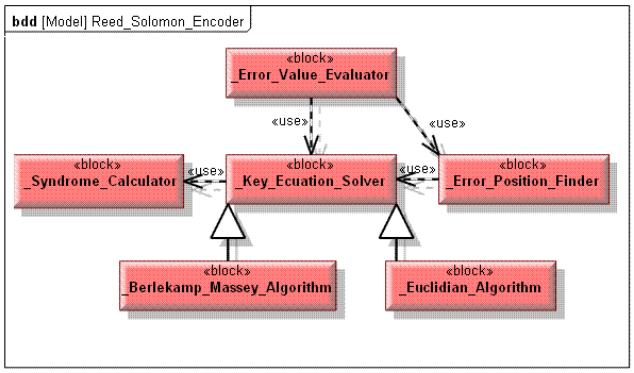

Figure 1. SysML Block Description Diagram of a simplified Reed-Solomon Decoder

The SysML block description diagram in figure 1 shows relationships between the component blocks. There are six generic blocks each performing an important functionality for the model. The Key_Equation_Solver implements one of the specific algorithms for ECC and here we have added, as an example, the Berlekamp-Massey algorithm and the Euclidian algorithm. The relationship between these three blocks is generalization. We can add up to the block any type of algorithm with the condition of specifying that algorithm through specific requirement diagrams, but also with more detailed block diagrams. Other blocks of the diagram are the Syndrome_Calculator, the Error_Position_Finder and the Error_Value_Evaluator.

The SysML possibilities of descriptions are numerous and the solution for describing ECC may not always be the same for all problems, this is why different granularities of the diagrams are allowed and we might even say that they are mandatory.

The SysML diagram is very generic, but another level of granularity can be obtained from SysML exactly as in the UML Class Diagram presented in figure 2. Afterwards, each block can be described in more detail, by its own block diagram.

In the "Syndrome Calculator" Class, errors are detected by calculating the syndrome polynomial, which is used by the "Key Equation Solver" Class to solve the key equation. The structure is centered on the "Key Equation Solver" class from which other two classes are derived: "Berlekamp-Massey" and the "Euclidian". The Berlekamp-Massey algorithm is a shift-register synthesis algorithm that has a complex structure but use fewer gates to be implemented than the Euclidian algorithm [5]. Based on the "Error Position Finder" class locations of the errors are determined. The magnitude of the errors is determined by the error evaluator polynomial in the "Error Value Evaluator" Class.

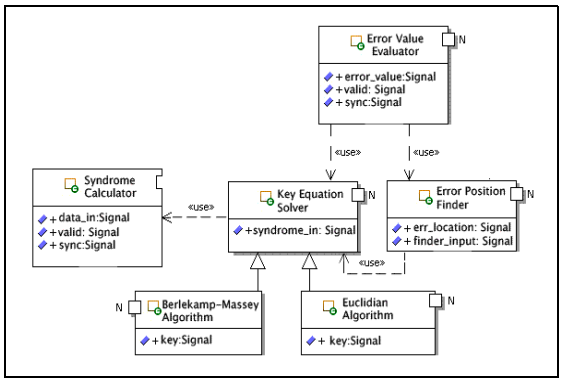

Figure 2. UML Class Diagram of a simplified Reed-Solomon Decoder

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI
xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:uml="http://schema.omg.org/spec/UML/2.2"
xmi:version="2.1">
 <uml:Package name="RS_Model"
visibility="public">
    <packagedElement xmi:type="uml:Package"
name="RS_Requirements" visibility="public"/>
    <packagedElement xmi:type="uml:Package"
```

```
name="RS_Use_Cases" visibility="public"/>
    <packagedElement xmi:type="uml:Package"
name="RS_Structure" visibility="public">
     <packagedElement xmi:type="uml:Class"
name="RS_Block" visibility="public"/>
     <packagedElement xmi:type="uml:Class"
name="Key_Equation_Solver" visibility="public">
… … … … …
     </packagedElement>
     <packagedElement xmi:type="uml:Class"
name="Syndrome_Calculator" visibility="public">
… … … … …
     </packagedElement>
     <packagedElement xmi:type="uml:Class"
name="Error_Position_Finder" visibility="public">
… … … … …
     </packagedElement>
     <packagedElement xmi:type="uml:Class"
name="Error_Value_Evaluator" visibility="public">
… … … … …
     </packagedElement>
… … … … …
```

Figure 3. XMI file to be translated to Verilog

The SysML diagrams can be exported to XMI as in figure 3. A parser and a translator will be used to transform XMI to VHDL based on the rules described in table 1.

The SysML or UML diagram can be mapped directly in Verilog by direct mapping as we can see from the next code:

```
module RSDecoder (data_in, valid, sync,
error_value, corr_recword, key, syndrome_in,
err_location, finder_input;);

input  data_in;
input valid, sync;
output error_value;
output corr_recword;,
wire key;
wire syndrome_in;
wire err_location;
wire finder_input;

//instantiation of Syndrome_Calculator  module
Syndrome_Calculator Syndrome_Calculator (data_in,
validm sync);
Key_Solver Key_Solver (syndrome_in,key);
Error_Finder Error_Finder (err_location,
finder_input);
endmodule
```

IV. CONCLUSION

We have presented a methodology for specifying and verifying ECC circuits by using the SysML meta-model to obtain the Verilog implementation for a specific circuit. The methodology includes the SysML Block Description Diagram, Requirements Diagram and State Machine Diagram conception, the XMI generation and XMI-XSLT translation to Verilog. This methodology can be applied for implementing various circuits with a diverse range of polynomials p(x), code length, detecting and correcting capabilities, specified in the appropriate and detailed SysML model.

REFERENCES

[1]Object Management Group, SysML partners. Systems Modeling Language (SysML) Specification. September 2009.

[2]Cadence Design Systems, Open Verilog International. Verilog, language reference manual. August 1, 1996.

[3]J. Rumbaugh, I. Jacobson, and G. Booch, The Unified Modeling Language Reference Manual. Addison-Wesley, 1998.

[4]F. Pugnetti, Using Model Driven Architecture (MDA), July 2009.

[5]Shu Lin, D.J. Costello, Error Correcting Codes, Prentice Hall, 2'nd Ed., 2004

[6]Friedenthal, S., Burkhart, R. "Extending UML From Software To Systems," Proceedings Of The INCOSE 2003 International Symposium, 2003.

[7]Erik Herzog, Asmus Pandikow, "SysML – an Assessment", Proceedings of the 15th INCOSE International Symposium, 2005

[8]M. Mura, A. Panda, M. Prevostini, Executable Models and Verification from MARTE and SysM a Comparative Study of Code Generation Capabilities, DATE 2008

[9]E. R. Berlekamp, "Algebraic Coding Theory," *Aegean Park Press*, 1984.

[10] R. E. Blahut, "Algebraic Codes for Data Transmission," *Cambridge University Press*, 2003.

[11] S. B. Wicker, "Error Control Systems for Digital Communication and Storage," *Prentice Hall Inc.*, 1995.