Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 17, No. 2, 2025

Model-driven Reed-solomon encoder and decoder design

Yang Nie

School of Physics and Electronic Information Engineering, Jining Normal University, Ulanqab 012000, China

Abstract: This paper presents a model-driven design methodology for Reed-Solomon (RS) encoder/decoder systems, leveraging Model-Based Design (MBD) as the core framework throughout the entire development lifecycle, from requirements specification to verification. By integrating mathematical modeling, algorithmic design, and hardware implementation within a unified environment, the proposed approach enables automated code generation for both RS encoder and decoder modules, along with corresponding test benches for comprehensive functional verification

Keywords: FPGA; Model-driven; Reed-solomon Code; MBD.

1. Introduction

To evaluate the performance of novel communication algorithms and architectures, engineers must develop conceptual validation prototypes and prepare new designs for field trials [1] . Typically, FPGA hardware with embedded processors is employed to construct prototypes as integral components. These platforms, commonly referred to as hardware testing platforms, facilitate rapid prototype development and enable the testing of new technologies and design iterations. However, developing communication prototypes and FPGA-based testing platforms often poses challenges for typical engineering teams without external support. While engineers possess extensive expertise in signal processing and communication algorithm development, they often lack experience in hardware implementation. This experience gap is further exacerbated by the absence of integrated tools and streamlined workflows. Although engineers frequently utilize high-level languages such as MATLAB, hardware design typically relies on specialized tools and HDL (Hardware Description Language), creating a disconnect between algorithmic development and hardware implementation.

The traditional manual development process has become increasingly inadequate to meet current industry demands. To address these challenges, integrating model-based development (MBD) with visual design methodologies into circuit design is imperative [2]. MBD ensures that the final product aligns with system requirements by employing models as the central artifact throughout the development lifecycle. This approach enables multidisciplinary engineering teams to collaborate effectively, fosters seamless communication across design stages, facilitates early error detection and correction, and automates the generation of robust, efficient, and high-quality embedded software code and synthesizable HDL.

This paper demonstrates the efficacy of MBD in communication system design by implementing a Reed-Solomon (RS) encoder/decoder. The paper is structured as follows: Section 2 introduces the MBD methodology; Section 3 reviews the theoretical foundations of RS encoding/decoding; Section 4 presents the core contribution of this work, wherein the RS encoder/decoder is designed and validated using MBD; finally, Section 5 concludes the paper with key findings and future directions.

2. Model-Based Design

Model-Based Design (MBD) enhances design quality and expedites design and verification processes through the use of executable specifications. These specifications serve as the foundation for hardware-software partitioning, automated code generation for both hardware (e.g., HDL) and software components, and system-level validation within the context of the complete system architecture [3,4] . Central to the MBD methodology is the system model, which spans the entire development lifecycle—from requirements elicitation to design implementation and testing (Fig. 1). By establishing floating-point, fixed-point, and system-level models that encapsulate comprehensive design requirements, MBD enables cross-functional engineering teams to collaborate effectively and maintain seamless communication across development phases.

A key advantage of MBD is its enforcement of continuous verification and validation (V&V) throughout the design process. Leveraging specialized MBD tools, engineers can conduct iterative testing at each developmental stage, ensuring design correctness and compliance with system requirements. This approach minimizes error propagation, reduces rework, and facilitates early detection of inconsistencies between model specifications and implementation artifacts. Through automated code generation and integrated V&V frameworks, MBD streamlines the transition from conceptual design to production-ready hardware and software, thereby accelerating time-to-market while maintaining rigorous quality standards.

Model-Based Design (MBD) facilitates the rapid and cost-efficient development of dynamic systems, encompassing control systems, signal processing algorithms, and communication protocols. Central to MBD is the use of executable specifications, which are iteratively refined throughout the development lifecycle. Following model development, comprehensive simulations validate the model's correctness against system-level requirements, enabling exhaustive testing and verification of all design aspects. Fig. 2 illustrates the collaborative MBD workflow, which integrates MATLAB/Simulink for system modeling with HDL toolchains for hardware implementation.

The HDL workflow advisor automates FPGA

programming for leading platforms (e.g., Xilinx® and Altera®), offering granular control over HDL architecture, critical path analysis, and hardware resource utilization estimation. This toolchain generates both VHDL and Verilog test benches, enabling rapid verification of synthesized HDL code. HDL cosimulation model is applied for performing HDL cosimulation with Simulink and an HDL simulator, such as Cadence Incisive or Mentor Graphics ModelSim. FPGA-in-the-loop (FIL) cosimulation model is applied for verifying design with Simulink and an FPGA board.

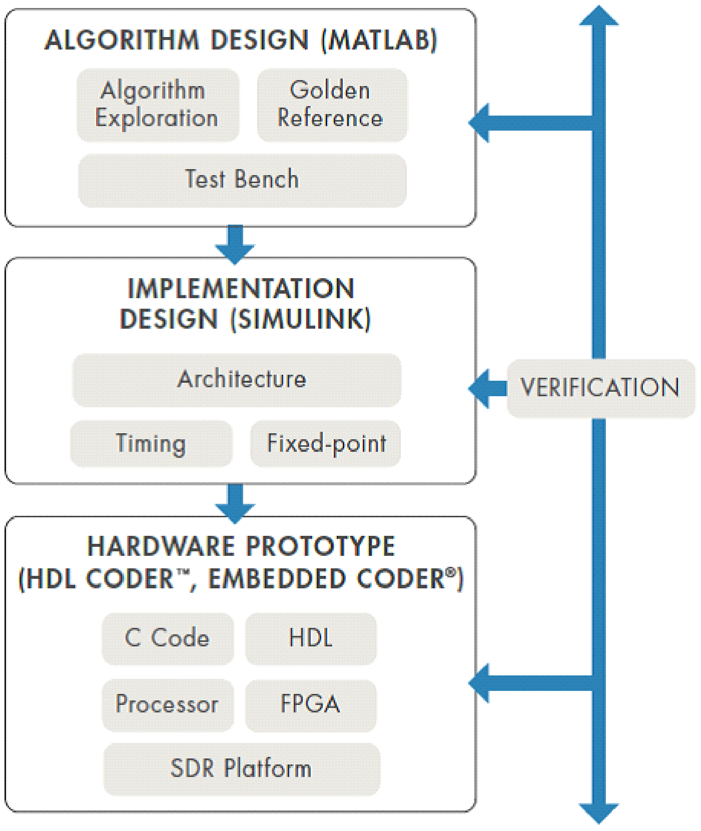


Fig.1 The Workflow of MBD

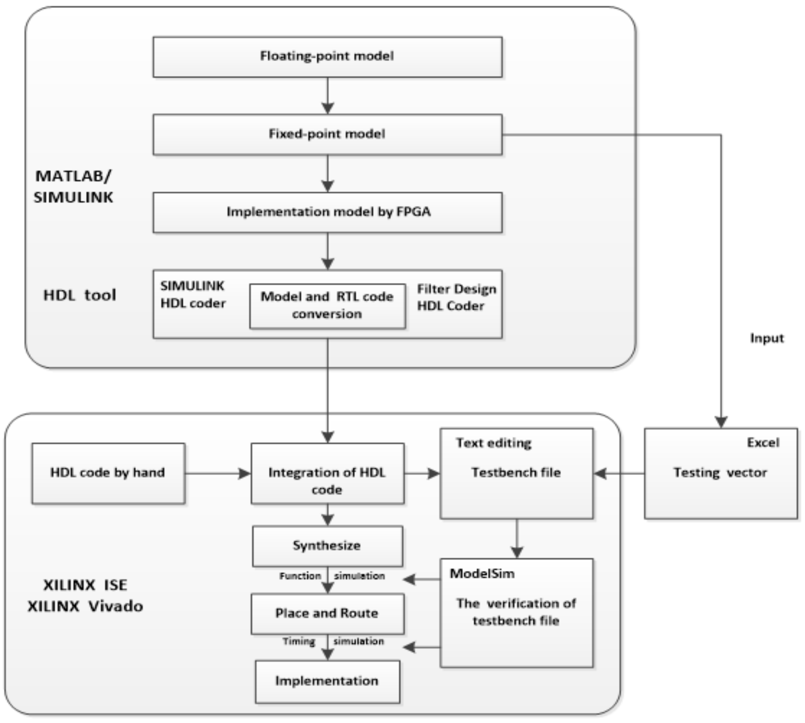


Fig. 2 The Workflow of MBD based on XILINX development tools

3. Reed–Solomon Codes

Reed–Solomon codes named after Reed and Solomon [5] following their publicationin 1960 have been used together with hard decision decoding in a wide range ofapplications. Reed–Solomon codes are maximum distance separable (MDS) codesand have the highest possible minimum Hamming distance. The codes have symbols from F_q with parameters $(q-1,k,q-k)$. They are not binary codes but frequently are used with $q=2^m$, and so there is a mapping of residue classes of a primitive polynomial with binary coefficients [6] and each element of F_{2^m} is represented as a binarym-tuple. Thus, binary codes with code parameters $\left(m(2^m-1), km, 2^m-k\right)$ can be constructed from Reed–Solomon codes. Reed–Solomon codes can be extended in length by up to two symbols and in special cases extended in length by up to three symbols. In terms of applications, they are probably the most popular family of codes.

The IEEE 802.16 Broadband Wireless Access standard [6] employs a shortened version of the RS $(N=255, K=239, T=8)$ code generated on GF(256), where N is the byte length of the coded codeword, K is the byte length of the input information before the encoding, and T is the maximum number of bytes that can be corrected. The code primitive polynomial is

$$p(x)=x^8+x^4+x^3+x^2+1 \tag{1}$$

The code generator polynomial is

$$g(x)=(x+\alpha)(x+\alpha^2)(x+\alpha^3)\text{L}\ \ (x+\alpha^{2T}) \tag{2}$$

Where $\alpha^i\left(1\le i\le 2T\right)$ are are nonzero elements of GF $\left(2^m\right)$.

If the input information is $m(x)$, the code word can be obtained by using the coding formula of the cyclic code.

$$c(x)=m(x)x^{N-K}+\left[x^{2T}m(x)\right]\bmod g(x) \tag{3}$$

The code word gets the RS code after truncating the bit of information and deleting the check bit. In the IEEE 802.16d protocol, the supported RS codes include RS (32, 24, 4), RS (40, 36, 2), RS (64, 48, 8), RS (80, 72, 4), RS (108, 96, 6), RS (120, 108, 6). RS encoder introduces parity symbols, which are used by the RS decoder to detect and correct symbol errors. The code can correct up to symbol errors in each codeword.

4. The design RS encoder / decoder by MBD

This section shows how to implement encoder and decoder for the IEEE 802.16 standard using the MBD method, which includes the encoder and decoder design of RS code. Integer-Input RS encoder block and integer-Input RS decoder block of Simulink library. In Fig. 3, it shows the model diagram of the entire design, which includes the source, the RS subsystem and destination. The RS subsystem is composed of the RS encoder module and the RS decoder module, and the structure is shown in Fig. 4. The ErrorGen subsystem adds noise to the RS encoded message.

The Source repeatedly transmits the message followed by a guard interval. The model has parameters messagelength, for the number of symbols in the message to encode; and period, which includes the messagelength and the length of the guard interval. The guard interval between messages accommodates the latency of the encoder adding parity check symbols to the message, and the decoder performing a Chien search. In the initFcn callback of the model, the messagelength is set to 36 and period is set to 236 (which suggest that the guard interval has a length of 200 symbols).

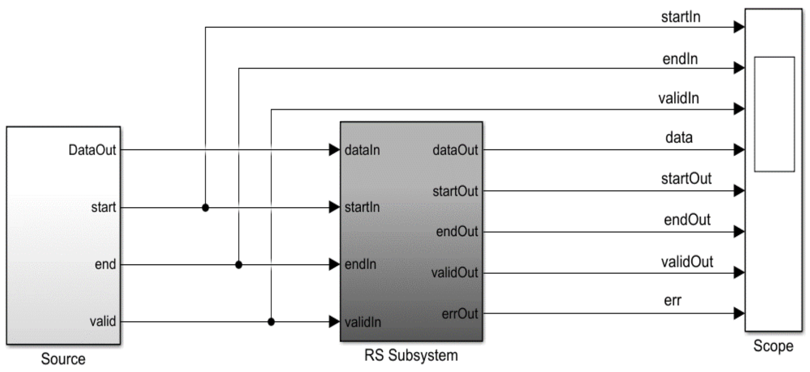


Fig. 3 The block diagram of the whole design model

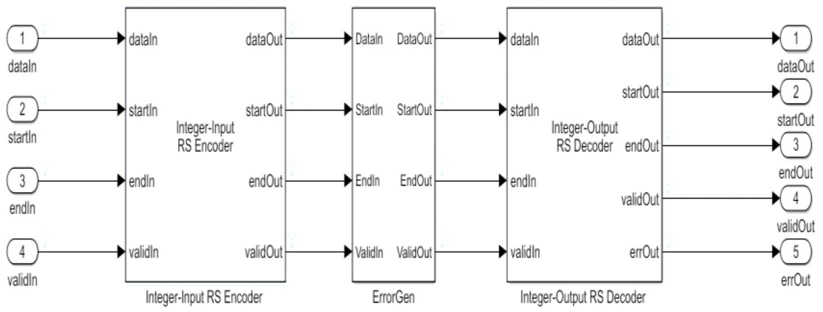
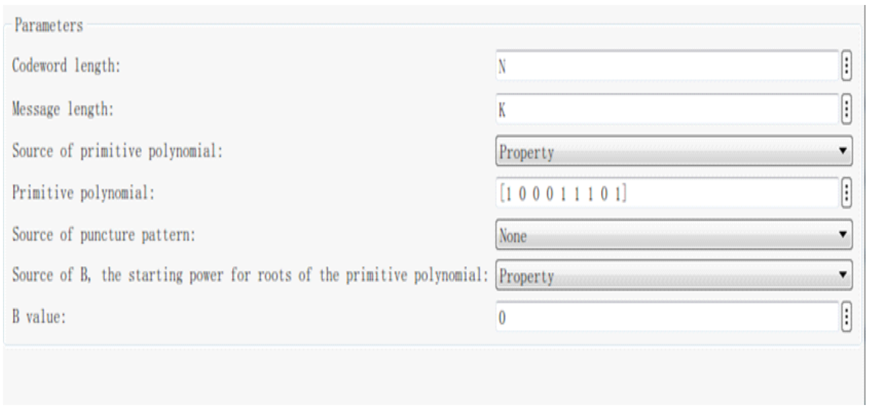


Fig. 4 The block diagram of the whole design model

4.1. The design of RS encoder by MBD

The RS encoder is designed by the Integer-Input RS Encoder block, which encode data using a Reed-Solomon encoder.The Integer-Input RS Encoder block creates a Reed-Solomon code with message length K and codeword length N. We can specify N and K directly in the block dialog. The symbols for the code are integers between 0 and 2M-1, which represent elements of the finite field GF(2M)[7]. The default value of M is the smallest integer that is greater than or equal to log2(N+1), that is, ceil(log2(N+1)). We can change the value of M from the default by specifying the primitive polynomial for GF(2M). An (N, K) Reed-Solomon code can correct up to floor((N-K)/2) symbol errors (not bit errors) in each codeword[8,9]. The configuration parameters of the Integer-Input RS Encoder block are shown in Fig. 5, where primitive polynomial parameter is determined by (1).

The following Fig. 6 illustrates possible input and output signals to this block when codeword length N is set to 7, message length K is set to 5, and the default primitive and generator polynomials are used.Suppose M = 3, N = 23-1 = 7, and K = 5. Then a message is a vector of length 5 whose entries are integers between 0 and 7. A corresponding codeword is a vector of length 7 whose entries are integers between 0 and 7.


Parameters

Codeword length:	N
Message length:	K
Source of primitive polynomial:	Property
Primitive polynomial:	[1 0 0 0 1 1 1 0 1]
Source of puncture pattern:	None
Source of B, the starting power for roots of the primitive polynomial:	Property
B value:	0

Fig. 5 The parameter configuration of the Integer-Input RS Encoder

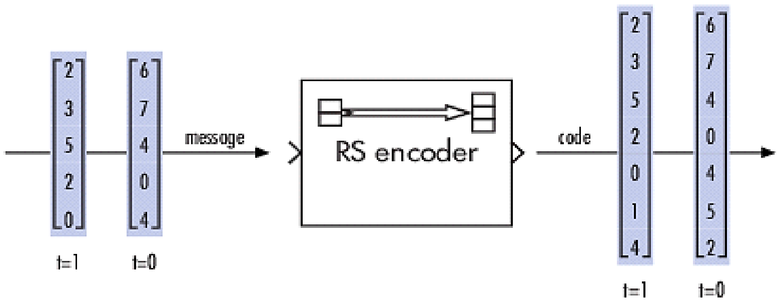
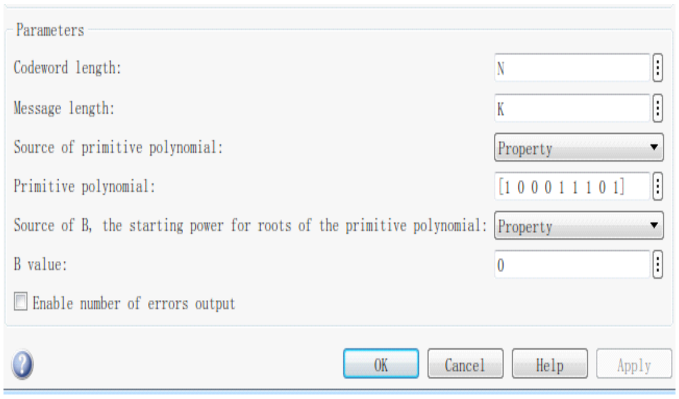


Fig. 6 The relationship between the input and output of the RS encoder

4.2. The design of RS decoder by MBD

The RS decoder is designed by the Integer-Input RS DEcoder block, which decode Reed-Solomon code to recover integer vector data. This block uses the Berlekamp-Massey decoding algorithm [10]. The Integer-Output RS Decoder block recovers a message vector from a Reed-Solomon codeword vector. For proper decoding, the parameter values in this block must match those in the corresponding Integer-Input RS Encoder block. If the decoder is processing multiple codewords per frame, then the same puncture pattern holds for all codewords.

The block can output shortened codewords when the Shortened message length S is specified. In this case, the codeword length N and message length K should specify the full-length (N, K) code that is shortened to an (N–K+S, S) code. The second output is the number of errors detected during decoding of the codeword. A -1 indicates that the block detected more errors than it could correct using the coding scheme. An (N,K) Reed-Solomon code can correct up to floor((N-K)/2) symbol errors (not bit errors) in each codeword. The configuration parameters of the Integer-Input RS DEcoder block are shown in Fig. 7.


Parameters

Codeword length:	N
Message length:	K
Source of primitive polynomial:	Property
Primitive polynomial:	[1 0 0 0 1 1 1 0 1]
Source of B, the starting power for roots of the primitive polynomial:	Property
B value:	0

☐ Enable number of errors output

Fig. 7 The parameter configuration of the Integer-Input RS Decoder

4.3. The Simulation of RS encoder/decoder by MBD

The Logic Analyzer can be used to view multiple signals in one window and viewing signals this way makes it easier to observe transitions. The simulation results of Fig. 8 show that, in the Logic Analyzer output the inputdata signal represents the input of the RS encoder block and this is the 36 byte message given in the IEEE 802.16 specification. The encoded data shows the output of the RS encoder block. Note that the IEEE 802.16 specification performs puncturing of the parity bytes and retains only the first four bytes of the 16 bytes. In this demo all 16 bytes of parity are used and the first four bytes of parity are 49, 31, 40, and BF, matching the IEEE 802.16 specification.

The errdata signal represents the encoded data with noise added in the specified noise locations. These noise locations are marked with 1s in the inserterr signal.The decoded and corrected message out of the RS decoder block is shown by the outputdata signal. Note that the RS decoder block introduces about 3 period lengths of latency. Observe outputdata to see that the errors induced by noise are corrected.

In the development flow based on MBD, HDL Workflow Advisor is a MATLAB tools for supporting the FPGA design , which verifies the model , automatically generates HDL code and conFig.s FPGA. Using XILINX FPGA development ISE as synthesis tool, the HDL code automatically generated of RS encoder/decoder is synthesized and implement by XILINX FPGA chip. The following simulation result shows the ModelSim HDL simulator after running the generated, which is shown in Fig. 9. It can be seen from the simulation results that it is not only accurate, but also fast and effective.

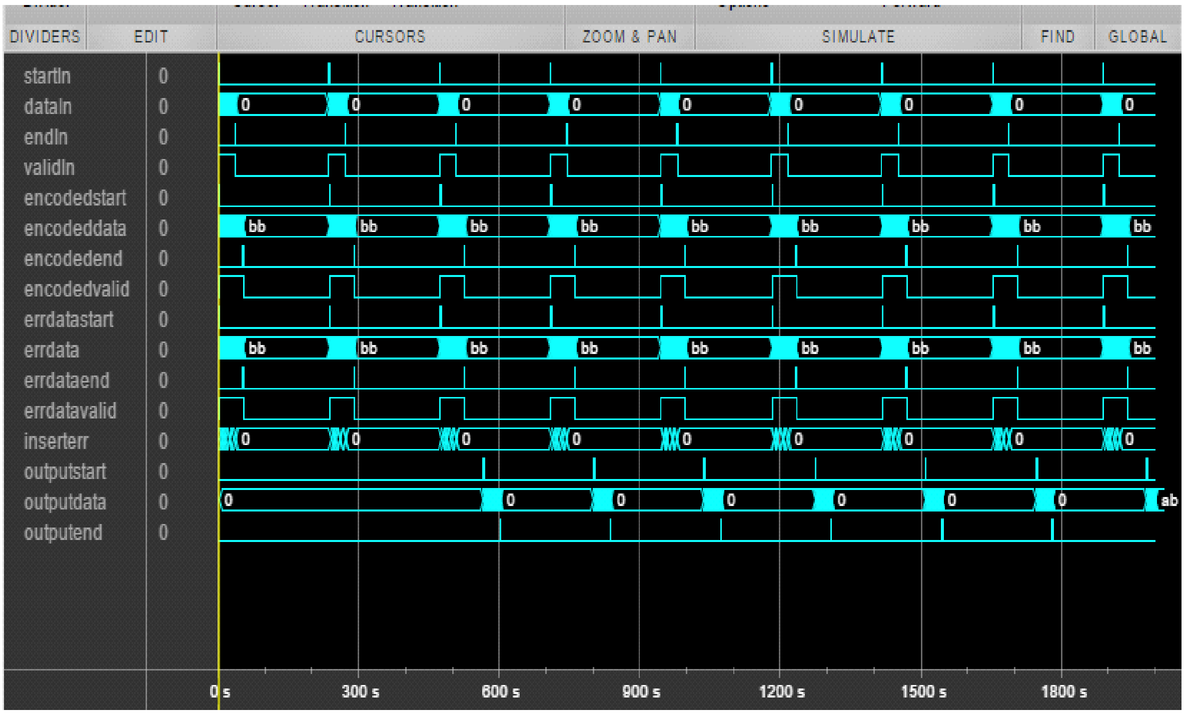


Fig. 8 The simulation results of the RS encoder/decoder

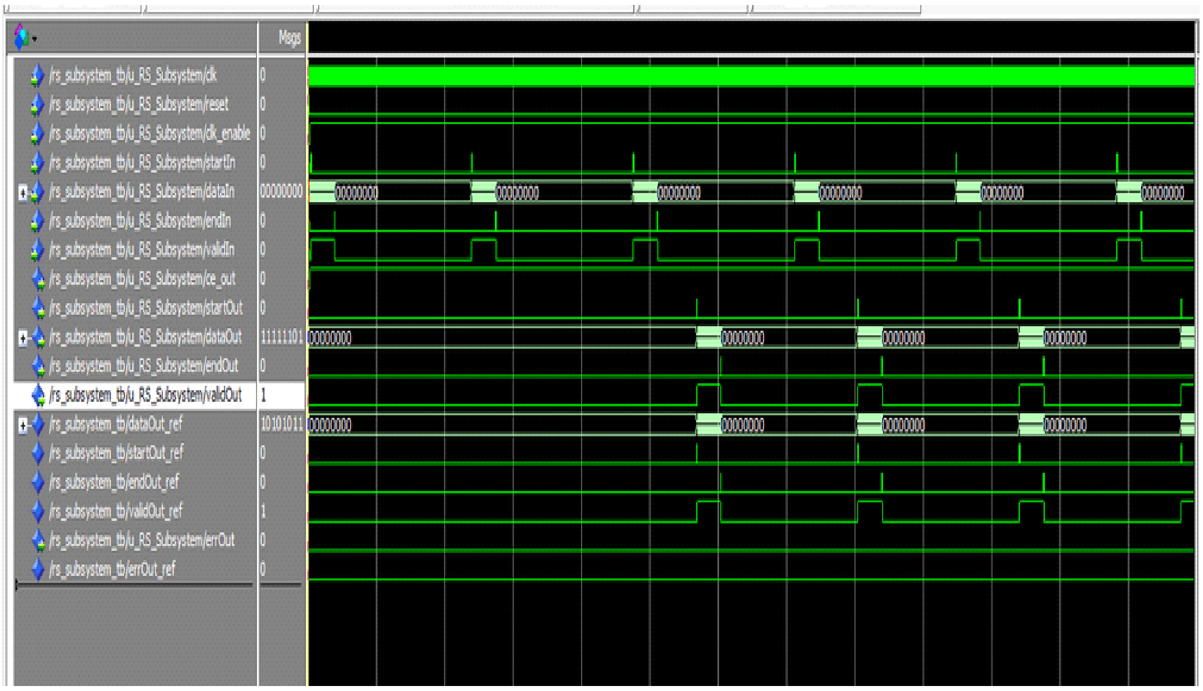

Msgs
/rs_subsystem_tb/u_RS_Subsystem/clk
/rs_subsystem_tb/u_RS_Subsystem/reset
/rs_subsystem_tb/u_RS_Subsystem/clk_enable
/rs_subsystem_tb/u_RS_Subsystem/startIn
/rs_subsystem_tb/u_RS_Subsystem/dataIn
/rs_subsystem_tb/u_RS_Subsystem/endIn
/rs_subsystem_tb/u_RS_Subsystem/validIn
/rs_subsystem_tb/u_RS_Subsystem/ce_out
/rs_subsystem_tb/u_RS_Subsystem/startOut
/rs_subsystem_tb/u_RS_Subsystem/dataOut
/rs_subsystem_tb/u_RS_Subsystem/endOut
/rs_subsystem_tb/u_RS_Subsystem/validOut
/rs_subsystem_tb/dataOut_ref
/rs_subsystem_tb/startOut_ref
/rs_subsystem_tb/endOut_ref
/rs_subsystem_tb/validOut_ref
/rs_subsystem_tb/u_RS_Subsystem/errOut
/rs_subsystem_tb/errOut_ref

Fig. 9 The ModelSim simulation results of the RS encoder/decoder

5. Conclusions

The design and implementation of the communication algorithm is a very complex process, and the hardware implementation is very difficult. In order to meet the fast hardware implementation of communication algorithms, this paper proposed an optimized approach of the RS encoder/decoder using MBD workflow. Through the method of MBD, the complex communication algorithm can automatically generate HDL code, quickly complete the functional validation of FPGA design. The described methodology allows accelerating the design process of communication systems. Compared with the traditional design methods, the MBD method can improve product quality and reduce development time.

References

[1] Nie Y, Ge H, Jing L. Model-Based Design Methodology for Digital Up and Down Conversion of Software Defined Radio[J]. International Journal of Multimedia and Ubiquitous Engineering, 2016, 11(4): 27-36.

[2] Chatterjee S, Kleijn W B. Auditory Model-Based Design and Optimization of Feature Vectors for Automatic Speech Recognition[J]. IEEE Transactions on Audio Speech & Language Processing, 2011, 19(6):1813-1825.

[3] Wang S, Shin K G. Task construction for model-based design of embedded control software[J]. IEEE Transactions on Software Engineering, 2006, 32(4):254-264.

[4] Costabile M F, Fogli D, Mussio P, et al. Visual Interactive Systems for End-User Development: A Model-Based Design Methodology[J]. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 2007, 37(6):1029-1046.

[5] Wicker S B, Bhargava V K. Reed-Solomon Codes and Their Applications[M]. John Wiley & Sons, Inc. 1999.

[6] IEEE 802.16: IEEE Standard for Air Interface for Broadband Wireless Access Systems(Revision of IEEE Std 802.16-2009). IEEE-SA. 8 June 2012.

[7] Berlekamp E. Bit-serial Reed - Solomon encoders[J]. IEEE Transactions on Information Theory, 2003, 28(6):869-874.

[8] Hsu I S, Reed I S, Truong T K, et al. The VLSI Implementation of a Reed—Solomon Encoder Using Berlekamp's Bit-Serial Multiplier Algorithm[J]. IEEE Transactions on Computers, 2006, C-33(10):906-911.

[9] Dash A R, Lenka T R. VLSI implementation of Reed-Solomon encoder algorithm for communication systems[J]. Radioelectronics & Communications Systems, 2013, 56(9):441-447.

[10] Clark, George C. Jr., and J. Bibb Cain, Error-Correction Coding for Digital Communications, New York, Plenum Press, 1981.