SDR-TARGETED DESIGN FLOW: FROM EXECUTABLE SPECIFICATION TO SIGNAL PROCESSING SUBSYSTEM CODE GENERATION AND SCA-FOCUSED TOOL INTEGRATION

Zoran Kostić (The MathWorks, Natick, MA, USA; wireless.sdr@mathworks.com)
Alex Rodriguez (The MathWorks, Natick, MA, USA; wireless.sdr@mathworks.com)

ABSTRACT

SDR-based radios rely on embedded software for code portability, reuse, and upgradeability. This paper explores a robust design process for SDR and embedded-code generation, for different applications and for diverse hardware and software platforms. The paper elaborates on the methodology that is based on the executable specification captured in mixed graphical/textual environment. This design flow supports: 1) System architecture exploration, 2) Communications performance evaluation, 3) Partitioning of a complex communication system into subsystems/components; 4) Automated portable embedded code generation suitable for integration with SCA components-based tools.

We present the design flow through an interactive example of a communication system which contains complex signal processing functionality, several levels of subsystems, multi-sampling-rates and which is of interest to SDR community. We show code structure and illustrate flexibility in choosing automated code optimization strategies targeted at execution speed, memory size or other features. The ability of code to be integrated with customized schedulers is illustrated. We discuss features of the code structure that facilitate integration of the code into SCA component-based tools.

1. INTRODUCTION

Industry and government have experienced an explosion in the number of communication systems. Demand for interconnectivity, portability, reuse and multi-standard support is challenging the current paradigm of communication radio architecture and design. Software Defined Radio (SDR) has emerged as a promising technology to solve these issues, and with support of U.S. and international organizations, it is poised to become the next breakthrough in wireless communications. SDR is a major paradigm shift in the way engineers think of design and deployment of communications radios. Software defined radio (SDR) paradigm implies the ability to develop systems and software once, run anywhere, and modify at the highest possible conceptual level. In practice, SDR relies on flexible re-targetable embedded software.

Recognizing that traditional text-based specifications are incompatible with SDR hardware portability, The Joint Tactical Radio System (JTRS) Joint Program Office (JPO) has provided a series of guidelines in the Software Communications Architecture (SCA) recommendations. The guidelines suggest that the implementation of executable specifications be based on an Implementation Independent Model (IIM) and an Implementation Specific Model (ISM). IIM and ISM have originally been proposed by the Object Management Group (OMG). The SCA interposes a core framework (CF) designed to provide interoperability between waveforms (such as GSM, 802.11 air-interface standards) and programmable radio hardware. The goal is to design waveforms in form of software applications and be able to download them to any radio set that supports the SCA specification.

The concepts of Model-Based Design and underlying executable specifications are means of enabling the realization of IIM and ISM paradigms. Model-Based Design ensures design portability, code reuse, multi-standard support, interconnectivity among applications and platforms, and flexible, efficient and streamlined design and deployment of SDR radios.

The paper shows how the concepts of executable specification, IIM and ISM, originally proposed in the context of the software architecture, have been extended to the design, implementation and test of the Signal Processing Subsystems. The paper discusses the rationale, requirements and benefits of this design methodology. It elaborates on the process of waveform development, automatic and portable code generation for a variety of hardware platforms, and closed loop development process. We present an example of a design flow for implementing a modulation/demodulation portion of the FM3TR reference waveform.

1. TRADITIONAL SDR DESIGN

Every major increase in communications capability has demanded enormous exertions on the part of hardware and software developers, and SDR is no exception. Leading defense contractors, electronics components suppliers, systems integrators and others are hard at work developing the next generation of SDRs which promise to transform wireless communications by allowing all types of wireless devices to interoperate with each other and to add new features quickly with software downloads.

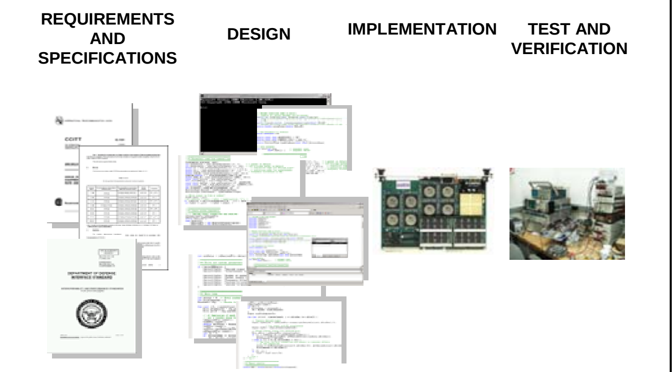


Figure 2. Traditional Design Flow

The challenge gets order of magnitude harder when there is a need to support the ability to communicate with virtually any imaginable waveform using software programs running on flexible hardware. In today's military and commercial wireless platforms, the hardware has been optimized for the waveform and, in particular, the RF section is optimized for the narrow frequency band. Moving from specific to generic involves compromises that can have a negative impact on real-time performance, power consumption, and size. Until now, SDR developers have largely used traditional design methods. Specifications are conceptualized by systems architects as text documents. These documents are used to guide the work of project teams who specialize in areas such as signal processing, RF, VLSI, testing. Project teams specify hardware, design circuits, write software, run simulations, perform tests, and generate mountains of data. Obtained information is delivered back to the systems architect who views the raw information and matches it up against the specifications.

A key problem with this approach is that problems are not detected until all of the different modules can be tested together at the prototype stage. If the prototype results do not match the specification, engineers need to go back and take a close look to determine whether the requirements were valid, if the requirements were properly coded, whether the simulation models were correct, if the interface was correct, or if the problem was with the target HW platform. This weakness can increase development costs because the cost of fixing a problem increases by an order of magnitude as the design progresses through successive stages.

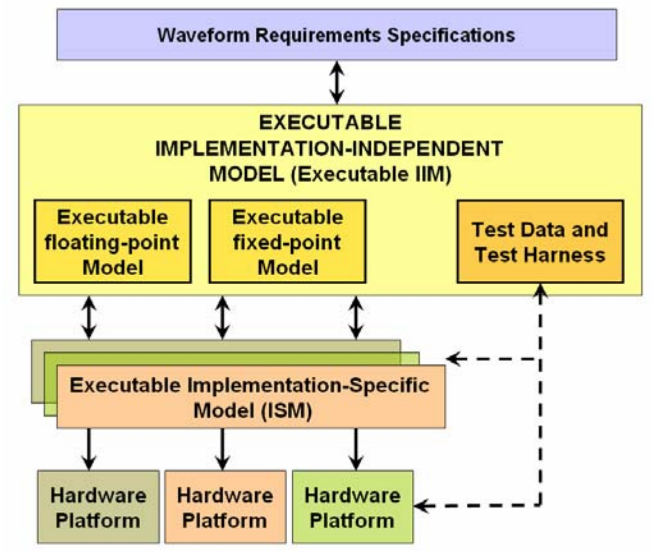

Waveform Requirements Specifications
EXECUTABLE IMPLEMENTATION-INDEPENDENT MODEL (Executable IIM)
Executable floating-point Model
Executable fixed-point Model
Test Data and Test Harness
Executable Implementation-Specific Model (ISM)
Hardware Platform
Hardware Platform
Hardware Platform

Figure 1. Model Based Design

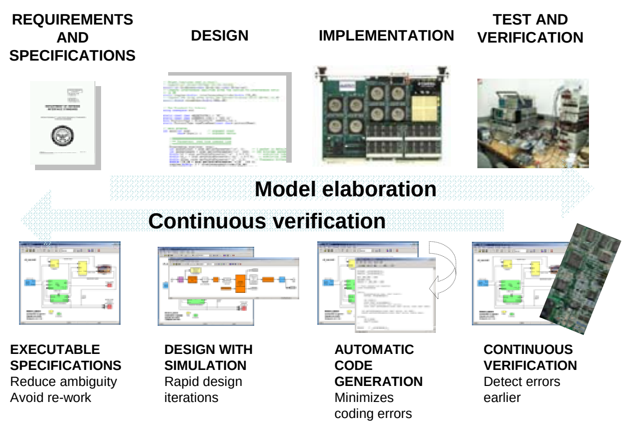


Figure 3. Model Based Design

All of these challenges, however, pale in comparison to the need to develop for multiple targets. Digital signal processing is at the heart of SDR and this function can be accomplished using a range of hardware solutions including general-purpose processors (GPPs), digital signal processors (DSPs), and field programmable gate arrays (FPGAs). Every major development program must consider a wide range of different hardware options and hardware selection is often not finalized until far into the development process. Even after the hardware choice is made, developers must be prepared for hardware upgrades and derivative products that may take a completely different direction, such as a switch from DSPs to FPGAs. The biggest problem with traditional development methods is that they are specific to particular hardware architecture, therefore making it necessary to restart the development process.

Recognizing that traditional design methodologies have slowed JTRS development significantly, the JTRS Joint Program Office (JPO) has provided guidelines for a new design methodology in which waveform specifications are defined in the form of an **Implementation Independent Model (IIM)**, which is independent of the hardware specification, and an **Implementations Specific Model (ISM)**, which is specific to the hardware implementation. The intent of guidelines is to enable porting of waveforms on multitude of HW/SW platforms.

2. IMPLEMENTATION INDEPENDENT AND IMPLEMENTATION SPECIFIC MODEL

An Implementation Independent Model includes information to define, characterize and validate how the waveform behaves, which can be verified and traced to the waveform requirements documents. The signal flow, control flow, and networking aspects of the waveform are defined using information such as waveform subsystem boundaries, subsystem jitter, latency and timing requirements, subsystem processing requirements, and signal port sample times. An ***executable IIM*** provides a test bench to verify waveform functional blocks or systems against system requirements and validate their performance (Figure 1).

An Implementation Specific Model, on the other hand, reflects the details of an intended implementation as ported to a specific radio set architecture. It contains information about the allocations of process components to various processor resources, enabling a detailed understanding of the implementation. It includes a model of execution timing on the target processors, latencies, memory, and queue sizes, allowing the developer and subsequent porting efforts to understand the impact of changes in resource capability on the system level behavior, such as throughput, jitter, latency, memory consumption, power, and real-time performance.

3. MODEL-BASED DESIGN ENABLES IIM AND ISM IMPLEMENTATION

Model-Based Design helps engineers and designers overcome the difficulties of traditional development approaches. Comprehensive, system-level mathematical models form the basis of Model-Based Design. Models serve as executable specifications, replacing ambiguous text documents. Model-Based Design eliminates ambiguity and allows communication throughout an organization and to customers, contractors, and suppliers. Using Model-Based Design, algorithm developers, RF designers, software, and hardware engineers can cooperate to make tradeoffs and evaluate solutions, thus enhancing performance and reducing costs, as illustrated in Figure 3. Designers can simulate and iterate as many times as necessary to refine the model to meet the constraints of the target environment, and to validate the system behavior against the requirements. Once the design is validated, designers can automatically generate code from the model, eliminating the need for hand coding and the errors that manual coding introduces. Engineers can ensure quality throughout the development process by integrating tests into models. This continuous verification and simulation helps identify errors early, when they are easier and less expensive to fix.

Model-Based Design captures both conceptually high-level model descriptions of Implementation Independent Models, as well as fine nuances of the Implementation Specific Models.

The executable specification of the waveform is initially defined at a high conceptual level using graphical descriptions with pre-built primitives/blocks that can describe both simple and complex functions and algorithms.

Table 1. Generated Code Fragment for Demodulator

```
/*
 * File: Demodulator.c
 */

#include "Demodulator.h"
#include "Demodulator_private.h"

/* Block signals (auto storage) */
BlockIO_Demodulator Demodulator_B;

/* Block states (auto storage) */
D_Work_Demodulator Demodulator_DWork;

/* External inputs (root inport signals with auto storage) */
ExternalInputs_Demodulator Demodulator_U;

/* External output (root outports fed by signals with auto storage) */
ExternalOutputs_Demodulator Demodulator_Y;

/* This function implements a deterministic rate-monotonic multitasking scheduler for a system with 2 rates.  The function is called by the generated step function
 */
static void rate_monotonic_scheduler(void)

/* Model step function for TID0 */
void Demodulator_step0(void)
    rate_monotonic_scheduler();

/* Model step function for TID1 */
void Demodulator_step1(void)
/* Sample time: [0.00004s, 0.0s] */

/* Model initialize function */
void Demodulator_initialize(boolean_T firstTime)


/* Model terminate function */
void Demodulator_terminate(void)
```

As per needs of the model or the designer, subcomponents of the model can be incorporated into the block-based Model-Based Design model in programming languages such as C,C++, Fortran, MATLAB® or HDL. Custom and/or legacy code is embeddable using interfacing/API rules. The specification can then be executed to determine the performance that would be delivered by incorporated components. By performing detailed simulations of the system behavior under different conditions, parameter values, and inputs, engineers can quickly identify, isolate, and fix system design problems. Designers can make changes by adding, subtracting, or moving blocks or changing parameters and immediately assessing the impact of the changes. Designers can evaluate the effects of quantization from floating-point, typically used during the early stages of design, to fixed-point representations, typically used in hardware implementation.

In a text-based approach, implementation of hardware or software is typically done by manual recoding. This process is time-consuming and error-prone. Model-Based Design incorporates both the IIM and the ISM. The IIM consists of hardware-independent functional blocks while the ISM consists of blocks that have been optimized for a particular hardware implementation. The model increases in detail as it moves from the specifications phase, through design, implementation and into the overall system validation and testing but remains a single, unambiguous representation of the system throughout the development process. While a model maintains the intellectual property, it can be used to automatically generate code for a wide range of hardware platforms, including C code for general purpose processors, C and assembly for digital signal processors, and HDL for FPGAs. Automatic code generation provides a coding standard for the generated code, because the same constructs are used with every build. This approach eliminates hand-coding errors and limits potential error sources between the simulation code and the embedded code. Since the code is directly traceable to the simulation, errors must either be in the interface or in the execution under real-time constraints. Because the model is developed independently of an embedded hardware target, it can easily be retargeted to different platforms and re-used in future systems.

Table 2. Code Generation Files for Demodulator

```
Demodulator.c
Demodulator_data.c
ert_main.c
Demodulator.h
Demodulator_private.h
Demodulator_types.h
rtwtypes.h
```

Throughout the process, quality is ensured by integrating tests into the model. The model has a test suite of cases with baseline results. This continuous verification and validation helps identify errors early, when they are easier and less expensive to fix. The system-level model developed by the system architect can be used later in the design process to incorporate realistic data generated by simulation or testing to validate the design.

4. FM3TR REFERENCE WAVEFORM

Future Multi-Band, Multi-Waveform, Modular, Tactical Radio (FM3TR) is a reference waveform chosen by the SDR Forum for as a test and demonstration vehicle [2].The waveform contains modulation, channel coding, speech coding and data framing as representatives of important communications functionalities. In this paper, we focus on the lowest physical layer– modulation and demodulation. We demonstrate Model-Based Design and show the following: high-level conceptual design capture, simulation and evaluation, elaboration and refinement in the domain of fixed-point analysis, code generation for different software/hardware platforms and code optimization approaches. We illustrate the organization of the generated code for purposes of integration with SCA component paradigm.

A. MODEL OF FM3TR MOD/DEMOD

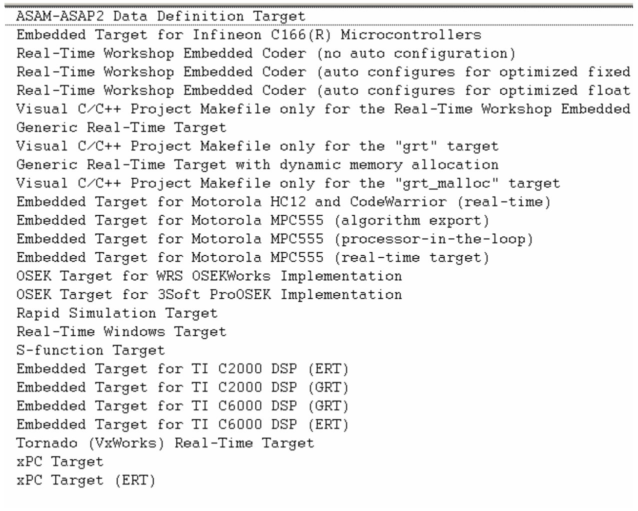
An illustration of the FM3TR modulation/demodulation is shown in Figure 4. The model implements the MLSE-based MSK receiver as proposed by Rimoldi [3, 4]. The model is capable of simulations with arbitrary fixed point specifications, for evaluating DSP or VLSI SW or hardware architectures. This is a representative of executable specifications and IIM, in that the model fully captures functional aspects of the design.

B. CODE GENERATION AND OPTIMIZATION

From the IIM model of the previous section, a large number of ISM models can be built in automatic fashion, by means of generating code – either generic C or one of many targeted processors. A selection of target SW/HW platforms is illustrated in Table 3. Generated code can run as a part of the IIM, or can be run as a part of the external SW/HW platform. An arbitrary subsystem of the original IIM model can be selected for transformation into the ISM model. For instance, one can select the whole demodulator for code generation, or only a part of the demodulator running at the symbol rate. Whatever segment is chosen for generation is fully controllable by the ISM-level model, and any modifications or errors can be traced back, evaluated and corrected for at the conceptually high level IIM model.

Automatic code generation is controllable through code optimization. Exemplary optimization options include storage reuse, parameter inlining, loop unrolling, initialization options, C or C++ interfacing, memory mapping specification, custom library calls and other.

Table 3. Code Targets



```
ASAM-ASAP2 Data Definition Target
Embedded Target for Infineon C166(R) Microcontrollers
Real-Time Workshop Embedded Coder (no auto configuration)
Real-Time Workshop Embedded Coder (auto configures for optimized fixed
Real-Time Workshop Embedded Coder (auto configures for optimized float
Visual C/C++ Project Makefile only for the Real-Time Workshop Embedded
Generic Real-Time Target
Visual C/C++ Project Makefile only for the "grt" target
Generic Real-Time Target with dynamic memory allocation
Visual C/C++ Project Makefile only for the "grt_malloc" target
Embedded Target for Motorola HC12 and CodeWarrior (real-time)
Embedded Target for Motorola MPC555 (algorithm export)
Embedded Target for Motorola MPC555 (processor-in-the-loop)
Embedded Target for Motorola MPC555 (real-time target)
OSEK Target for WRS OSEKWorks Implementation
OSEK Target for 3Soft ProOSEK Implementation
Rapid Simulation Target
Real-Time Windows Target
S-function Target
Embedded Target for TI C2000 DSP (ERT)
Embedded Target for TI C2000 DSP (GRT)
Embedded Target for TI C6000 DSP (GRT)
Embedded Target for TI C6000 DSP (ERT)
Tornado (VxWorks) Real-Time Target
xPC Target
xPC Target (ERT)
```

As an example, choosing the demodulator for generation of embeddable generic C-code would generate functions listed in Table 2. Fixed-point IIM model is shown in Figure 5. Fragments of generated code are shown in Table 1.

C. ADAPTING CODE FOR INTEGRATION INTO SCA COMPONENTS

Code generated automatically from Simulink Model-Based Design Platform (RTW or Embedded Coder products), in a default version, contains the following basic functions: a) initialization, b) update/output – which contain the main algorithmic functionality, and c) terminate. On the other hand, SCA recommendations require the existence of the following functions: a) initialize, b) start, c) stop, d) release, e) afterConfigure, f) beforeQuery, g) run, and h) test. To properly support SCA-requirements, proper interface functionality needs to be built. Component-based specifications in SCA space are currently lacking precise timing management functionality description, which also introduces a need to accommodate needs of timing-critical communications operations. Customized templates and state machines that can parse SCA commands and control timing-critical functions can be means of creating and managing the interface software. State machine can parse SCA calls into function calls recognizable by blocks carrying communications functionality, as well as provide control to timing-critical physical layer functions. In the presentation, we discuss means of utilizing a tool called Stateflow for purposes of semi-automatic generation of interface code.

CONCLUSION

Model-Based Design is a key component in the development approach necessary for the realization of SDRs. It supports the process of automatic code-generation and code portability across hardware, software, and SCA core-framework platforms. Model-Based Design streamlines development through the creation of executable specifications and IIM and ISM models, maintaining traceability to the original waveform specifications and ensuring continuous verification throughout the development process. SDR design and deployment using Model-Based Design is significantly simpler and more robust than traditional design processes. The outcome is the improvements in cost, performance and reliability of SDR systems.

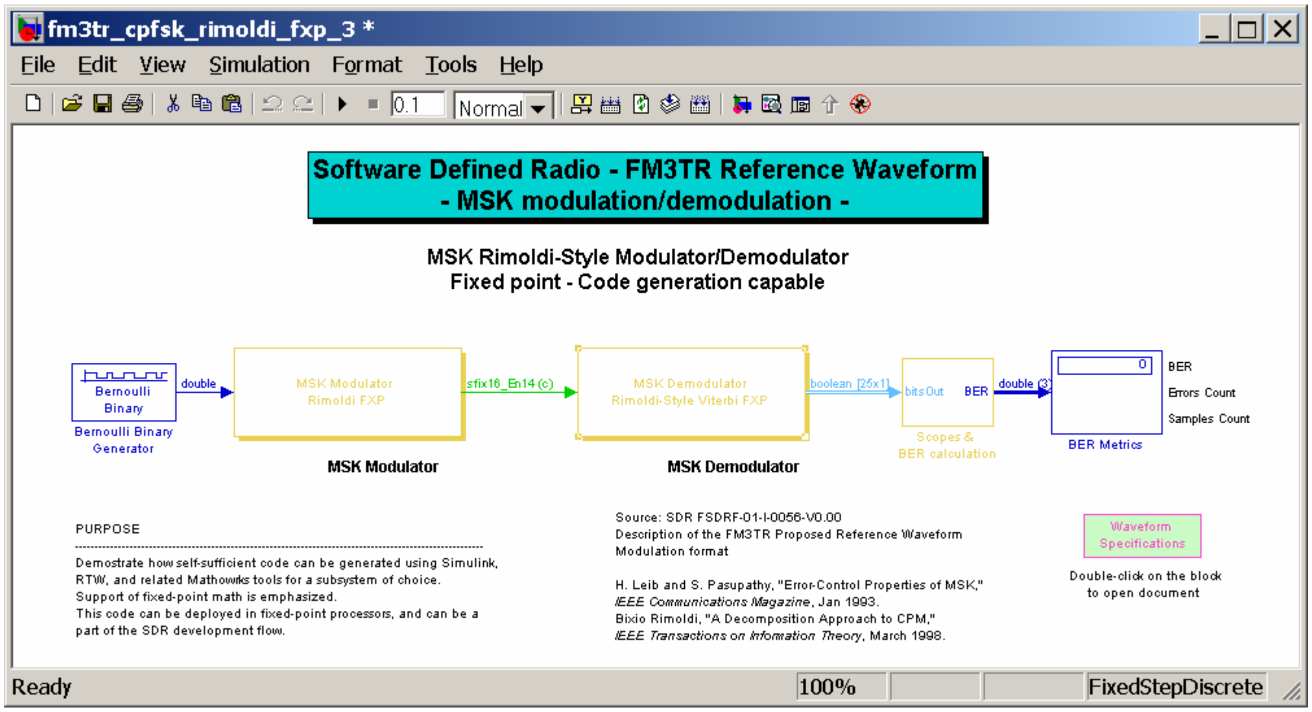
REFERENCES

[1] Acquisition Guidance to the JTRS Software Communication Architecture (SCA) Specification JTRS-5000 SP V3.0 30 June 2004

[2] R. N. Smith, “Description of the FM3TR Proposed Reference Waveform,” Document number SDRF-01-I-0056-V0.00, August 30, 2001.

[3] Bixio Rimoldi, "A Decomposition Approach to CPM," IEEE Trans. on Information Theory, March 1998.

[4] H. Leib and S. Pasupathy, "Error-Control Properties of MSK," IEEE Communications Magazine, Jan 1993.



fm3tr_cpfsk_rimoldi_fxp_3 *
File Edit View Simulation Format Tools Help
0.1
Normal
Software Defined Radio - FM3TR Reference Waveform
- MSK modulation/demodulation -
MSK Rimoldi-Style Modulator/Demodulator
Fixed point - Code generation capable
Bernoulli Binary
Bernoulli Binary Generator
double
MSK Modulator Rimoldi FXP
sfix16_En14 (c)
MSK Demodulator Rimoldi-Style Viterbi FXP
boolean [25x1]
bitsOut
BER
Scopes & BER calculation
double (3)
0
BER
Errors Count
Samples Count
BER Metrics
MSK Modulator
MSK Demodulator
Waveform Specifications
Double-click on the block to open document
Ready
100%
FixedStepDiscrete

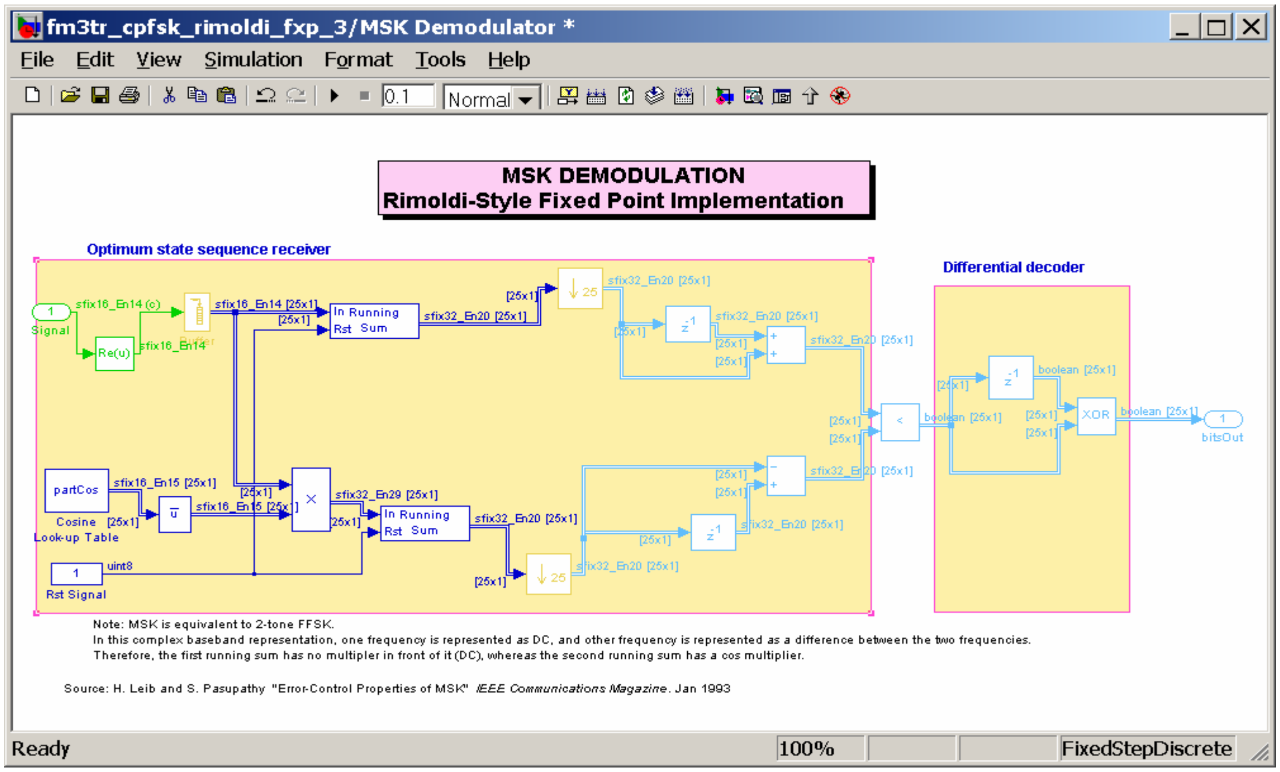
PURPOSE

Demostrate how self-sufficient code can be generated using Simulink, RTW, and related Mathowrks 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.

Figure 4. Model of the Modulator and Demodulator



Note: MSK is equivalent to 2-tone FFSK.
In this complex baseband representation, one frequency is represented as DC, and other frequency is represented as a difference between the two frequencies.
Therefore, the first running sum has no multipler in front of it (DC), whereas the second running sum has a cos multiplier.

Source: H. Leib and S. Pasupathy "Error-Control Properties of MSK" *IEEE Communications Magazine*. Jan 1993

Figure 5. Fixed point Model