

# Communication System Design for Software Defined Radio

**Mike McLernon**

Manager, Communications Development

The MathWorks

June 15, 2006



# Outline

## Overview of Software Defined Radio (SDR)

- Working Definition of SDR
- MathWorks Activities in SDR
- Design Flows for SDR Development

## MathWorks Tools for SDR Designs

- Demo: SDR Reference Waveform – FM3TR
  - Simulations
  - Fixed-Point System Design
  - Automatic Code Generation
- Example Design Flows for Target Platforms

## Conclusion

# A Working Definition of SDR

- SDR is a collection of hardware and software technologies that enable reconfigurable system architectures for wireless networks and user terminals.
- Embedded, portable, reusable software
  - Across diverse software and hardware platforms
  - Across teams, projects, and in time
  - For multistandard support
  - For reduction of development cost and time
- Defense industry driving the technology via JTRS

# The MathWorks Activities in SDR

- Active member of the SDR Forum since 2002
- Participation in work groups:
  - Design Process and Tools
  - Hardware Abstraction Layer
  - Commercial Technology
- Delivered workshops on code portability and embeddable transceiver code generation

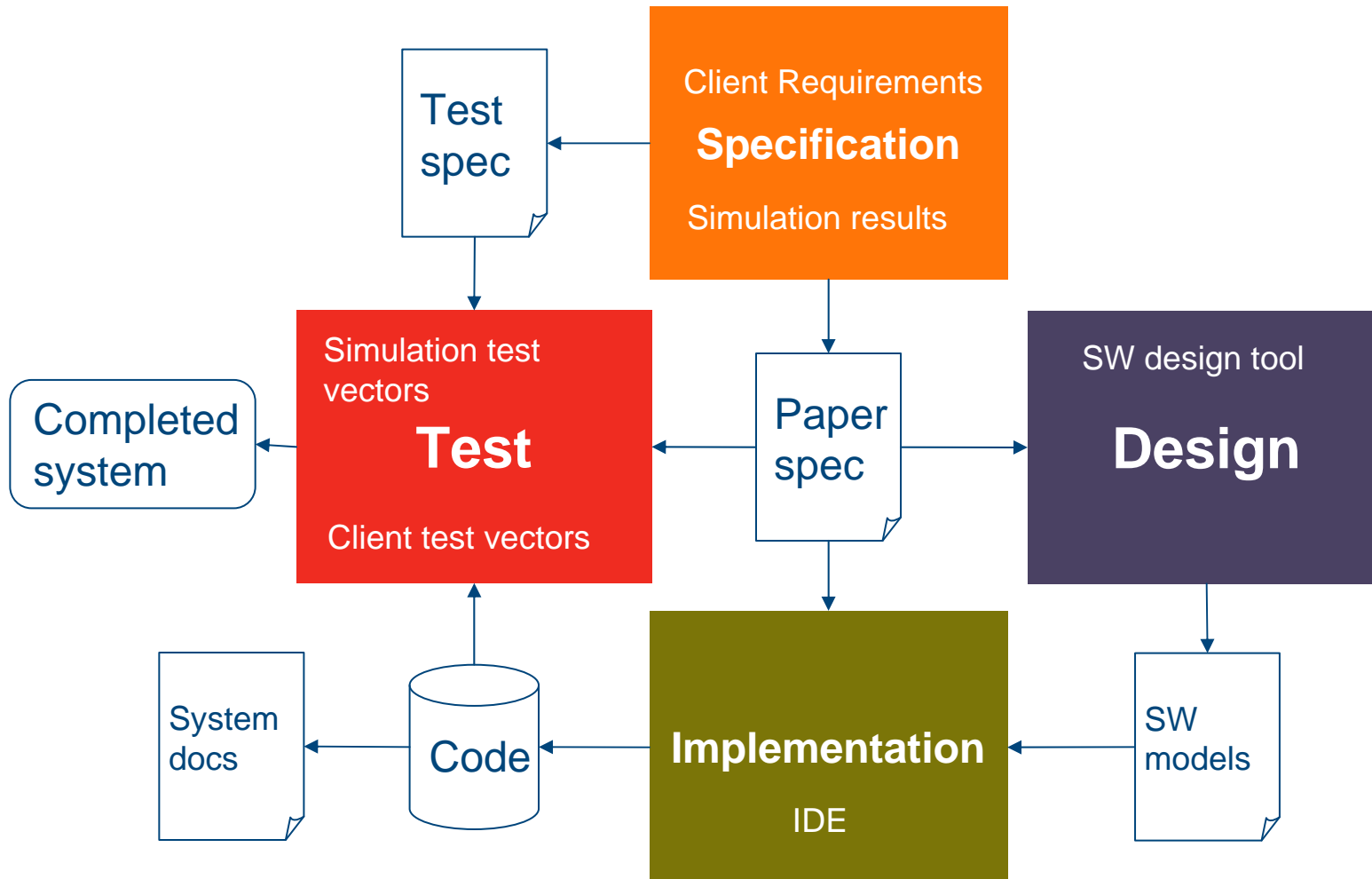


Developing Software Defined Radio Systems with  
**MATLAB & Simulink**

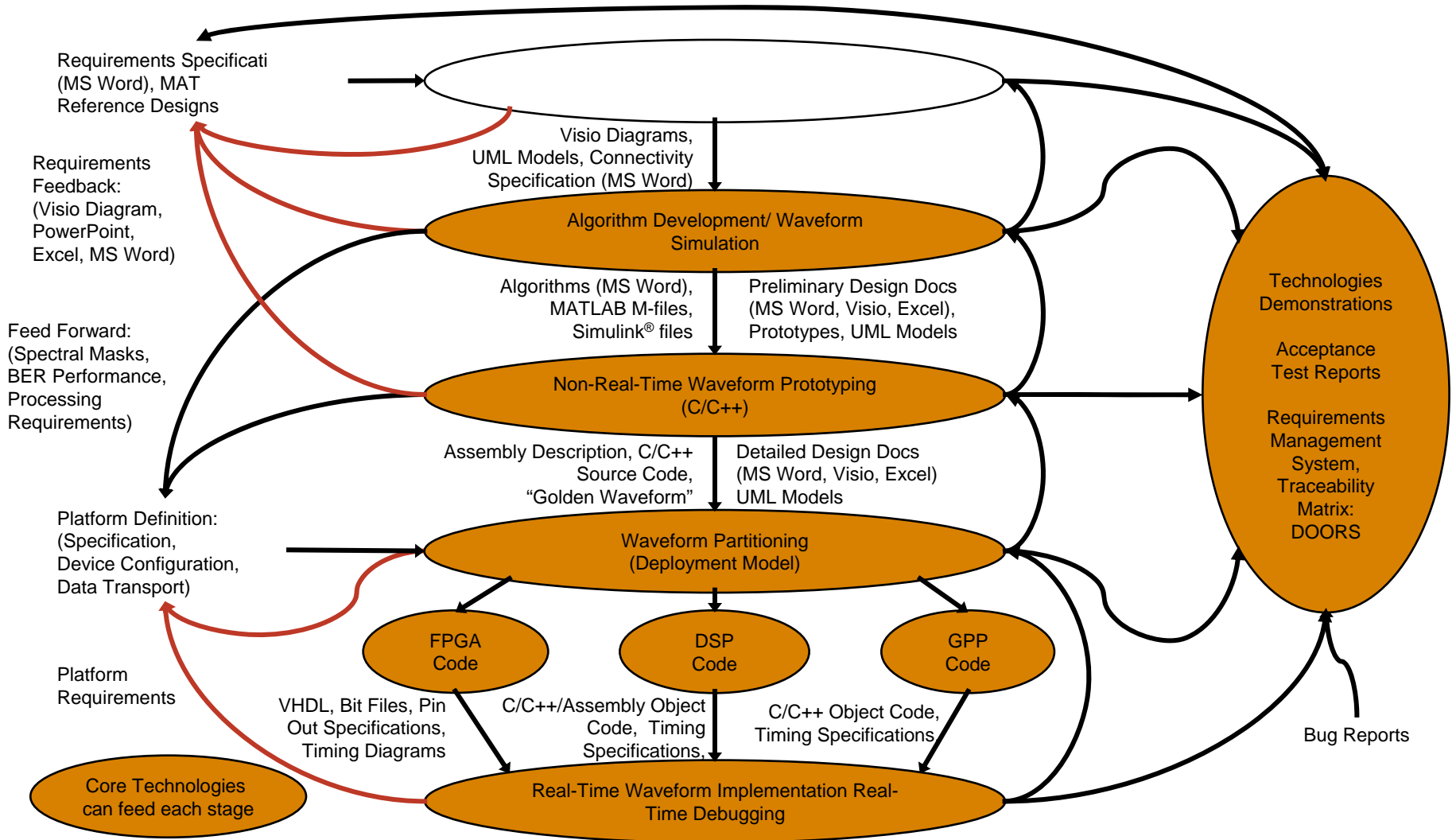


# Design Flows for SDR Development

# Traditional Design Flow

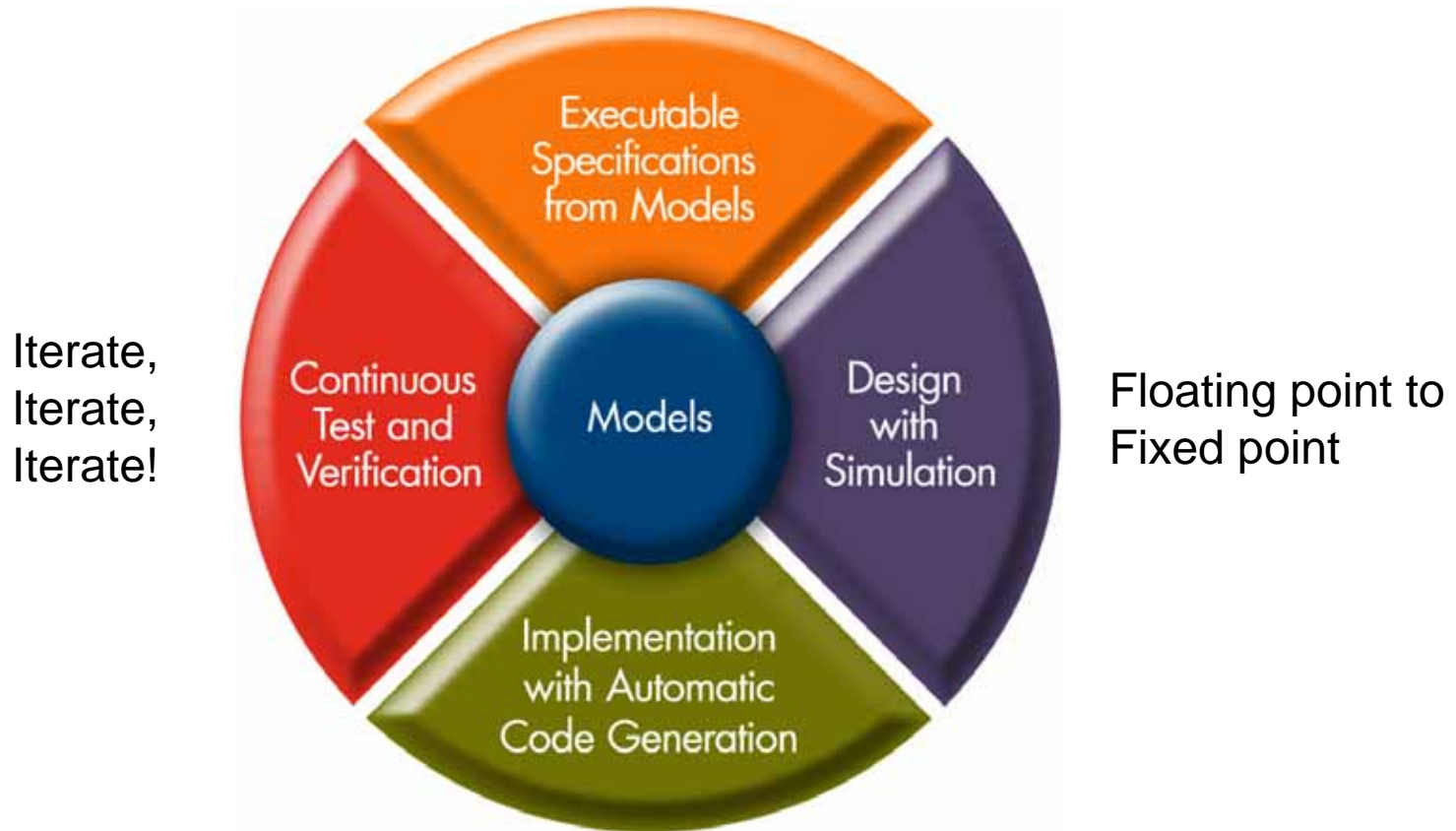


# Waveform Design (by SDR Forum Tools Work Group)



# Overview of Model-Based Design

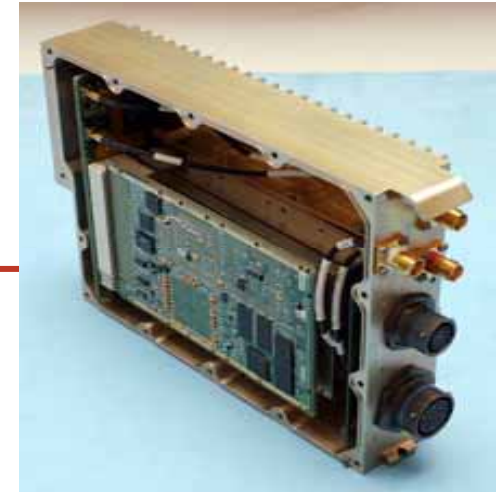
Links to paper-based specs



Import your own custom code



## BAE Systems Achieves 80% Reduction in Software-Defined Radio Development Time with Model-Based Design



### The Challenge

To develop a military standard SDR waveform for satellite communications

### The Solution

Use Simulink and Xilinx System Generator to rapidly design, debug, and automatically generate code for an SDR signal processing chain

### The Results

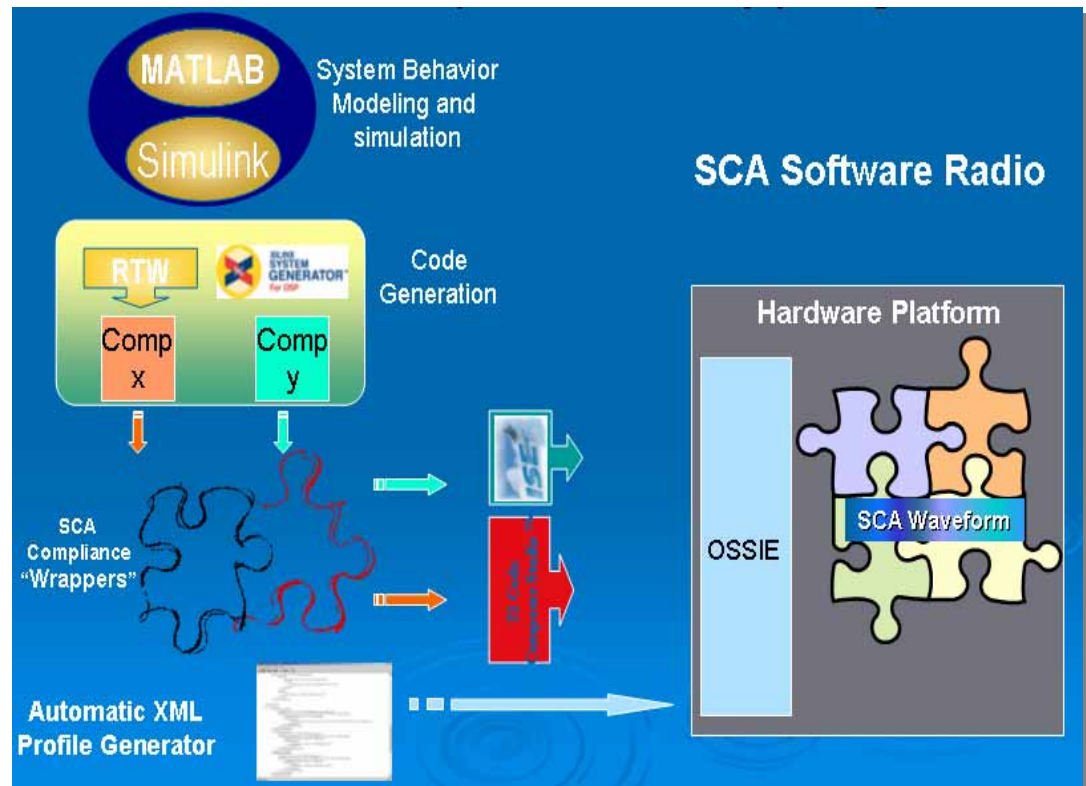
- Project development time reduced by 80%
- Problems found and eliminated faster
- Clocking and interfacing simplified

**“Using Simulink and Xilinx System Generator we designed and developed the signal processing chain of the SDR and achieved a 10-to-1 reduction in development time.”**

**Dr. David Haessig,  
BAE Systems**

# One SCA Rapid Development Project

- Develop glue code and wrappers that encapsulate signal processing code and interface it to SCA core framework
- Automatic code generation using MathWorks Real-Time Workshop® and Xilinx System Generator™
- TI Code Composer Studio™
- Zeligsoft CE for Automatic XML Profile Generation
- BAE expertise in SCA
- Virginia Tech Glue Code Development



# Demo: SDR Reference Waveform – FM3TR

# FM3TR Reference Waveform

- Future Multiband, Multiwaveform, Modular, Tactical Radio Waveform (Reference waveform for SDR Forum)

<b>Frequency range</b>	<b>30-400 kHz</b>
<b>Channel spacing</b>	<b>25 kHz</b>
<b>Modulation type</b>	<b>CPFSK</b>
<b>Modulation rate</b>	<b>25 kbps</b>
<b>Frequency hopping</b>	<b>250-500 hops/second</b>
<b>Framing, packetization</b>	<b>Switched, packet</b>
<b>CVSD voice coder</b>	<b>16 kbps</b>
<b>Coding</b>	<b>Reed-Solomon</b>

# The MathWorks FM3TR Software Defined Radio Example

- Design, simulation, performance analysis
- Fixed-point analysis
- Automatic code generation
- Flexible system partitioning into components

# Software Defined Radio Example

**Software Defined Radio - FM3TR Reference Waveform  
Fixed-Point MSK Modulation/Demodulation**

**MSK Rimoldi-Style Modulator/Demodulator**

TRANSMITTER → AWGN → RECEIVER → BER → 0

BER  
Errors Count  
Samples Count

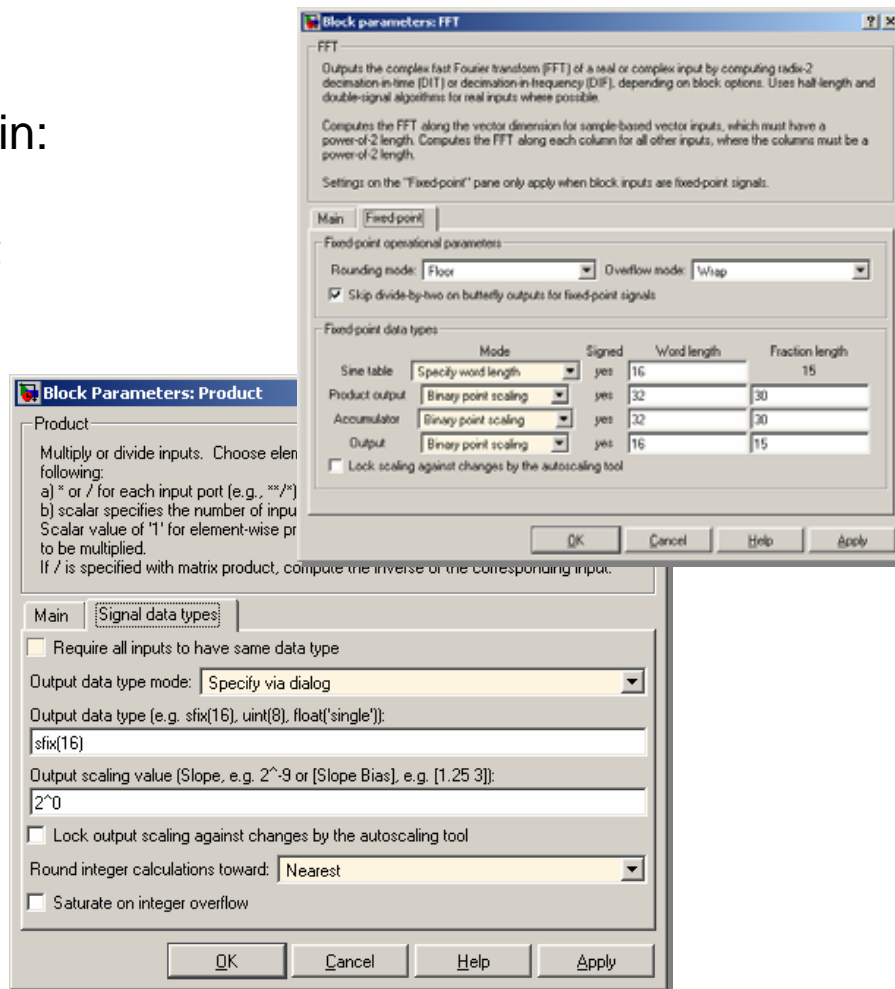
**References:**  
SDR FSDRF-01-I-0056-V0.00  
Description of the FM3TR Proposed Reference Waveform Modulation Format  
H. Leib and S. Pasupathy, "Error-Control Properties of MSK,"  
IEEE Communications Magazine, Jan. 1993  
B. Rimoldi, "A Decomposition Approach to CPM,"  
IEEE Transactions on Information Theory, Mar. 1998

Ready 100% FixedStepDiscrete

# Fixed-Point System Design

# Fixed-Point Design in Simulink®

- Simulink Fixed Point
  - Enables fixed-point support in:
    - Simulink
    - Signal Processing Blockset
    - Stateflow®
- Fixed-point settings at:
  - Block level
  - Subsystem level
  - Model level
- Control over
  - Inputs
  - Outputs
  - Internal values



**Block parameters: FFT**

FFT  
Outputs the complex fast Fourier transform (FFT) of a real or complex input by computing radix-2 decimation-in-time (DIT) or decimation-in-frequency (DIF), depending on block options. Uses half-length and double-signal algorithms for real inputs where possible.

Computes the FFT along the vector dimension for sample-based vector inputs, which must have a power-of-2 length. Computes the FFT along each column for all other inputs, where the columns must be a power-of-2 length.

Settings on the "Fixed-point" pane only apply when block inputs are fixed-point signals.

Main | Fixed point

Fixed point operational parameters

Rounding mode: Floor    Overflow mode: Wrap

Skip divide-by-two on butterfly outputs for fixed-point signals

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Sign table	Specify word length	yes	16	15
Product output	Binary point scaling	yes	32	30
Accumulator	Binary point scaling	yes	32	30
Output	Binary point scaling	yes	16	15

Lock scaling against changes by the autoscaling tool

OK    Cancel    Help    Apply

---

**Block Parameters: Product**

Product

Multiply or divide inputs. Choose element following:  
 a) \* or / for each input port (e.g., \*\*/\*)  
 b) scalar specifies the number of input  
 Scalar value of '1' for element-wise product.  
 If / is specified with matrix product, compute the inverse of the corresponding input.

Main | Signal data types

Require all inputs to have same data type

Output data type mode: Specify via dialog

Output data type (e.g. sfixed(16), uint(8), float('single')):  
sfixed(16)

Output scaling value (Slope, e.g. 2^-9 or [Slope Bias], e.g. [1.25 3]):  
2^0

Lock output scaling against changes by the autoscaling tool

Round integer calculations toward: Nearest

Saturate on integer overflow

OK    Cancel    Help    Apply



# Tool: Fixed-Point Settings

Fixed-Point Settings - fm3tr\_fixpt\_dsp/Receiver/MSK Demodulator1

Select current system:

Fixed-Point settings for blocks in the current system

Logging mode:  Controlled by: fm3tr\_fixpt\_dsp

Data type override:

Simulation data logged for current system

Block Name:	Min	Max	Data Type	Scaling
Matrix Sum : Accumulator	-69.2	61.39	S32	V=Q*2 <sup>-11</sup>
Matrix Sum : Output	SATURATION occurred 358 time(s).			
Add1	-32	32	S16	V=Q*2 <sup>-10</sup>
Product1	-64	64	S32	V=Q*2 <sup>-24</sup>
Data Type Conversion1	-14.13	13.99	S32	V=Q*2 <sup>-26</sup>
Matrix Sum1 : Accumulator	-13.23	13.99	S16	V=Q*2 <sup>-11</sup>
Matrix Sum1 : Output	-69.88	75.37	S32	V=Q*2 <sup>-11</sup>
Add2	-66.88	75.37	S16	V=Q*2 <sup>-8</sup>
Data Type Conversion2	-102.8	107.7	S32	V=Q*2 <sup>-24</sup>
Data Type Conversion3	-64	64	S16	V=Q*2 <sup>-8</sup>
	-102.8	107.7	S16	V=Q*2 <sup>-8</sup>

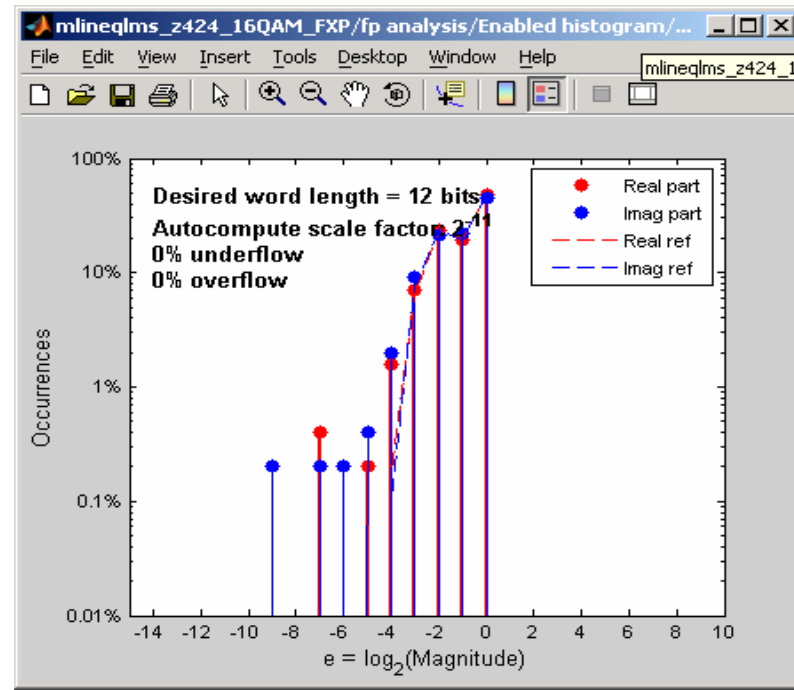
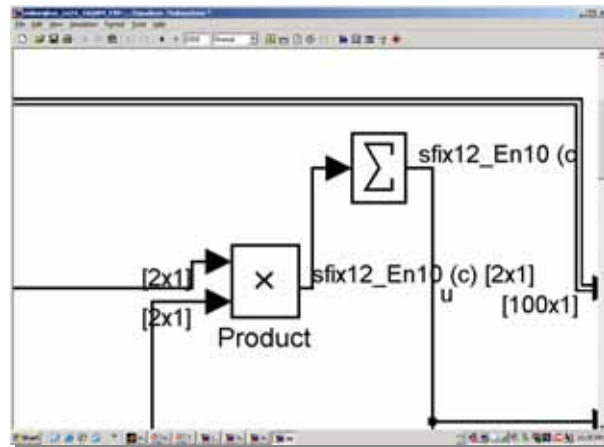
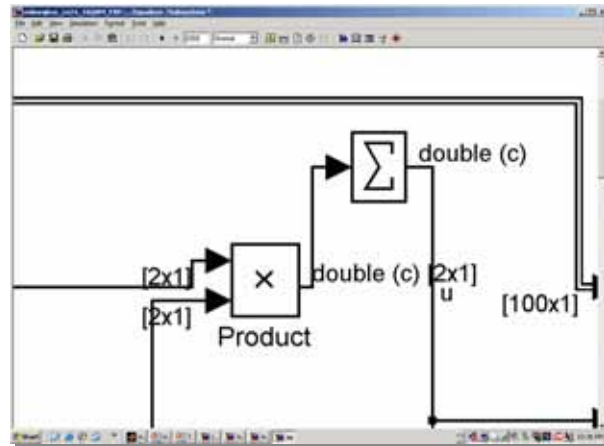
Logging type:

Autoscale current system

Safety margin:

# Fixed-Point Design Aids

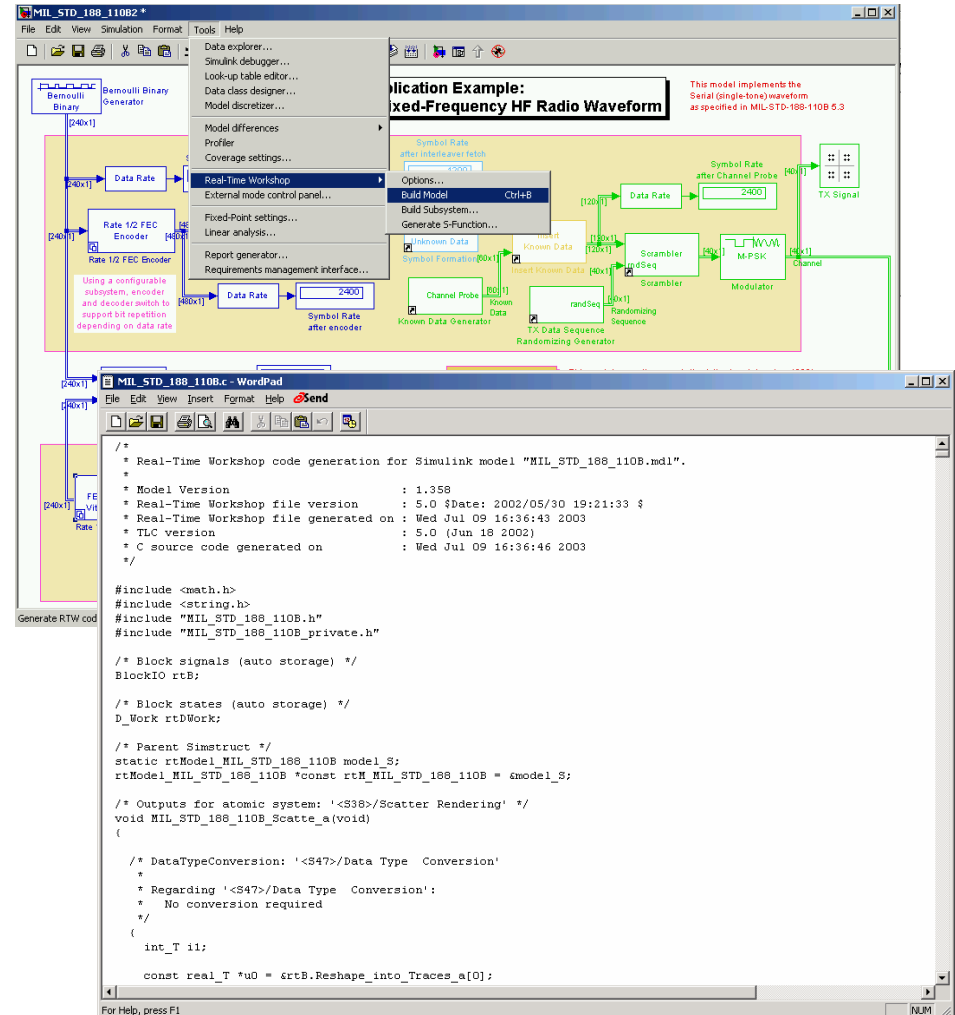
- Model annotation
- Histogram techniques



# Automatic Code Generation

# Code Generation from Models

- Target generic C-programmable devices
- Production Code for embedded systems
  - ROM size, RAM size, Execution Speed: Comparable with optimized handwritten code
  - Compact ERT code format ANSI and ISO floating-point libraries
  - Customized main program: deploy on target with or without OS
  - Supports user-defined data objects and S-functions
  - Detailed HTML Reports



The screenshot displays the MATLAB/Simulink environment. The top window shows a Simulink model titled 'MIL\_STD\_188\_11082'. The model is a 'Fixed-Frequency HF Radio Waveform' implementation. It includes blocks for a Bernoulli Binary Generator, Data Rate, Rate 1/2 FEC Encoder, Symbol Rate after Channel Probe, Symbol Rate after encoder, Known Data, Channel Probe, Known Data Generator, TX Data Sequence Randomizing Generator, randSeq, Scrambler, Scrambler, WMM M-PSK Modulator, and TX Signal. A context menu is open over the 'Rate 1/2 FEC Encoder' block, with 'Build Model' selected. A tooltip for the encoder block reads: 'Using a configurable subsystem, encoder and decoder switch to support bit repetition depending on data rate'. A red text box on the right states: 'This model implements the Serial (single-tone) waveform as specified in MIL-STD-188-110B 5.3'. Below the model, a 'WordPad' window shows the generated C code for 'MIL\_STD\_188\_1108B.c'. The code includes headers, comments about the generation process, and the start of a main function.

```

/*
 * Real-Time Workshop code generation for Simulink model "MIL_STD_188_1108.mdl".
 *
 * Model Version                : 1.358
 * Real-Time Workshop file version : 5.0 [Date: 2002/05/30 19:21:33 $]
 * Real-Time Workshop file generated on : Wed Jul 09 16:36:43 2003
 * TLC version                  : 5.0 (Jun 18 2002)
 * C source code generated on    : Wed Jul 09 16:36:46 2003
 */

#include <math.h>
#include <string.h>
#include "MIL_STD_188_1108.h"
#include "MIL_STD_188_1108_private.h"

/* Block signals (auto storage) */
BlockIO rtB;

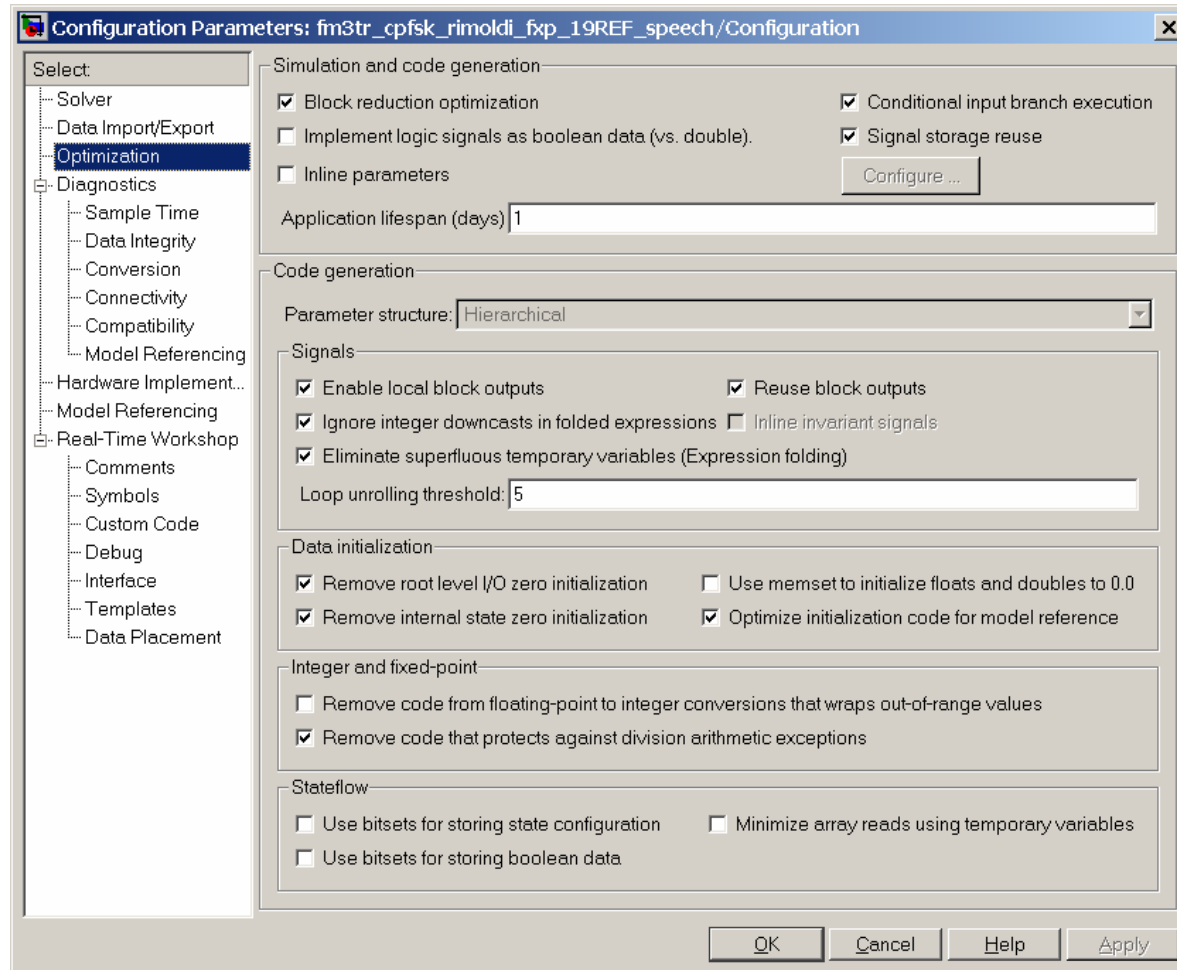
/* Block states (auto storage) */
D_Work rtDWork;

/* Parent Simstruct */
static rtModel_MIL_STD_188_1108 model_S;
rtModel_MIL_STD_188_1108 *const rtm_MIL_STD_188_1108 = smodel_S;

/* Outputs for atomic system: '<S38>/Scatter Rendering' */
void MIL_STD_188_1108_Scatter_a(void)
{
    /* DataTypeConversion: '<S47>/Data Type Conversion'
     *
     * Regarding '<S47>/Data Type Conversion':
     * No conversion required
     */
    int_T i1;

    const real_T *u0 = <rtB.Reshape_into_Traces_a[0]>;
  
```

# Code Optimization Options



# Code Customization Options

The image displays four overlapping screenshots of the Configuration Parameters dialog box for the 'fm3tr\_inopt\_dsp/Configuration' project, illustrating various code customization options.

- Overall control:** Shows options for including comments, auto-generated comments (Simulink block comments, Show eliminated statements, Verbose comments for Simulink Global storage class), and custom comments (Simulink block descriptions, Simulink data object descriptions, Custom comments for MPT objects only, Stateflow object descriptions, Requirements in block comments).
- Auto-generated identifiers naming rules:** Shows identifier format control (Global variables, Global types, Field name of global types, Subsystem methods, Local temporary variables, Local block output variables, Constant macros) and Simulink data object naming rules (Signal naming, Parameter naming, Rdefine naming).
- Software environment:** Shows target floating-point math environment (ANSI-C), utility function generation (Auto), support for floating-point numbers, non-finite numbers, complex numbers, absolute time, continuous time, and non-inlined S-functions. It also includes code interface options (GRT compatible call interface, Single output/update function, Terminate function required, Generate reusable code, Reusable code error diagnostic) and verification options (Create Simulink (S-Function) block, MAT file logging).
- Memory sections:** Shows package selection (Simulink) and memory sections for model functions and subsystem defaults (Initialize/terminate, Execution, Constants, Inputs/Outputs, Internal data, Parameters).

# Example Design Flows for Target Platforms



# DSP Design Flow

The MathWorks



Link for Code Composer Studio™, Real-Time Workshop®, Embedded Target for TI C6000™ DSP

MATLAB & SIMULINK

Specific Peripheral Software Drivers

Target-Specific and Optimized Code



Code Composer Studio™ (Compiler, Linker, and Loader)



Customer-Specific Board





# FPGA Design Flows

Algorithm Development, Simulation, Modeling

Translation, Code Gen to VHDL/Verilog

HDL Simulation

Synthesis

Implementation and Verification

The MathWorks



MATLAB & SIMULINK



Model Sim®



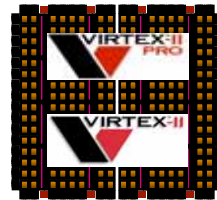
XST®



Leonardo Spectrum®



ISE® 4.1i



Synplify®



ALTERA

Signal Compiler®

ALTERA



Leonardo Spectrum®



Synplify®

Quartus II®



SYNOPSYS® I

# Conclusion

Model-Based Design is a crucial methodology for successful design of Software Defined Radios

- Simulations
- Fixed-point design
- Code generation
- Deployment into diverse hardware and software platforms
- Integration with SCA tools