MODEL BASED DESIGN AND SDR

Mansour Ahmadian[*], **Zhila (Jila) Nazari**[†],**Nory Nakhaee**[‡],**Zoran Kostic**[⋆]

[*] Sundance Multiprocessing Technology Ltd, Chesham, UK (mansour.a@sundance.com)
[†] School of Engineering and Electronics. University Of Edinburgh, Scotland, UK (z.j.nazari@ed.ac.uk)
[‡] Sundance Digital Signal Processing Inc, Reno, USA (nory.n@sundance.com)
[⋆] The MathWorks, Natick, USA (wireless.sdr@mathworks.com)

Keywords: SDR, Model Based Design, DSP, Embedded software development

Abstract

This paper focuses on the major phases present in the development of an SDR system: design; simulation; code generation and verification. The paper will illustrate, through an example, the use of Model Based Design methodology and tools to integrate the major development phases into one continuous design cycle. Advanced system design concepts including, simulation, code generation, hardware in the loop testing will be presented. Implementation of an FM^3TR (Future Multi-Band, Multi-Waveform, Modular, Tactical Radio) modulator-demodulator will be discussed and presented.

1 Introduction

In a traditional development, usually a system engineer defines the overall system specification and presents it as a design document to software engineers, who will have the task of implementing those ideas into a fully working solution. However, the main problem with this approach is the fact that in most cases the ideas presented by the system engineer via the specification document may widely differ to the implemented software. Even the most detailed and diligently prepared type of documentation may not always guarantee that the design document generated by the system engineers would be fully undestood and accurately understood and interpreted correctly by the implementing software programmers. The new emerging technique to solve this problem is ”model based design” [7], In this technique, system engineers design and simulate the model using a model based design tool and he present the ideas behind the solution as a working model of the system . Some of the sophisticated model based design tools, have advanced features for automatic generation of source code that would mimic the simulated system and hence cutting time to market and reducing development costs.

In recent years model based design methodology has become the preferred method for designing, modelling and simulating complex dynamic systems [9] [5]. The traditional system design methodology, which is shown in Fig.1, is rather time consuming and error prone in nature.

In this simple model, the design and development of a system and its introduction to the market consists of 4 separate steps.

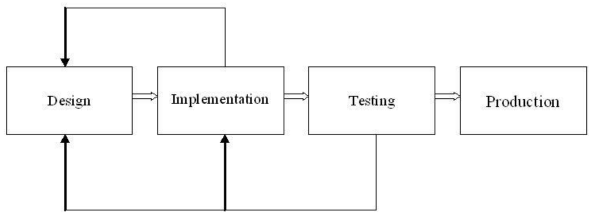


Fig. 1. Simplified presentation of traditional system design methodology.

The first phase involves the definition and design of the objective system. This is the usually carried out by the system designers, who define the system specification and presents it as a document to software designers. This stage can have the highest risk as it is the biggest source of error since different code developers may interpret these definitions differently.

The next step is the implementation part. At this stage the developers will generate suitable software code in a selected language to implement the system. For embedded systems, the preferred programming language is C. If the system definitions are not clear, the programmers would refer to system designer for clarification. Even after these clarifications, it is always possible that the developed system may be different from the one that system designers had in mind.

After code development, the next step is testing. It is very common that the test result will be different from what the programmers or system designer had in their minds or specified. If the tests show that there are mistakes in the system design, then system designer should revise his design and pass it to code developers to implement the revised system. Of course it can also be the case that the implementation is not a true representation of the specification and hence the test results show errors. In this case the programmer has to make sure he has a better understanding of the specification and either correct the algorithm used or simply debug the application. Either of these processes may be repeated several times. To complicate the matter even further one can also consider the case where the test procedures or implementation are wrong and thus giving erroneous results.

The main problem of this approach is the fact that in most

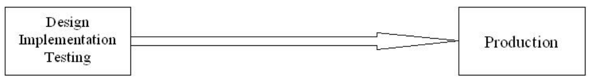


Fig. 2. Model based system design steps.

cases the ideas developed by the system engineer/designers and contained in the specification document and the implemented software, may be different.

After extensive testing and only when the system designers are convinced that the implemented system is operating correctly and as intended production can start. The complexity of this lengthy process makes project management ineffective and very difficult.

2 Model based design

To improve on the above approach and in order to avoid miscomprehension the model-based system design was invented [10] [8]. Model-based design is now an established approach to develop efficient solutions to complex engineering problems [2]. In this method, complicated systems can be created by using mathematical models representing system components and their interactions with their surrounding environment. These models have many applications in the design process, including system simulation, stability analysis, and control algorithm design. By introduction of advanced, automated code-generation technology, another application of these models has become viable. These models can be used as the input to an automatic code generation tool. Advanced, state of the art code generators can produce optimized, embeddable C source codes from these models [1].

Using model based design methodology reduces the no of development stages by combining the design, implementation and test stages in into one process as shown in Fig.2.

The reduction of required steps, compared to the traditional method, and complexity will result in better project management and mitigation of product development risks. The systems designed using this approach reach the market faster and end up costing less than the systems designed using the conventional methods.

3 Model based Design and SDR

Complexity of software defined radio (SDR) applications and the difficulty of testing, debugging and validating of such systems, makes them particularly suitable for development with model based design tools as part of the core design process. To demonstrate the viability of this methodology for design, implementation and testing of a complex system, the model based system design technique was used to implement and test an FM^3TR (Future Multi-Band, Multi-Waveform, Modular, Tactical Radio) modulator-demodulator.

The model based design tool that was used for this work was Simulink [14]. Simulink is a well-known tool for modelling, analysing, and simulating a very wide variety of physical and mathematical systems, including those with non-linear elements. As an extension of Matlab, Simulink adds many features specific to dynamic systems while retaining all of Matlab's general purpose functionality and features. Using Simulink, one models a system graphically, sidestepping much of the nuisance associated with conventional programming.

SMT6050 from Sundance [4] is used for code generation. SMT6050 uses Real Time Workshop [13] from MathWorks Inc. [3] to generate optimized; embeddable code targeting Sundance hardware. Using SMT6050, generating code for Sundance DSP boards is only a click away. Sundance SDR kit is used as the basic hardware platform for testing the developed code. This kit consists of the following parts:

- PCI carrier board (310Q): To hold DSP and DAQ board and build a very high speed data communication channel between DSP and PC.
- DSP board (SMT365G): Texas Instrument TMS320C6416 DSP processor running at 1GHz with 4Mbytes ZBT-RAM. It also contain a Xilinx Virtex II XC2V1000-4 FPGA.
- DAQ module(SMT370): Two 14-bit ADCs (AD6645) sampling up to 105 MHz. Dual 16-bit TxDAC (AD9777) sampling up to 400 MHz (interpolation).

To implement the application first it was designed and tested with Simulink. When the design phase finished and the system behaviors was proved to meet the specifications, the created model was prepared for code generation. To test the generated code, we used "Hardware In the Loop" (HIL) procedure. In HIL technique, the generated code is run on the real hardware. Simulink inputs the test vectors to the hardware under consideration and collects the system output and processes or displays them to validate the system under test.

4 FM^3TR modulator-demodulator

The implementation of a reference FM^3TR (Future Multi-Band, Multi-Waveform, Modular, Tactical Radio) modulator-demodulator [12] was selected as the SDR system to be developed using model based system methodology. The FM^3TR was an international cooperative effort between the United States, Germany, France and the United Kingdom to develop a reconfigurable communication system for ground and airborne applications. FM^3TR will allow communications systems to be more affordable and be able to potentially bridge the interoperability gap between nations. The primary goal in this information exchange is improvement in the state of the art

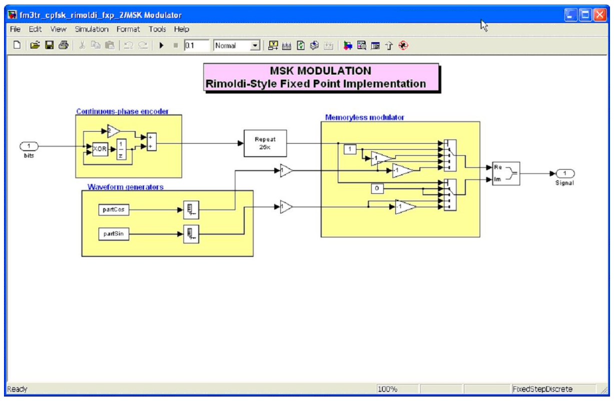


Fig. 3. FM^3TR modulator implemented in Simulink. (courtesy of The MathWorks)

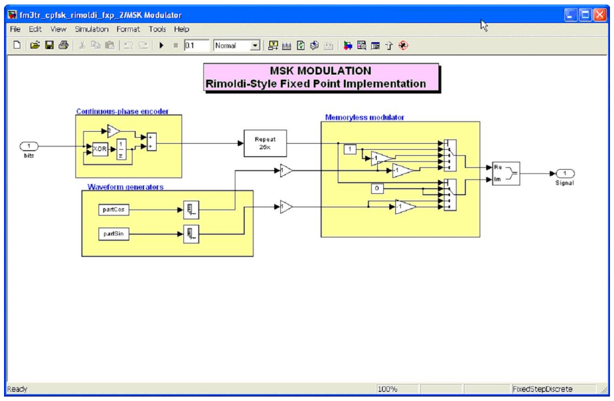


Fig. 4. FM^3TR demodulator implemented in Simulink. (courtesy of The MathWorks)

of multifunctional radios, primarily focused on hardware and software architectures, digital signal processing, man-machine interfaces, RF and digital technology and interconnecting networks.

5 Building modulator-demodulator in Simulink

Modulator and demodulator were built and simulated in Simulink. The modulator is show in Fig.3. It was built using standard Simulink blocks.

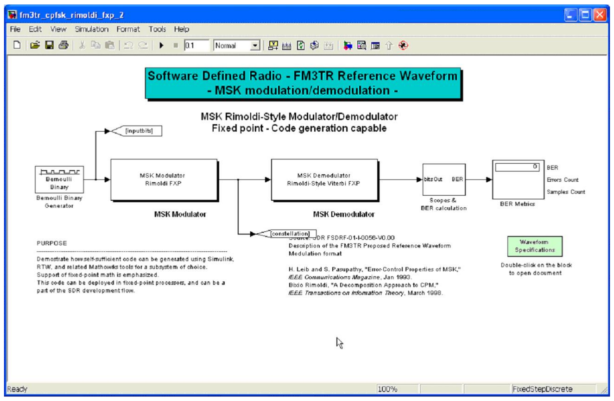
Modulator consists of three sub-systems:

1. Continues-phase encoder.
2. Waveform generators.
3. Memory-less modulator.

After building the modulator model in Simulink, it was simulated and its functionality was tested. The demodulator was also built using standard Simulink blocks. It is shown in Fig.4.

The demodulator consists of two sub-systems:

1. Optimum state sequence receiver.


fm3tr_cpfsk_rimoldi_fxp_2
Software Defined Radio - FM3TR Reference Waveform
- MSK modulation/demodulation -
MSK Rimoldi-Style Modulator/Demodulator
Fixed point - Code generation capable
[inputbits]
Bernoulli Binary
Bernoulli Binary Generator
MSK Modulator Rimoldi FXP
MSK Modulator
MSK Demodulator Rimoldi-Style Viterbi FXP
MSK Demodulator
bitsOut
BER
Scopes & BER calculation
BER
Errors Count
Samples Count
BER Metrics
[constellation]
Waveform Specifications
Double-click on the block to open document
Ready
100%
FixedStepDiscrete

PURPOSE

Demostrate how self-sufficient code can be generated using Simulink, RTW, and related Mathowks tools for a subsystem of choice.
Support of fixed-point math is emphasized.
This code can be deployed in fixed-point processors, and can be a part of the SDR development flow.

Source: SDR FSDRF-01-I-0056-V0.00
Description of the FM3TR Proposed Reference Waveform Modulation format

H. Leib and S. Pasupathy, "Error-Control Properties of MSK," *IEEE Communications Magazine*, Jan 1993.
Bixio Rimoldi, "A Decomposition Approach to CPM," *IEEE Transactions on Information Theory*, March 1998.

Fig. 5. Test bench to test the compatibility between developed modulator and demodulator. (courtesy of The MathWorks)

2. Differential decoder.

Demodulator was also simulated and tested in Simulink.

To test the compatibility between the modulator and the demodulator, a test bench was developed using Simulink as shown in Fig.5.

To make sure that the modulator and the demodulator are compatible with each other, the output of modulator was fed to the input of demodulator. By comparing the input signal to modulator and the signal out of the demodulator, the compatibility of modulator-demodulator was verified. The modulator was fed by a Bernoulli-distributed random binary number generator. Simulation showed that the designed modulator-demodulator were matched and the error between modulator input and demodulator output were zero.

6 Code generation

SMT6050 was used for code generation. SMT6050 generates not only all of the necessary C source code but also some other valuable files such as:

- The linker command file that instructs the linker how to place objects codes into DSP memory. The generated linker command file is compatible with targeted hardware and no user intervention is needed for this purpose.
- Make file to compile and link the generated source codes. The compiler and linker switches are set according to target hardware and target DSP without any need for user knowledge of underlying hardware architecture and how it would affect the developed software.
- Batch file to set-up required operating system environmental variable and commands for running make utility.

After code generation, SMT6050 runs the generated batch file to build the output file. It also creates required libraries for Simulink block-sets, which are used in the designed model. This created library is compatible with the target hardware and DSP. As one needs to test the generated code and as already

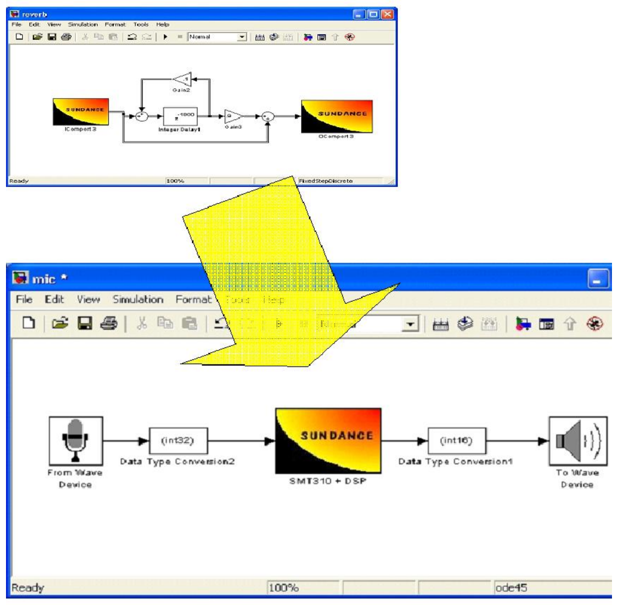


Fig. 6. Hardware in the loop demo that shipped with SMT6050.

there was a test bench generated in Simulink, it was logical to use that for testing the application code.

7 Hardware in the loop simulation and testing

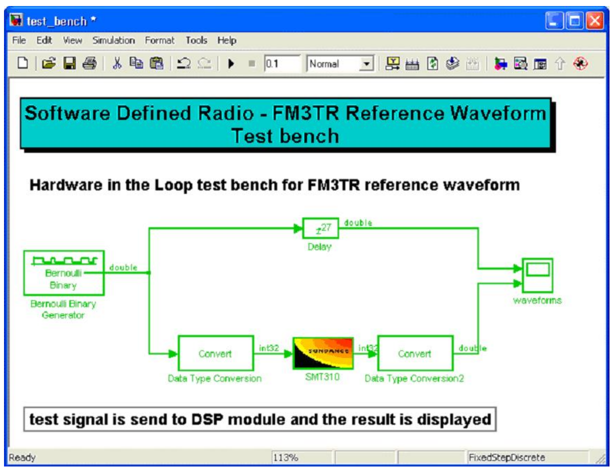
In normal simulation all parts of simulation takes place on the host computer (in case of Simulink, all of the simulation takes place on the PC). In ”Hardware In the Loop” (HIL) simulation, part of the simulation takes place on the host and another part is executed on the DSP board [6]. SMT6050 includes a demo that shows how to build and test a HIL system. In that demo, data is captured from a microphone on PC and is sent to DSP for processing. DSP processor adds reverb to the sound and returns it back to the PC. The processed sound is then played on PC speaker (Fig.6).

To test the generated code for FM^3TR modulator-demodulator, the Simulink model shown in Fig.5 was broken into two parts:

7.1 Test bench

Test bench would be run on PC and would feed test vectors to the DSP and then receive the result back from the DSP. By comparing the received signal to the feed signal, the validity of generated code could be verified as shown in Fig.7.

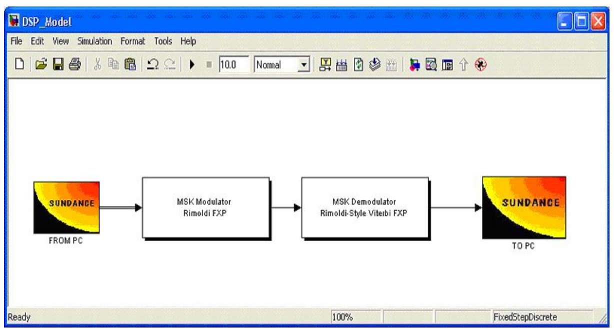
The test bench model shown in Fig.5 was used to construct hardware in the loop test bench. Modulator-demodulator were removed from the model and SMT310 block was placed instead of them. This block would then send data to DSP and then receive data from DSP after processing. Some blocks were added to convert data types that would be transmitted to/received from the DSP. Since modulator-demodulator generate 27 time step delays, signal generated by Bernoulli binary generator was delayed by 27 so both the original and processed


Software Defined Radio - FM3TR Reference Waveform Test bench

Hardware in the Loop test bench for FM3TR reference waveform

test signal is send to DSP module and the result is displayed

Fig. 7. Test bench model prepared for hardware in the loop simulation and testing.



DSP_Model
SUNDANCE
FROM PC
MSK Modulator Rimoldi FXP
MSK Demodulator Rimoldi-Style Viterbi FXP
TO PC
Ready
100%
FixedStepDiscrete

Fig. 8. Modulator-demodulator block prepared for hardware in the loop code generation.

signals would be in phase with each other. The generated and processed signals were shown on a scope.

7.2 Modulator-demodulator

This is the part that went through code generation and runs on the DSP. It consists of modulator-demodulator as shown in Fig.8. The Sundance blocks ”From PC” and ”To PC” were added to the model so the input of modulator came from PC and output of demodulator was sent to the PC.

7.3 Synchronization

Since the two different part of this simulation is running on different processor (test bench model on PC and modulator-demodulator on DSP), they should be synchronized with each other [11]. To solve this complex problem, SMT6050 uses data flow synchronization technique. In this technique, the programs on PC and DSP are running freely with their own clock. Test bench sends enough data to DSP so it can start its processing and wait for results to come back from DSP. Modulator-demodulator, which are running on DSP waits until enough data become ready in its input. At this point, DSP starts to process data as fast as it can and sends the resulting data back

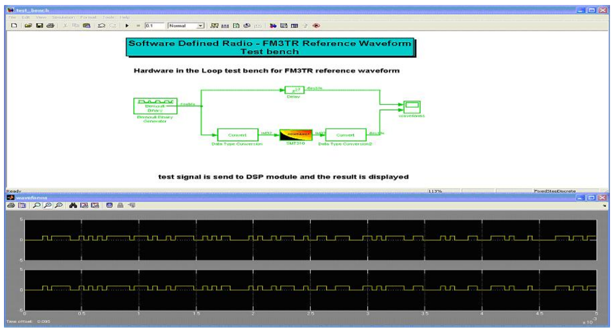

Software Defined Radio - FM3TR Reference Waveform Test bench
Hardware in the Loop test bench for FM3TR reference waveform
test signal is send to DSP module and the result is displayed

Fig. 9. Testing of modulator-demodulator using HIL technique.

to PC. SMT6050 generates all of the required source code for DSP to implement data flow synchronization. It also provide SMT310 block for Simulink so that data communication with DSP and synchronization become as simple as possible.

7.4 Testing

To test the validity of the design, code for modulator-demodulator is created and downloaded into DSP board. By running test bench model and comparing the DSP input and output signals, the validity of modulator and demodulator were verified as shown in Fig.9.

8 Conclusion

Model based system design is introduced and the capability of this methodology is investigated. An FM3TR reference waveform modulator-demodulator was developed using model based methodology and tested. SMT6050 was used to generate code for hardware in the loop simulation and testing. The generated code was tested using hardware in the loop technology. The result of this investigation shows that the application could be easily modeled and then developed without the problems associated with traditional design schemes.

9 Biography

Mansour Ahmadian is the senior software engineer in Sundance. His current focus and interest is on new development to use model based design for complete embedded (hardware/software) design. He worked as software developer and recently as software project manager in different organizations.

He received a B.Sc. in Electronic Engineering in 1989, and an M.Sc. in Biomedical Engineering in 1992. He was awarded PhD by the University of Edinburgh for his work in the field of medical signal processing in 2002. Mansour published more than 20 papers in national and international conferences and journals.

In 2001, Mansour won the "Business plan competition" in the university of Edinburgh. Two years later, he was awarded the prestiges SMART:SCOTLAND for his innovative technique for image generation from phased array systems with ultrasound, sonar and radar.

Mansour is a member of IEE.

Zhila (Jila) Nazari is working toward her PhD in the University Of Edinburgh. Her interest is digital signal processing and medical image processing.

She received a BSc. and an MSc in Computer software engineering. She developed several commercially available software applications.

Nory Nakhaee received a B.Tech degree in Automotive Engineering and design and a Masters degree in Information Technology from Loghborough University of Technology in UK. Nory received a Ph.D. degree from Nottingham Trent University in UK for a research titled "Automatic parallelising complier for MIMD/DF machines". From 1982, for five years Nory worked at 3L Limited in Scotland as software engineer, test engineer, commercial manager and then as a company director. Subsequently Nory founded Alpha Data Parallel Systems Ltd in Scotland and served as its Managing Director. In 1999 Nory moved to US and founded Sundance DSP Inc and since then he has been working for that company as its C.E.O.

Zoran Kostic obtained a PhD. degree in Electrical Engineering from University of Rochester and Dipl. Ing. Degree from University of Novi Sad in Yugoslavia. He has worked for Bell Labs, AT&T Research, Thomson Research, The MathWorks and taught at Columbia, UCLA, SUNY and GMU. His expertise is in the field of wireless communications research, design and SW and HW implementation across areas of systems, networks and components.

Acknowledgements

The authors would like acknowledge the help that they received during this work from MathWorks Inc. specially Amon Gai. Authors also would like to acknowledged helps from their colleagues in Sundance, specially Flemming Christensen, Stephen Malchi and Sebastien Maury.

References

[1] Mansour Ahmadian, Nory Nakhaee, and Andrew Nesterov. Rapid Application Development (RAD) and code optimization technique. *Global Signal Processing Conference (GSPx)*, 2004.

[2] M. Baleani, A. Ferrari, L. Mangeruca, A.L. Sangiovanni-Vincentelli, U. Freund, E. Schlenker, and H.J.Wolff. Correct-by-construction transformations across design environments for model-based embedded software development. *Proceedings of Design, Automation and Test in Europe*, 2:1044 – 1049, 2005.

[3] http://www.mathworks.com.

[4] http://www.sundance.com.

[5] J. Langenwalter. Embedded automotive system development process - steer-by-wire system. *Proceedings of Design, Automation and Test in Europe*, 1:538 – 539, 2005.

[6] Jim Ledin and Mike Dickens. Cracking the code automatically. *Electronic design europe*, pages 19 – 23, June 2005.

[7] J. Loyall, Jianming Ye R. Shapiro, S. Neema, N. Mahadevan, S. Abdelwahed, M. Koets, and D. Varner. A case study in applying qos adaptation and model-based design to the design-time optimization of signal analyzer applications. *IEEE Military Communications Conference (MILCOM)*, November 2004.

[8] Arun Mulpur. Faster and better embedded signal processing systems:. *Global Signal Processing Conference*, 2004.

[9] S. Neema, T. Bapty, J. Scott, and B. Eames. Signal processing platform: a tool chain for designing high performance signal processing applications. *Proceedings of SoutheastCon*, pages 302–307, April 2005.

[10] T. Runolfsson. Optimal design of uncertain complex dynamical systems. *43rd IEEE Conference on Decision and Control*, 2:2231 – 2236, December 2004.

[11] N. Scaife and P. Caspi. Integrating model-based design and preemptive scheduling in mixed time- and event-triggered systems. *Proceedings of 16th Euromicro Conference*, June 2004.

[12] Richard N. Smith. Description of the fm^3tr proposed reference waveform. Technical report, Software Defined Radio Forum, August 2001.

[13] The MathWorks, Inc. *Real-Time Workshop user's manual* , 2002.

[14] The MathWorks, Inc. *Simulink user's manual (version 6)*, 2005.