



4月 - 北京 · 上海 · 深圳

# 2015 MATLAB 巡回研讨会

技术融合的时代



# 运用MATLAB加速嵌入式算法开发

**MathWorks China**

应用工程师

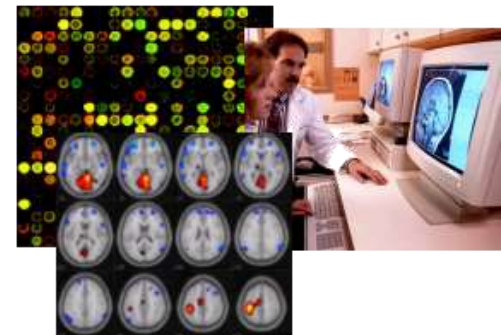
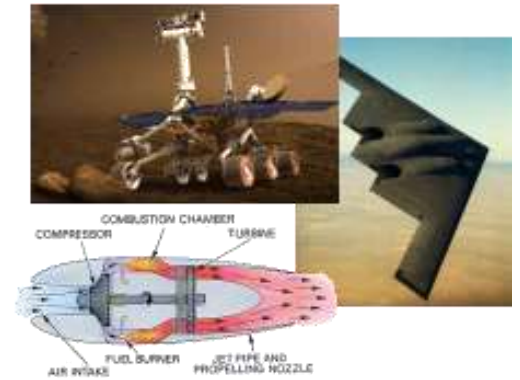
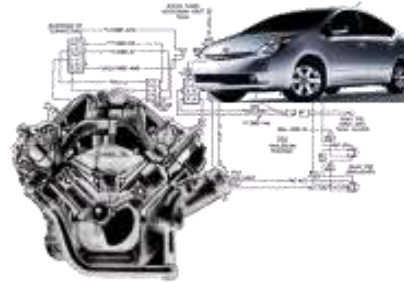
单博

# 主要内容

- 嵌入式应用仿真
- 浮点模型转化为定点模型的工作流程
  - 自动收集数据并建议数据类型
  - 推导数据类型
- 将定点模型自动转换为嵌入式C代码
  - 代码生成与优化
  - 验证 Polyspace
  - 硬件连接
- 国内典型用户案例分析

# 嵌入式系统

- 手机、pad
- 家电
- 网络设备
- 工业控制
- 仪器仪表
- 医疗
- 汽车
- 飞机
- 机器人
- 物联网



# 嵌入式算法的设计挑战

MATLAB / SIMULINK  
算法设计

FIXED-POINT DESIGNER  
浮点到定点的转换

MATLAB CODER

SIMULINK CODER

EMBEDDED CODER

HDL CODER



C/C++

VHDL/Verilog

MCU

DSP

FPGA

ASIC



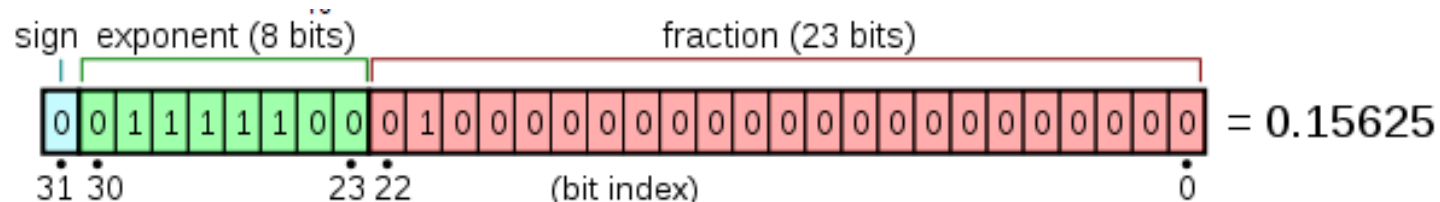
设计狮

定点化过程会  
占据总设计时  
间超过**25%+**



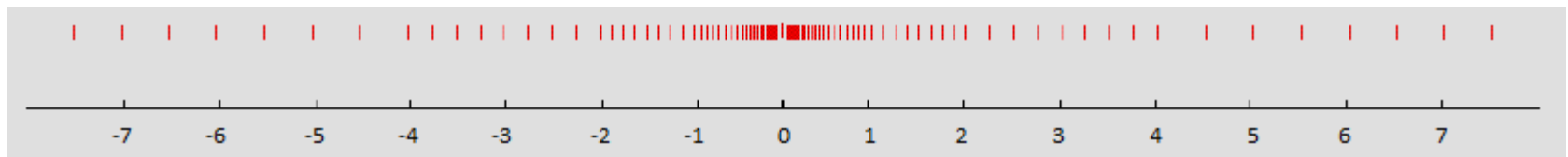
# What is floating-point?

- Characterized by **SIGN** bit, **MANTISSA**(Fraction) and **EXPONENT** IEEE 754 Single Precision format (Normalized): 32 bits word size(single)



$$value = (-1)^{sign} \left( 1 + \sum_{i=1}^{23} b_{-i} 2^{-i} \right) \times 2^{(e-127)}$$

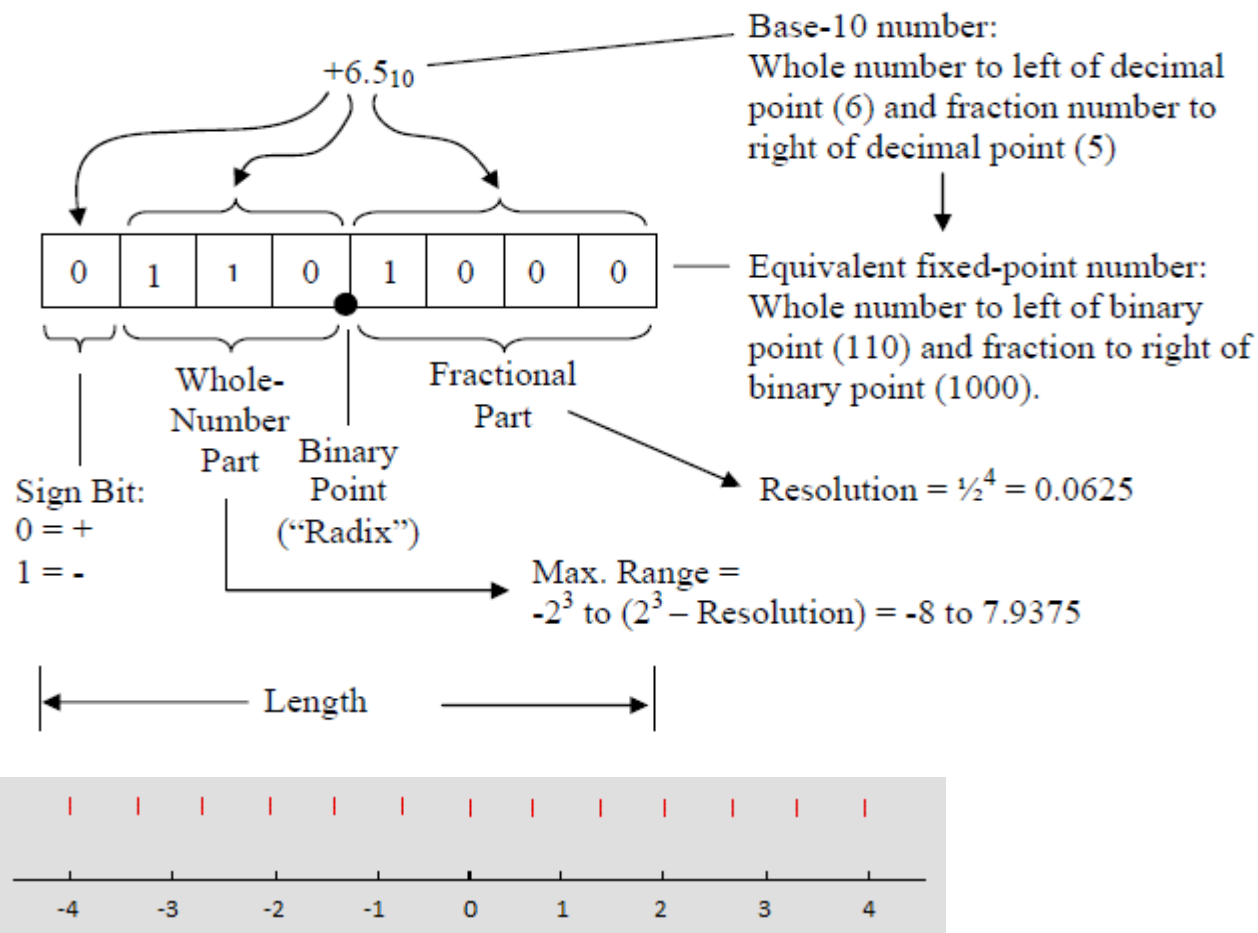
- Use of three separate fields **increases** both **range** (Exponent size) and **precision** (Fraction size) of floating point numbers



Above picture from [http://en.wikipedia.org/wiki/Single\\_precision\\_floating\\_point\\_format](http://en.wikipedia.org/wiki/Single_precision_floating_point_format)

# What is fixed-point?

- Characterized by single WORD with fixed RADIX Point
- Use fractional numbers without floating point
- For a fixed size, trade-off between Precision and Range



# 嵌入式目标器件

- **DSP/MCU** (TI, Analog Devices, etc.)
  - 定点DSP比浮点DSP便宜很多
  - 定点DSP的功耗更低
  - 定点DSP的时钟频率更高
  - 固定字长
- **FPGA** (Xilinx, Altera, etc.)
  - 在定点FPGA实现中，字长每增加1个比特都意味着消耗更多的片上资源和功耗
  - 设计师可以改变字长



## Example: 定点C手工实现

```
void differentialEq( void )
```

```
{  
    /* Implements a fixed point first order difference equation */
```

```
    int Prod;  
    long Accum;  
    static short lastVal=0;
```

```
    short a=0x7eb8; // 0.99 in s16,15  
    short oneminusa=0x0148; // .01 in s16,15  
    short temp;
```

```
    Prod = gAlg_in1 * gAlg_in1;
```

```
    temp = Prod >> 15;
```

```
    Accum = a*lastVal + oneminusa*temp;
```

```
    gAlg_out1 = (short)(Accum >> 15);
```

```
    lastVal = gAlg_out1;
```

```
}
```

把变量转换成整形

需要很多注释帮助理解代码

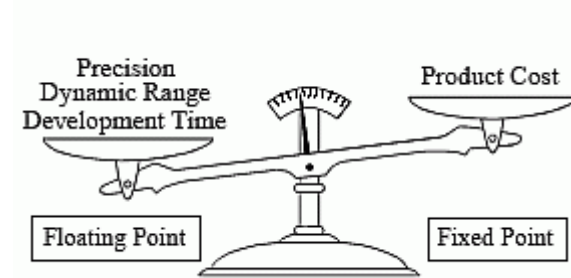
跟踪二进制小数点位置

没有饱和或取整

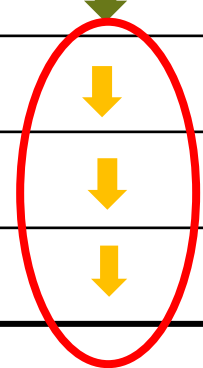
# DSP/MCU/FPGA上需要做的定点工作

- 芯片做整数运算，不是定点运算.
- 定点工作需要你自己做.
- 加减法：
  - 你需要对齐二进制小数点 (>> or <<), 然后相加.
- 乘法：
  - 整数相乘，然后你解释乘积的二进制小数点.
- 除法：
  - 你很可能用不了“/”. 你要自己写个函数或者调用库函数.
- 开方：
  - 不能用标准库函数，你要自己写一个函数.
- 取整/饱和：
  - 你想要？你自己做.

# Fixed Point Tradeoffs



Consideration	Floating Point	Fixed Point	Fixed Point with MathWorks Tools
RAM/ROM消耗	↑	↓	↓
执行速度	↓	↑	↑
硬件功耗	↑	↓	↓
硬件成本	↑	↓	↓
开发时间	↓	↑	↓
实现复杂度	↓	↑	↓
出错率	↓	↑	↓



# 定点的弊端及解决方案

## 弊端1

开发时间更长

## 解决方案

采用仿真、自动化验证和自动化量化工具将会缩短开发时间和减少开发资源

## 弊端2

因为实现复杂经常引入错误

## 解决方案

采用快速原型，在环测试，产品级代码生成手段能明显地帮助减少错误

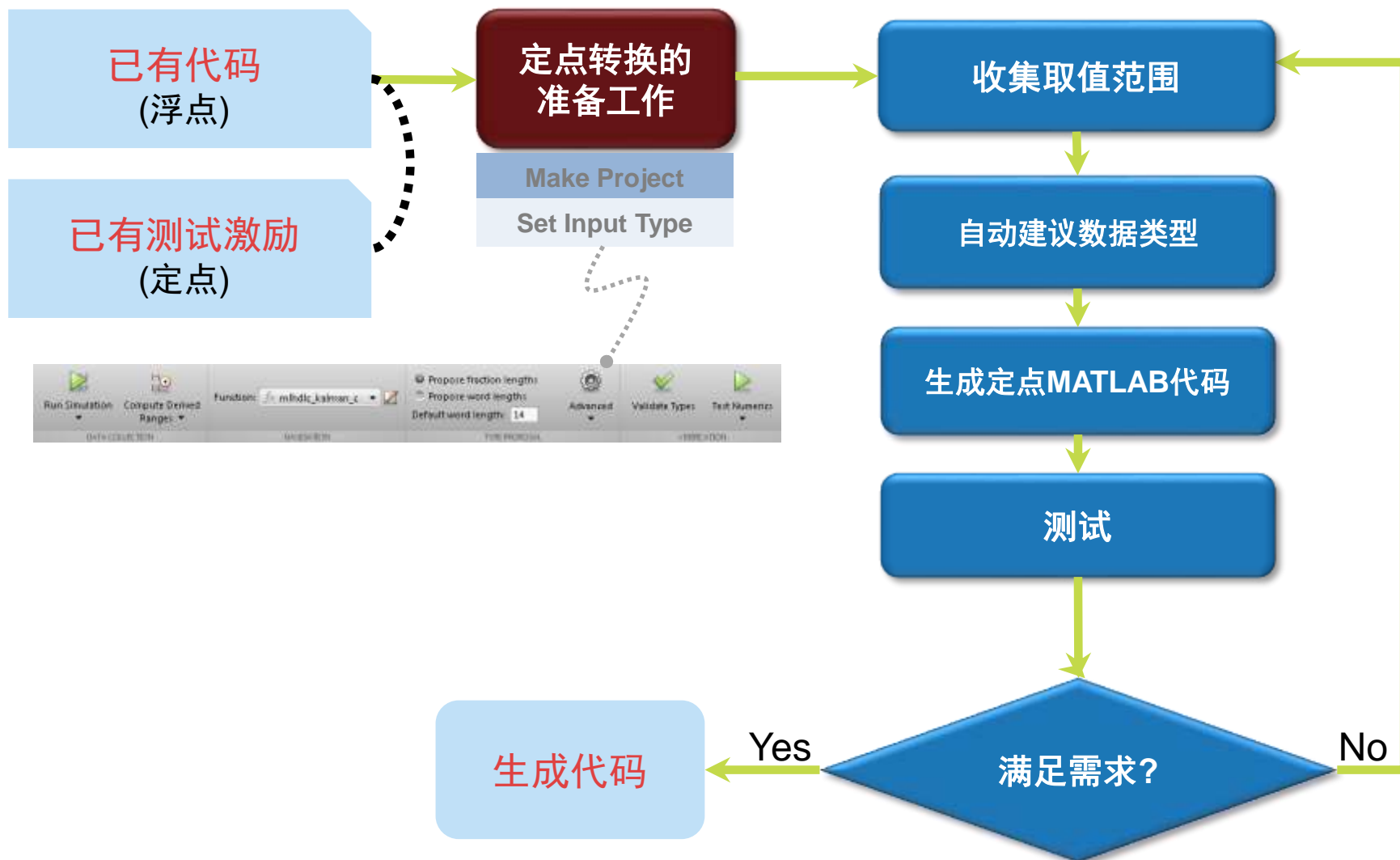
## 弊端3

动态范围变小引入量化误差

## 解决方案

合理的选择定标（小数点位置）和字长（硬件限制）能减少量化误差

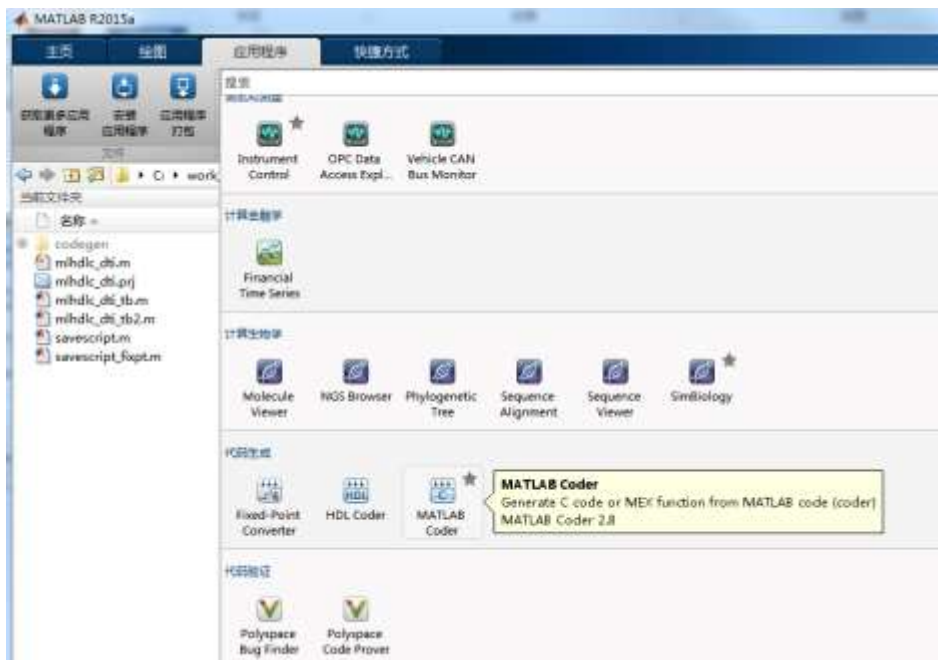
# MATLAB/Simulink 定点化工作流程



# 例子

- 工作流程: 浮点模型-> 定点模型-> C
- 仿真 vs 推导

## 1 打开MATLAB Coder APP



## 2 使能Fixed-point conversion 选择被测m文件

- 定点转换
- 生成C代码



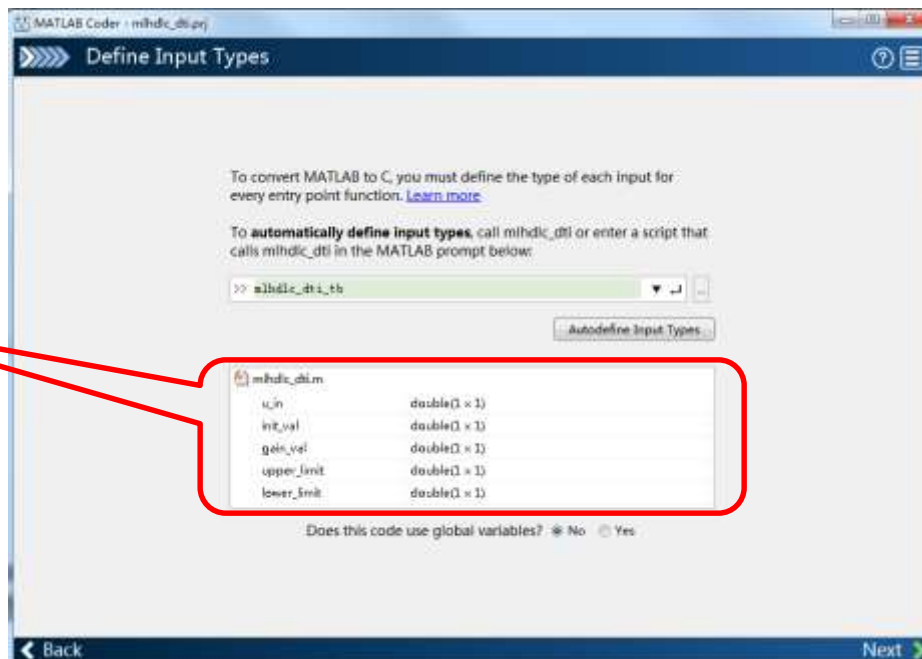
# 例子

## 3 选择testbench文件

- 自动识别被测文件的输入输出数据类型

### 实现约束

- 多形性
- 内存分配
- 矩阵行列处理
- 定点数据类型



function a= foo(b,c)  
a = b \* c;

**c**

```
double foo(double b, double c)
{
    return b*c;
}
```

```
void foo(const double b[15],
         const double c[30], double a[18])
{
    int i0, i1, i2;
    for (i0 = 0; i0 < 3; i0++) {
        for (i1 = 0; i1 < 6; i1++) {
            a[i0 + 3 * i1] = 0.0;
            for (i2 = 0; i2 < 5; i2++) {
                a[i0 + 3 * i1] += b[i0 + 3 * i2] * c[i2 + 5 * i1];
            }
        }
    }
}
```

# 例子

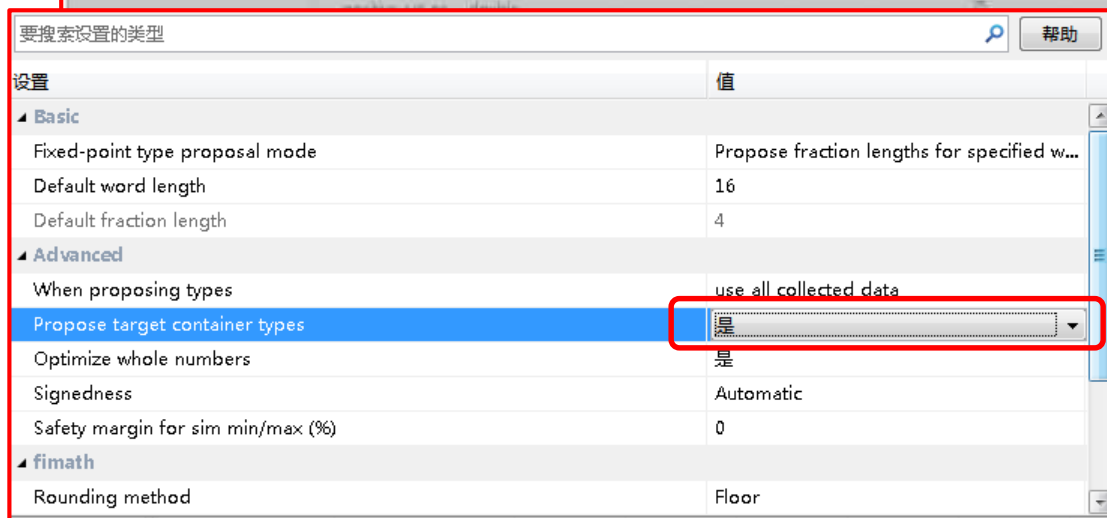
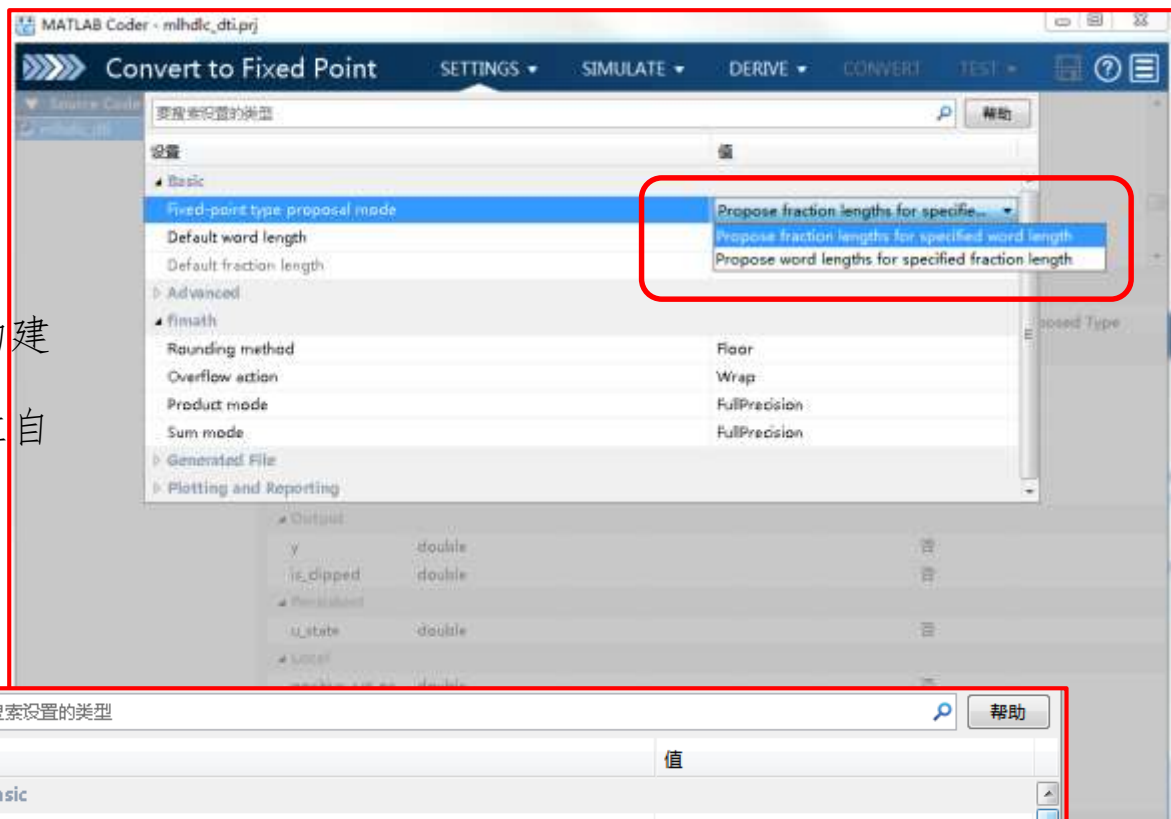
## 4 定点化配置

### 定点化模式

- C代码生成：固定字长自动建议小数位
- HDL代码生成：固定小数位自动建议字长

### 以目标器件类型

- 字长为8bit的整数倍





# 例子：用仿真结果自动建议定点类型

多个测试文件批量仿真

测试代码覆盖率  
用颜色区分

Figure 3: mihdc\_dti\_tb2\_plot

测试文件1

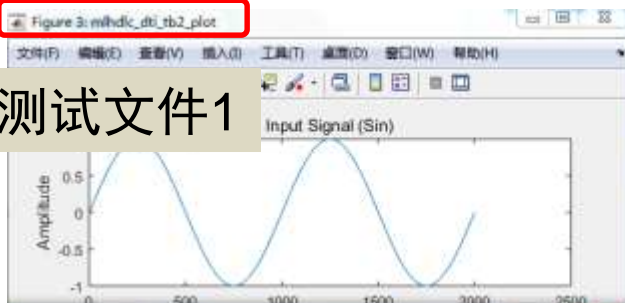
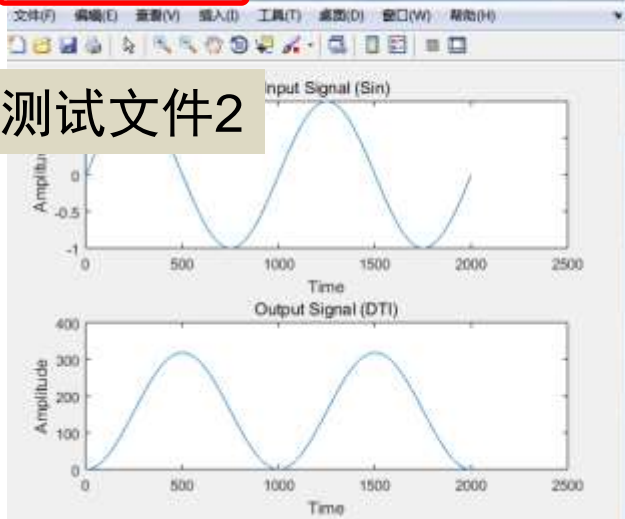


Figure 2: mihdc\_dti\_tb1\_plot

测试文件2



MATLAB Coder - mihdc\_dti.prj

Convert to Fixed Point

```

33 *****
34 *****
35 if (u_state >= upper_limit)
36     yt = upper_limit;
37     is_clipped = positive_sat_occurred;
38 elseif (u_state <= lower_limit)
39     yt = lower_limit;
40     is_clipped = negative_sat_occurred;
    
```

Reached 8% of the time

Not reached

Variable	Type	Sim Min	Sim Max	Static Min	Static Max	Whole N...	Proposed Type
Input							
u_in	double	-1	1			否	numerictype(1, 16, 14)
init_val	double	0	0			是	numerictype(0, 8, 0)
		1	2			是	numerictype(0, 8, 0)
		500	500			是	numerictype(0, 16, 0)
		-500	-500			是	numerictype(1, 16, 0)
is_clipped	double	0	1			否	numerictype(1, 16, 6)
Persistent							
u_state	double	-136.62	501.63			否	numerictype(1, 16, 6)
Local							
positive_sat_oc...	double	1	1			是	numerictype(0, 8, 0)
negative_sat_o...	double	-1	-1			是	numerictype(0, 8, 0)
no_sat_occurred	double	0	0			是	numerictype(0, 8, 0)
yt	double	-136.62	500			否	numerictype(1, 16, 6)
tared	double	-2	2			是	numerictype(0, 8, 0)

仿真过程中记录  
变量的最大最小值

自动推荐的定  
点类型，字长  
是8bit的倍数

# 例子：推导并自动建议定点类型

Convert to Fixed Point SETTINGS SIMULATE DERIVE CONVERT TEST

Source Code mlhdlc\_dti

```

1 #codegen
2 function [y, is_clipped] = mlhdlc_dti(u_in, init_val, gain_val, upper_limit, lower_limit)
3 % Discrete Time Integrator in MATLAB Function block
4
5 %
6 % Forward Euler method, also known as the Euler method, is a numerical method for solving ordinary differential equations (ODEs)
7 % or left-hand approximations of the continuous-time system. It is the simplest method for solving ODEs and is often used
8 % as a reference for comparing other methods. The output of the block at each time step is the value of the state variable.
9 %

```

根据输入和程序推导  
得到变量的最大最小值

Static Analysis Output

Variable	Type	Sim Min	Sim Max	Static Min	Static Max	Whole N...	Proposed Type
Input							
u_in	double	-1	1	-1	1	否	numerictype(1, 16, 14)
init_val	double	0	0	0	0	是	numerictype(0, 8, 0)
gain_val	double	1	2	1	2	是	numerictype(0, 8, 0)
upper_limit	double	500	500	500	500	是	numerictype(0, 16, 0)
lower_limit	double	-500	-500	-500	-500	是	numerictype(1, 16, 0)
Output							
y	double	-136.62	500	-500	500	否	numerictype(1, 16, 6)
is_clipped	double	0	1	-1	1	是	numerictype(1, 8, 0)
Persistent							
u_state	double	-136.62	501.63	-502	502	否	numerictype(1, 16, 6)
Local							
positive_set_oc...	double	1	1	1	1	是	numerictype(0, 8, 0)
negative_set_oc...	double	-1	-1	-1	-1	是	numerictype(1, 8, 0)
no_set_occurred	double	0	0	0	0	是	numerictype(0, 8, 0)
yt	double	-136.62	500	-500	500	否	numerictype(1, 16, 6)
tprad	double	-2	2	-2	2	否	numerictype(1, 16, 13)

Not reached

Back Next

# 例子：接受定点化

自动生成定点转换报告

The screenshot shows a 'Fixed-Point Report for mhdic\_dti'. It contains several code snippets with comments explaining the fixed-point conversion process, such as clipping the accumulator value and updating state variables. Below the code is a table summarizing the variables used in the report.

Variable Name	Type	Min	Max	Static Min	Static Max	Whole Number	Proposed Type (Best For M = 16)
gain_val	double	1	1	1	1	0	numeric(0, 8, 0)
init_val	double	0	0	0	0	0	numeric(0, 8, 0)
is_clipped	double	0	1	-1	1	1	numeric(1, 8, 0)
lower_limit	double	-500	-500	-500	-500	0	numeric(1, 14, 0)
negative_sat_occurred	double	-1	-1	-1	-1	1	numeric(1, 8, 0)
no_sat_occurred	double	0	0	0	0	0	numeric(0, 8, 0)
positive_sat_occurred	double	1	1	1	1	1	numeric(0, 8, 0)
upper	double	2	2	2	2	0	numeric(1, 14, 13)
u_state	double	1	1	1	1	1	numeric(1, 14, 13)

自动生成定点的m文件  
用于后续系统定点仿真

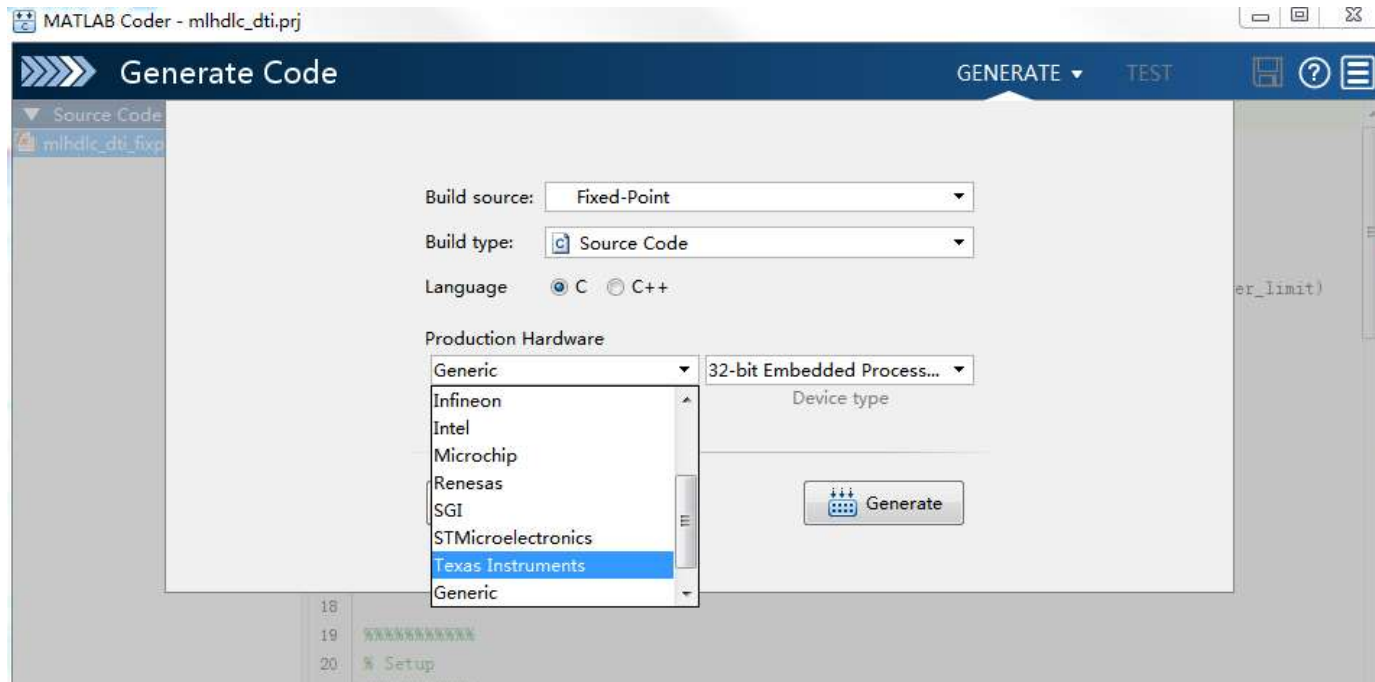
The screenshot shows the generated MATLAB file 'mhdic\_dti\_fixpt.m'. The code is a fixed-point implementation of the original system, using the 'fix' function for saturation and clipping. It includes comments for setup and state management.

```

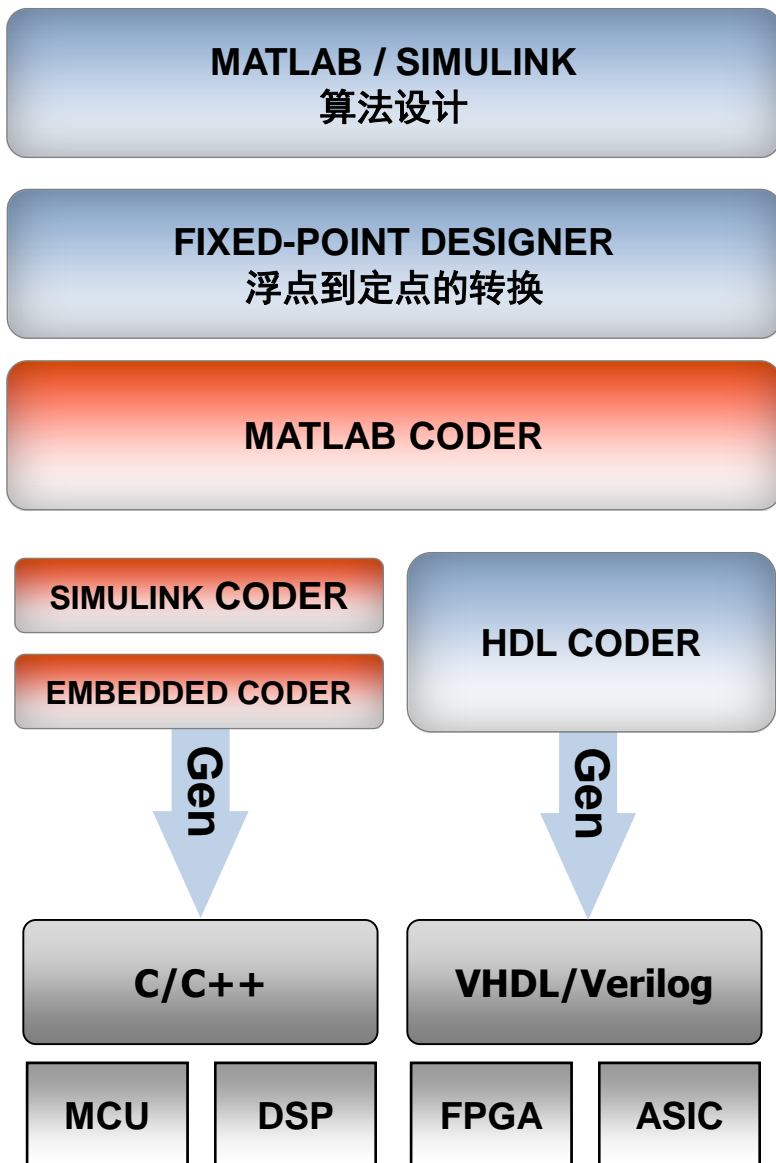
mhdic_dti_fixpt.m
0
1 % Setup
2
3 % numeric type to clip the accumulator value after each addition
4
5 % variable to hold state between consecutive calls to this block
6 fm = fix('RoundingMethod', 'Floor', 'OverflowAction', 'Wrap', 'ProductMode', 'FullPrecision', 'MaxProductWordLength', 128, 'SumMode', 'FullPrecision');
7
8 persistent u_state;
9 if isempty(u_state)
10     u_state = fi(init_val, 1, 16, 0, fm);
11 end
12
13 % clip flag status
14 positive_sat_occurred = fi(0, 8, 0, fm);
15 negative_sat_occurred = fi(-1, 1, 8, 0, fm);
16 no_sat_occurred = fi(0, 0, 8, 0, fm);
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
    
```

# 例子：自动生成C代码

选择目标硬件器件  
生成C源代码  
知识产权归客户



# 嵌入式算法的设计挑战



设计狮



程序猿

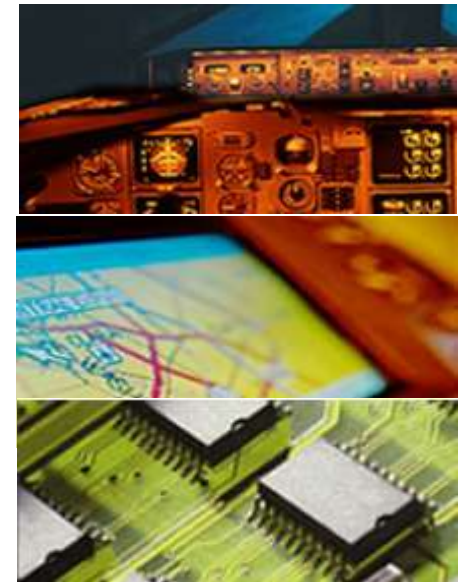
## 手工编码

- 费时
- 手写代码失误高
- 不易于保持 MATLAB 参考代码与 C 代码的一致性
- 不易于在开发阶段修改需求

# 嵌入式代码生成工具

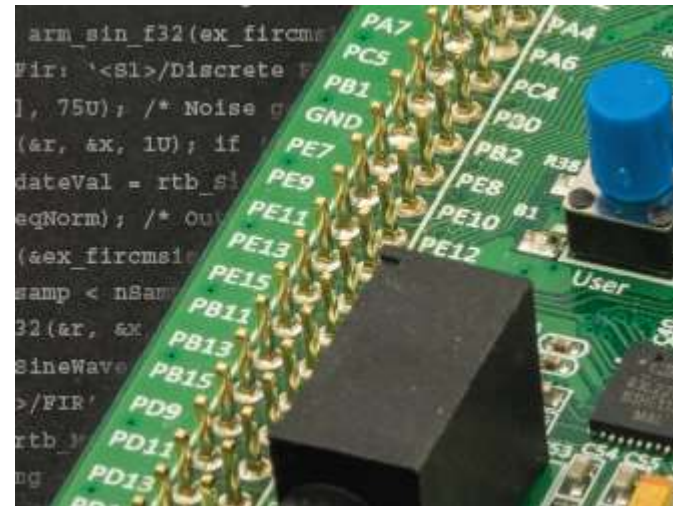
## Embedded Coder

- 从 MATLAB, Simulink, 和 Stateflow 生成 C/C++ code
- 针对嵌入式系统优化
- 可配置到多种器件
  - 快速原型开发板
  - 用于汽车控制器的MCUs
  - 用于信号处理系统的DSPs
  - 用于消费电子的ARMs
- 支持行业标准
  - DO-178, IEC 61508, ISO 26262, and EN 50128
  - AUTOSAR, ASAP2, and MISRA-C



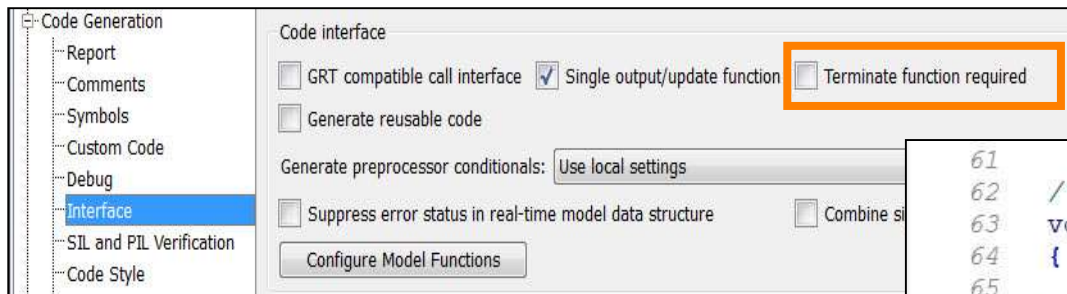
# 主要特性

- 用data dictionary管理存储类型，类型定义，别名
- 针对处理器优化
  - ARM Cortex-A and M
  - Intel IPP and Power Architecture SIMD
  - Customizable for any device
- 支持多速率、多任务、多核执行，包含或不包含RTOS
- 软件在环、处理器在环测试
  - 模型和代码结果比对
  - 代码剖析
  - 可集成代码覆盖率工具
- 定制注释、代码报告，支持需求文档、模型、代码之间的双向跟踪



# 移除不需要的代码或数据支持

- Removing initialization code (*Optimization pane*)
- Removing termination code (*Interface pane*)
- Removing data support (*Interface pane*)
- Disabling MAT-File Logging (*Interface pane*)
- Conditional Input Branch Execution (*Optimization pane*)



No termination  
function

```

61
62  /* Model initialize function */
63  void optim_example_initialize(void)
64  {
65      /* (no initialization code required) */
66  }
67
68  /*
69   * File trailer for generated code.
70   *
71   * [EOF]
72   */
73

```

An orange arrow points from the text 'No termination function' to the line containing '/\* [EOF]' in the code block.



# MATLAB /Simulink支持目标硬件

低成本硬件、免费硬件支持包



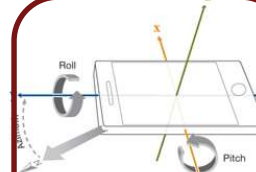
Arduino®



Raspberry Pi™



LEGO® NXT



iphone



BeagleBoard



PandaBoard



Gumstix® Overo



Microsoft Kinect



Zynq



TI C6000



USRP SDR



iRobot Create

<http://www.mathworks.cn/hardware-support/>

# 代码验证

## Polyspace C / C ++产品系列



- Polyspace Bug Finder
  - 快速找到嵌入式软件中的错误
  - 检查代码符合MISRA和JSF编码规则
  - 供软件工程师日常使用
- Polyspace Code Prover
  - 证明在给定的运行条件下，那些代码是 **安全可靠的**
  - 不需要穷举执行程序就可发现代码的 **运行时缺陷**, **边界条件** 和 **冗余代码**
  - 深度剖析代码的运行时行为和变量的数据范围
  - 证明软件内存共享的安全性
  - 采用数学分析的方法 – 不会漏报或误报任何运行时软件缺陷

# 自动代码生成已成为行业趋势

## User Story



Das Auto.

DAIMLER



BOSCH

DELPHI



THALES



JAGUAR



DENSO



Honeywell

CATERPILLAR®



TATA MOTORS

manroland



ALSTOM



ABB

medrad®  
Performance. For life.™



PHILIPS

# 通信与信号处理领域新特性

## 新产品

Antenna Toolbox(15a)

Vision HDL Toolbox(15a)

LTE System Toolbox(14a)

Phased Array System Toolbox(11a)

## 大调整

Fixed-point Designer(13a)

HDL Coder(12a)

Embedded Coder(11a)

Communication System Toolbox(11a)

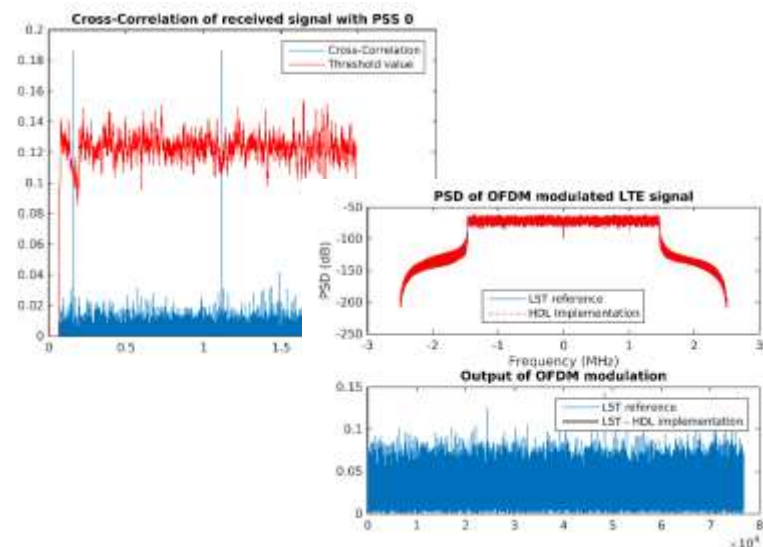
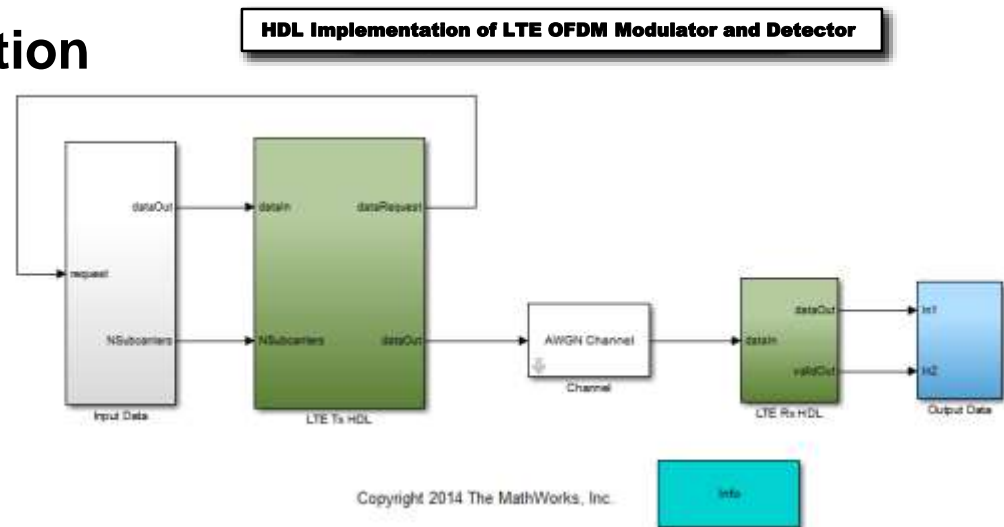
DSP System Toolbox(11a)

Computer Vision System Toolbox(10a)

# HDL Implementation of LTE OFDM Modulator and Detector

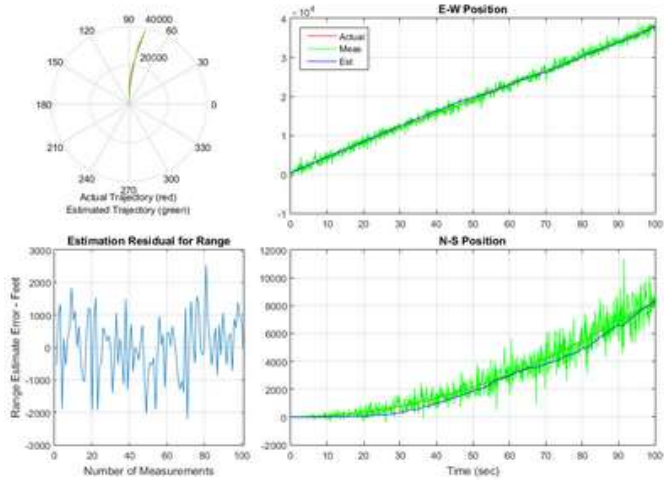
## HDL optimized implementation with golden reference verification

- Implementation of an LTE OFDM Modulator and Detector optimized for HDL code generation
- Verification of transmitter HDL implementation against a golden reference waveform using LTE System Toolbox
- Initial receiver detection and synchronization stages implemented in HDL coder

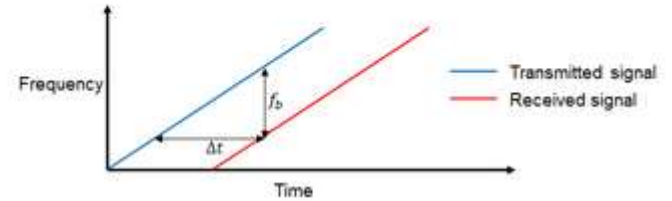


# 阵列信号处理

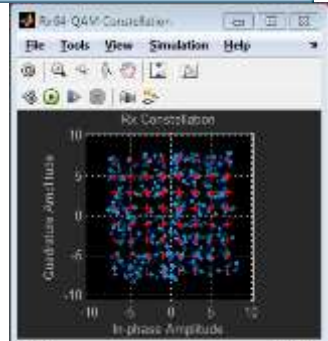
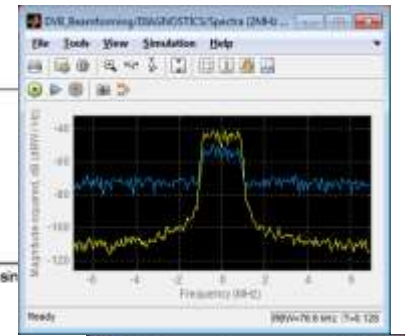
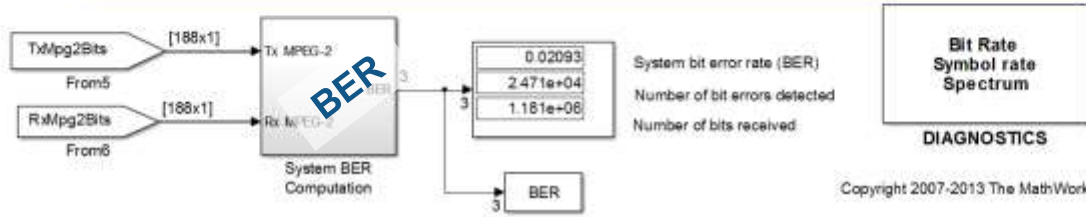
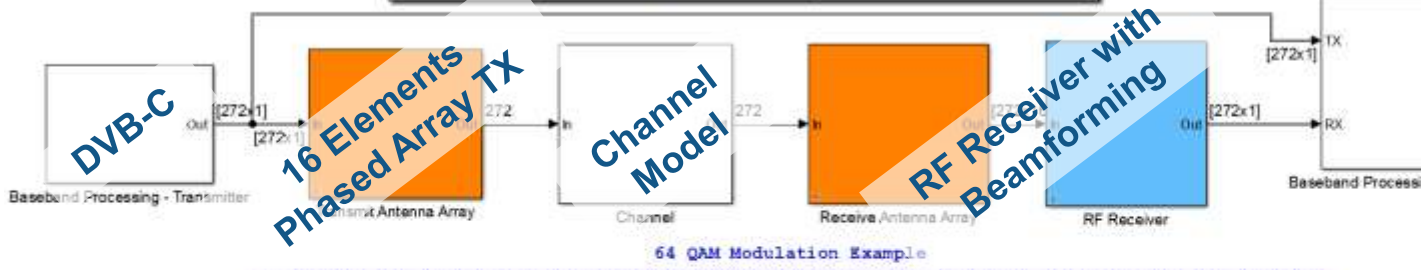
Radar tracking



Automotive Adaptive Cruise Control

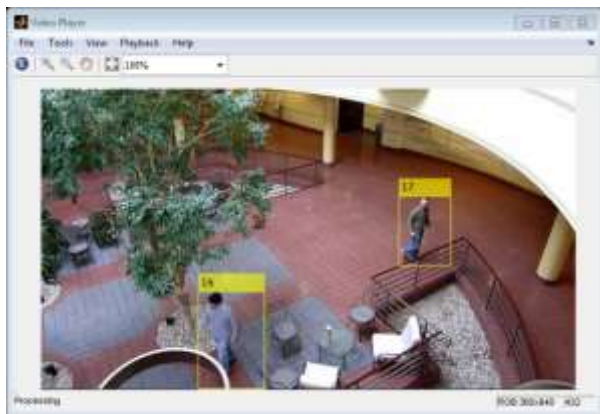


Wireless Digital Video Broadcasting with RF Beamforming

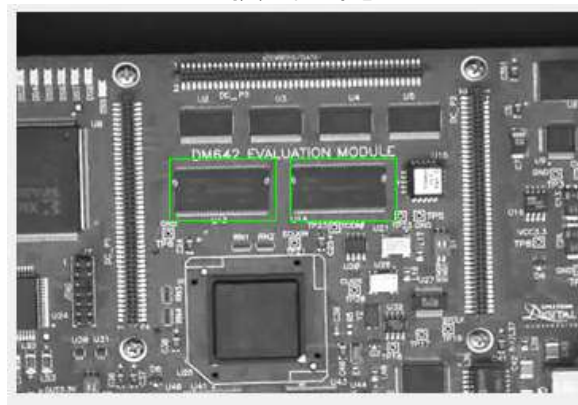


# 图像处理

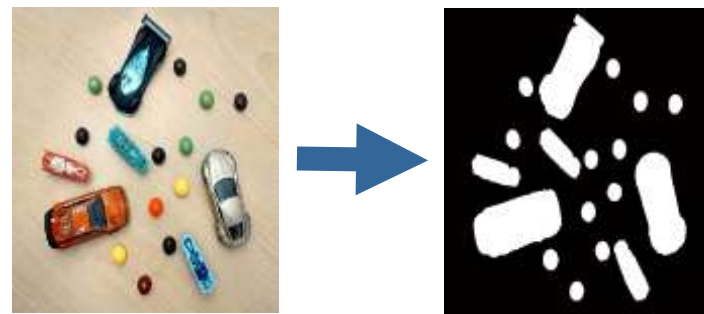
多目标跟踪



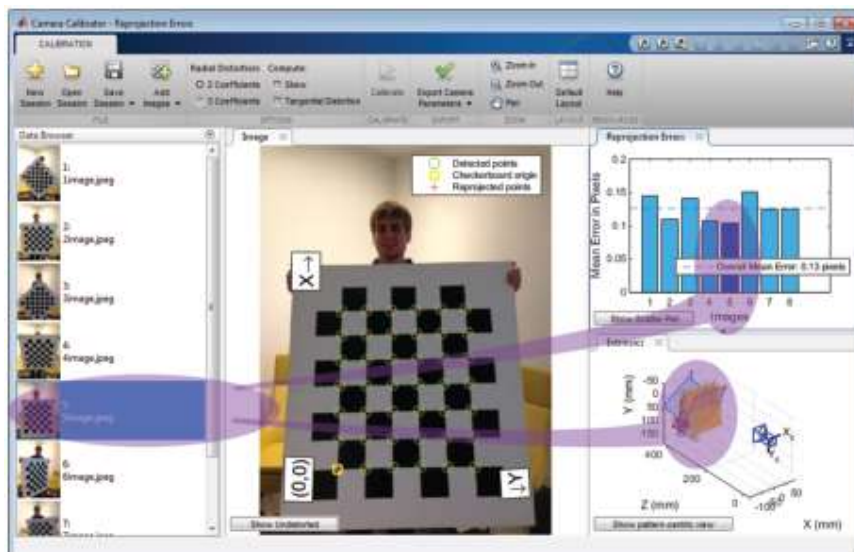
模式匹配



物体分割



相机校准APP



人脸识别与跟踪

