Latest Features in Fixed-Point Designer

October 2014

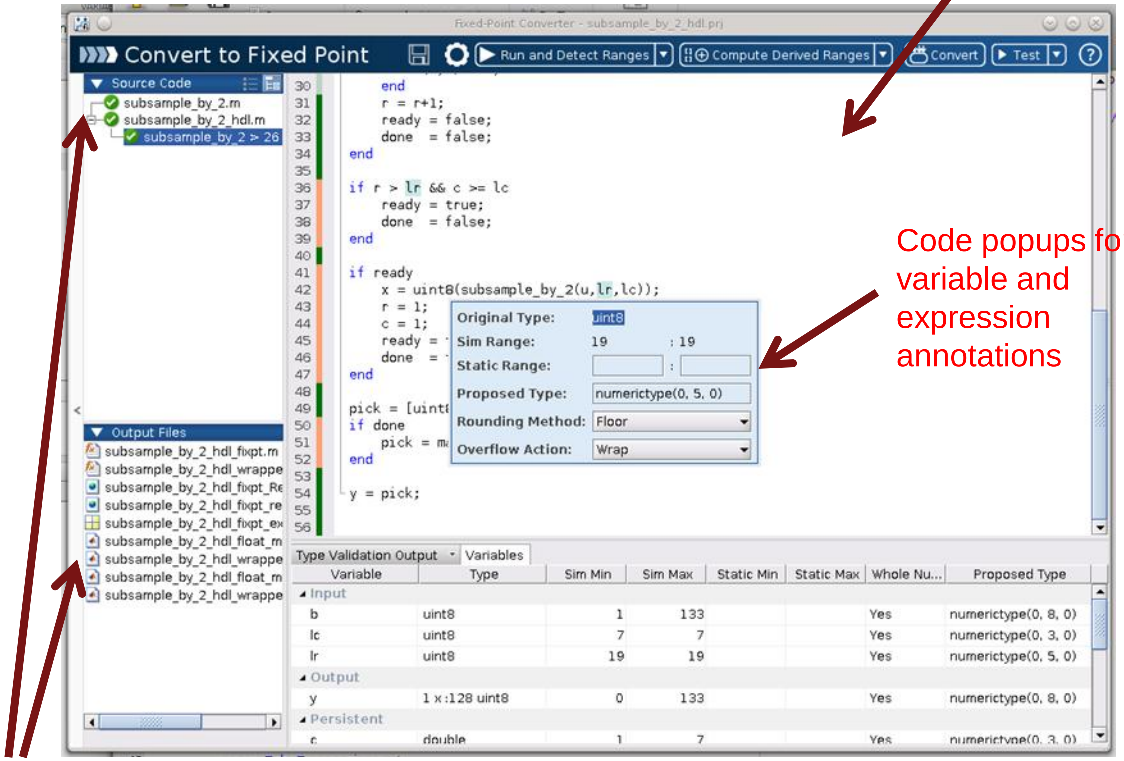
R2014b

R2014b

Fixed-Point Converter App for Automated Conversion of Floating-Point MATLAB Code

Standalone UI enables automatic conversion of MATLAB code to fixed point

- Run test benches and/or code snippets to autodefine input types or manually specify input types.
- Iteratively refine numeric types with simulations and derived ranges before building and testing the converted code.
- Works outside of MATLAB and HDL Coder workflows


Fixed-Point Converter - subsample_by_2_hdl.prj
Convert to Fixed Point
Run and Detect Ranges
Compute Derived Ranges
Convert
Test
Source Code
subsample_by_2.m
subsample_by_2_hdl.m
subsample_by_2 > 26
Output Files
subsample_by_2_hdl_fixpt.m
subsample_by_2_hdl_wrappe
subsample_by_2_hdl_fixpt_Re
subsample_by_2_hdl_fixpt_re
subsample_by_2_hdl_fixpt_ex
subsample_by_2_hdl_float_m
subsample_by_2_hdl_wrappe
subsample_by_2_hdl_float_m
subsample_by_2_hdl_wrappe
Type Validation Output
Variables

```
        end
        r = r+1;
        ready = false;
        done  = false;
    end

    if r > lr && c >= lc
        ready = true;
        done  = false;
    end

    if ready
        x = uint8(subsample_by_2(u,lr,lc));
        r = 1;
        c = 1;
        ready = 
        done  = 
    end

    pick = [uint
    if done
        pick = m
    end

    y = pick;
```

Original Type: uint8
Sim Range: 19 : 19
Static Range: :
Proposed Type: numerictype(0, 5, 0)
Rounding Method: Floor
Overflow Action: Wrap

Variable	Type	Sim Min	Sim Max	Static Min	Static Max	Whole Nu...	Proposed Type
Input							
b	uint8	1	133			Yes	numerictype(0, 8, 0)
lc	uint8	7	7			Yes	numerictype(0, 3, 0)
lr	uint8	19	19			Yes	numerictype(0, 5, 0)
Output							
y	1 x :128 uint8	0	133			Yes	numerictype(0, 8, 0)
Persistent							
c	double	1	7			Yes	numerictype(0, 3, 0)

Live editor for easy design modification

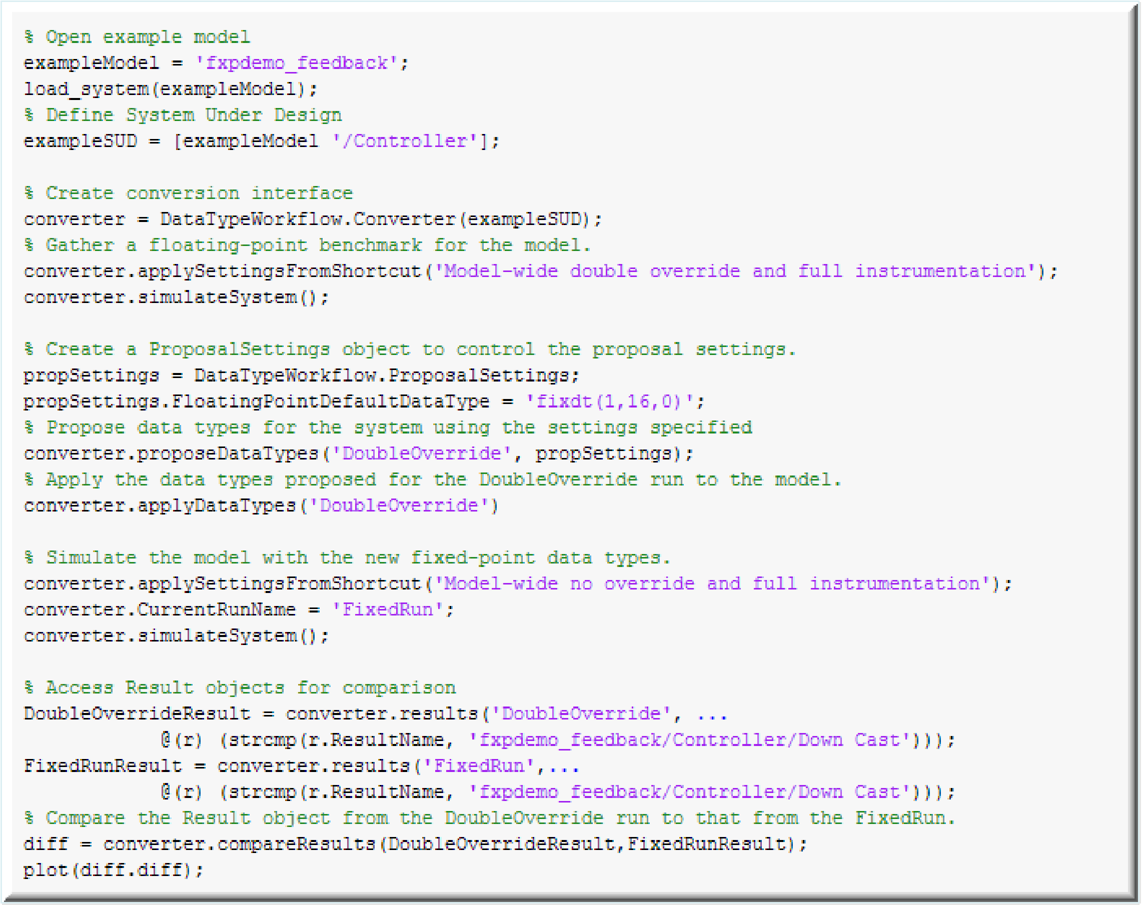
Code popups for variable and expression annotations

Integrated editor for simultaneously viewing source files and generated artifacts

Commands for Scripting Fixed-Point Conversion and Accessing the Collected Data in Simulink

Command-line API for model data-type conversion

- Enable scripting workflow steps with data from simulation or range analysis.
- Enable streamlining fixed-point conversion for large scale models through automated scripts
- Command-line access to range and data-type information for analysis and reporting



```
% Open example model
exampleModel = 'fxpdemo_feedback';
load_system(exampleModel);
% Define System Under Design
exampleSUD = [exampleModel '/Controller'];

% Create conversion interface
converter = DataTypeWorkflow.Converter(exampleSUD);
% Gather a floating-point benchmark for the model.
converter.applySettingsFromShortcut('Model-wide double override and full instrumentation');
converter.simulateSystem();

% Create a ProposalSettings object to control the proposal settings.
propSettings = DataTypeWorkflow.ProposalSettings;
propSettings.FloatingPointDefaultDataType = 'fixdt(1,16,0)';
% Propose data types for the system using the settings specified
converter.proposeDataTypes('DoubleOverride', propSettings);
% Apply the data types proposed for the DoubleOverride run to the model.
converter.applyDataTypes('DoubleOverride')

% Simulate the model with the new fixed-point data types.
converter.applySettingsFromShortcut('Model-wide no override and full instrumentation');
converter.CurrentRunName = 'FixedRun';
converter.simulateSystem();

% Access Result objects for comparison
DoubleOverrideResult = converter.results('DoubleOverride', ...
            @(r) (strcmp(r.ResultName, 'fxpdemo_feedback/Controller/Down Cast')));
FixedRunResult = converter.results('FixedRun',...
            @(r) (strcmp(r.ResultName, 'fxpdemo_feedback/Controller/Down Cast')));
% Compare the Result object from the DoubleOverride run to that from the FixedRun.
diff = converter.compareResults(DoubleOverrideResult,FixedRunResult);
plot(diff.diff);
```

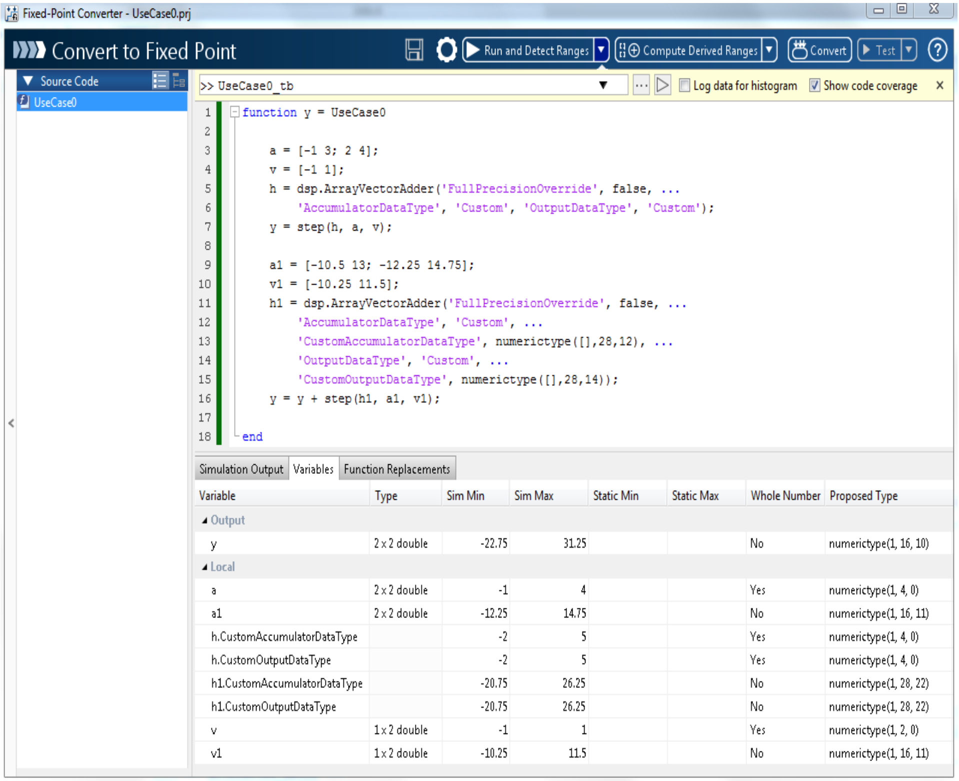
```
» DataTypeWorkflow.Converter(gcs)
```

Automated Fixed-Point Conversion for Commonly Used DSP System objects

Propose and apply fixed-point data types for some System objects based on simulation range data

Enable conversion of following DSP System Toolbox™ System objects to fixed point using the Fixed-Point Converter app:

- `dsp.BiquadFilter`
- `dsp.FIRFilter`, direct form only
- `dsp.FIRRateConverter`
- `dsp.LowerTriangularSolver`
- `dsp.UpperTriangularSolver`
- `dsp.ArrayVectorAdder`

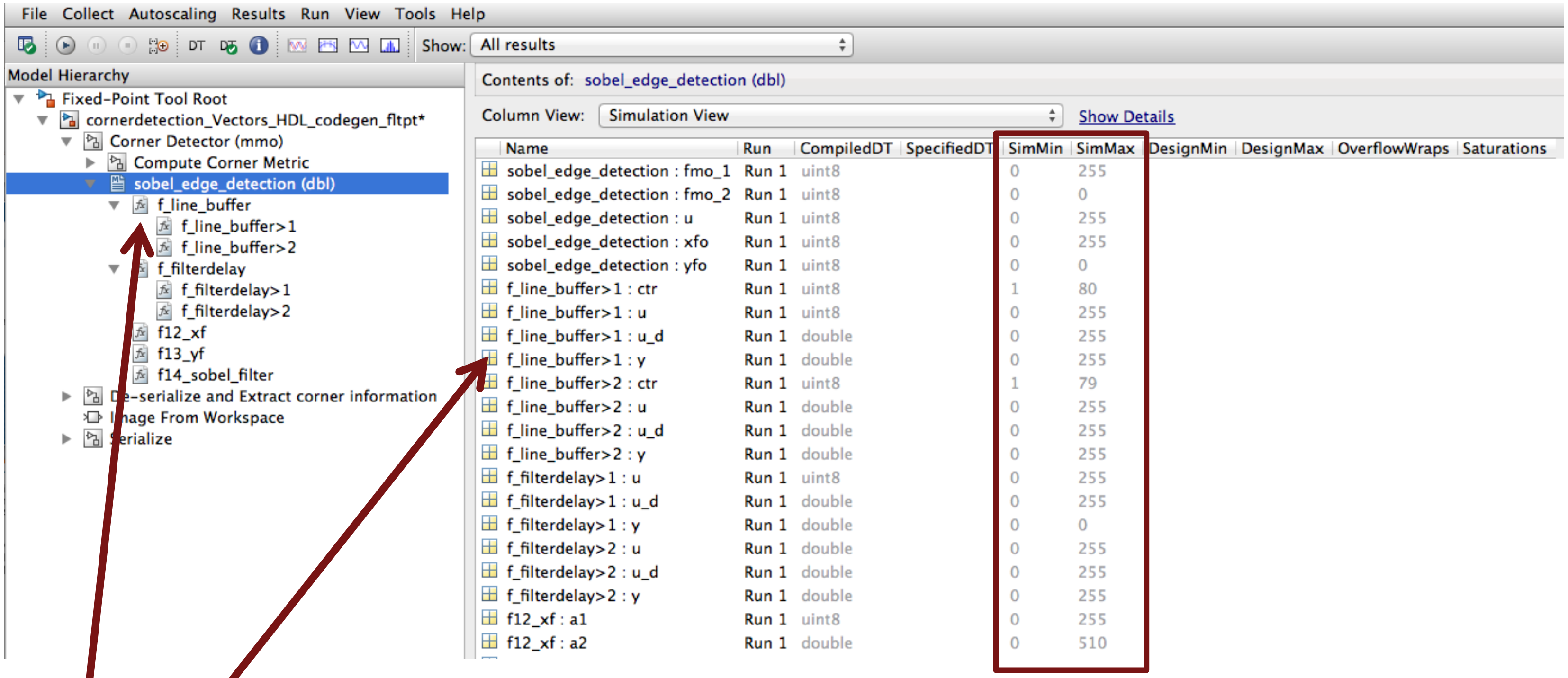


```
function y = UseCase0

    a = [-1 3; 2 4];
    v = [-1 1];
    h = dsp.ArrayVectorAdder('FullPrecisionOverride', false, ...
        'AccumulatorDataType', 'Custom', 'OutputDataType', 'Custom');
    y = step(h, a, v);

    a1 = [-10.5 13; -12.25 14.75];
    v1 = [-10.25 11.5];
    h1 = dsp.ArrayVectorAdder('FullPrecisionOverride', false, ...
        'AccumulatorDataType', 'Custom', ...
        'CustomAccumulatorDataType', numerictype([],28,12), ...
        'OutputDataType', 'Custom', ...
        'CustomOutputDataType', numerictype([],28,14));
    y = y + step(h1, a1, v1);

end
```

Variable	Type	Sim Min	Sim Max	Static Min	Static Max	Whole Number	Proposed Type
◢ Output							
y	2 x 2 double	-22.75	31.25			No	numerictype(1, 16, 10)
◢ Local							
a	2 x 2 double	-1	4			Yes	numerictype(1, 4, 0)
a1	2 x 2 double	-12.25	14.75			No	numerictype(1, 16, 11)
h.CustomAccumulatorDataType		-2	5			Yes	numerictype(1, 4, 0)
h.CustomOutputDataType		-2	5			Yes	numerictype(1, 4, 0)
h1.CustomAccumulatorDataType		-20.75	26.25			No	numerictype(1, 28, 22)
h1.CustomOutputDataType		-20.75	26.25			No	numerictype(1, 28, 22)
v	1 x 2 double	-1	1			Yes	numerictype(1, 2, 0)
v1	1 x 2 double	-10.25	11.5			No	numerictype(1, 16, 11)

Simulation Range Collection for MATLAB Function Blocks in Simulink

Visualize simulation ranges of named variables of a MATLAB function block within the Fixed-Point Tool



Show: All results

Contents of: sobel_edge_detection (dbl)

Column View: Simulation View

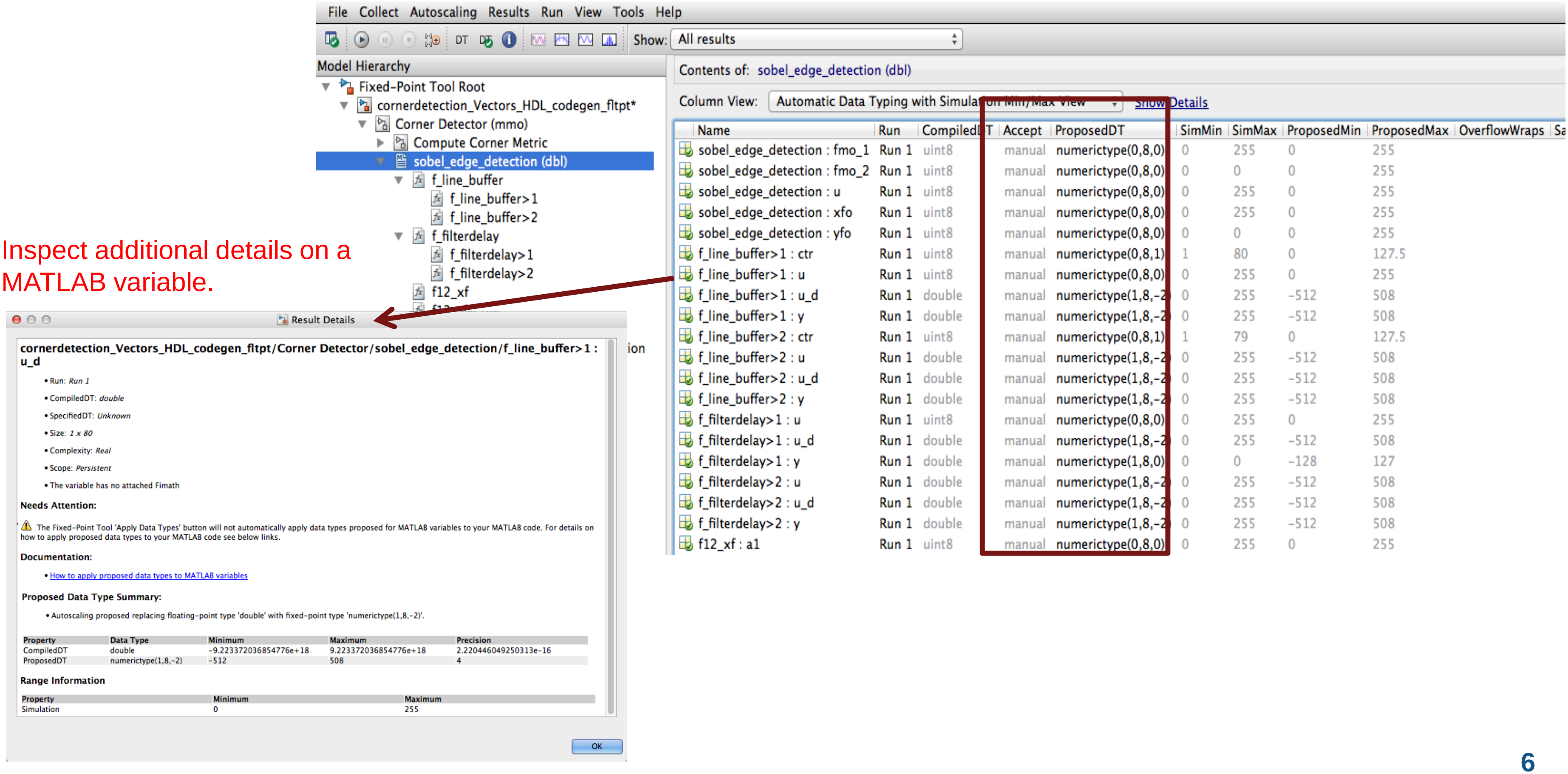
Name	Run	CompiledDT	SpecifiedDT	SimMin	SimMax	DesignMin	DesignMax	OverflowWraps	Saturations
sobel_edge_detection : fmo_1	Run 1	uint8		0	255				
sobel_edge_detection : fmo_2	Run 1	uint8		0	0				
sobel_edge_detection : u	Run 1	uint8		0	255				
sobel_edge_detection : xfo	Run 1	uint8		0	255				
sobel_edge_detection : yfo	Run 1	uint8		0	0				
f_line_buffer>1 : ctr	Run 1	uint8		1	80				
f_line_buffer>1 : u	Run 1	uint8		0	255				
f_line_buffer>1 : u_d	Run 1	double		0	255				
f_line_buffer>1 : y	Run 1	double		0	255				
f_line_buffer>2 : ctr	Run 1	uint8		1	79				
f_line_buffer>2 : u	Run 1	double		0	255				
f_line_buffer>2 : u_d	Run 1	double		0	255				
f_line_buffer>2 : y	Run 1	double		0	255				
f_filterdelay>1 : u	Run 1	uint8		0	255				
f_filterdelay>1 : u_d	Run 1	double		0	255				
f_filterdelay>1 : y	Run 1	double		0	0				
f_filterdelay>2 : u	Run 1	double		0	255				
f_filterdelay>2 : u_d	Run 1	double		0	255				
f_filterdelay>2 : y	Run 1	double		0	255				
f12_xf : a1	Run 1	uint8		0	255				
f12_xf : a2	Run 1	double		0	510				

Explore functions and variables from the Fixed-Point Tool tree.

Data Type Proposals for MATLAB Function Blocks in Simulink

R2014b

Propose fixed-point data types for MATLAB variables from the Fixed-Point Tool



Show: All results

Model Hierarchy

- Fixed-Point Tool Root
  - cornerdetection_Vectors_HDL_codegen_fltpt*
    - Corner Detector (mmo)
      - Compute Corner Metric
      - sobel_edge_detection (dbl)
        - f_line_buffer
          - f_line_buffer>1
          - f_line_buffer>2
        - f_filterdelay
          - f_filterdelay>1
          - f_filterdelay>2
        - f12_xf

Contents of: sobel_edge_detection (dbl)

Column View: Automatic Data Typing with Simulation Min/Max View

Show Details

Name	Run	CompiledDT	Accept	ProposedDT	SimMin	SimMax	ProposedMin	ProposedMax	OverflowWraps	Sa
sobel_edge_detection : fmo_1	Run 1	uint8	manual	numerictype(0,8,0)	0	255	0	255		
sobel_edge_detection : fmo_2	Run 1	uint8	manual	numerictype(0,8,0)	0	0	0	255		
sobel_edge_detection : u	Run 1	uint8	manual	numerictype(0,8,0)	0	255	0	255		
sobel_edge_detection : xfo	Run 1	uint8	manual	numerictype(0,8,0)	0	255	0	255		
sobel_edge_detection : yfo	Run 1	uint8	manual	numerictype(0,8,0)	0	0	0	255		
f_line_buffer>1 : ctr	Run 1	uint8	manual	numerictype(0,8,1)	1	80	0	127.5		
f_line_buffer>1 : u	Run 1	uint8	manual	numerictype(0,8,0)	0	255	0	255		
f_line_buffer>1 : u_d	Run 1	double	manual	numerictype(1,8,-2	0	255	-512	508		
f_line_buffer>1 : y	Run 1	double	manual	numerictype(1,8,-2	0	255	-512	508		
f_line_buffer>2 : ctr	Run 1	uint8	manual	numerictype(0,8,1)	1	79	0	127.5		
f_line_buffer>2 : u	Run 1	double	manual	numerictype(1,8,-2	0	255	-512	508		
f_line_buffer>2 : u_d	Run 1	double	manual	numerictype(1,8,-2	0	255	-512	508		
f_line_buffer>2 : y	Run 1	double	manual	numerictype(1,8,-2	0	255	-512	508		
f_filterdelay>1 : u	Run 1	uint8	manual	numerictype(0,8,0)	0	255	0	255		
f_filterdelay>1 : u_d	Run 1	double	manual	numerictype(1,8,-2	0	255	-512	508		
f_filterdelay>1 : y	Run 1	double	manual	numerictype(1,8,0)	0	0	-128	127		
f_filterdelay>2 : u	Run 1	double	manual	numerictype(1,8,-2	0	255	-512	508		
f_filterdelay>2 : u_d	Run 1	double	manual	numerictype(1,8,-2	0	255	-512	508		
f_filterdelay>2 : y	Run 1	double	manual	numerictype(1,8,-2	0	255	-512	508		
f12_xf : a1	Run 1	uint8	manual	numerictype(0,8,0)	0	255	0	255		

Inspect additional details on a MATLAB variable.

Result Details

cornerdetection_Vectors_HDL_codegen_fltpt/Corner Detector/sobel_edge_detection/f_line_buffer>1 : u_d

- Run: *Run 1*
- CompiledDT: *double*
- SpecifiedDT: *Unknown*
- Size: *1 x 80*
- Complexity: *Real*
- Scope: *Persistent*
- The variable has no attached Fimath

Needs Attention:

⚠ The Fixed-Point Tool 'Apply Data Types' button will not automatically apply data types proposed for MATLAB variables to your MATLAB code. For details on how to apply proposed data types to your MATLAB code see below links.

Documentation:

- How to apply proposed data types to MATLAB variables

Proposed Data Type Summary:

- Autoscaling proposed replacing floating-point type 'double' with fixed-point type 'numerictype(1,8,-2)'.

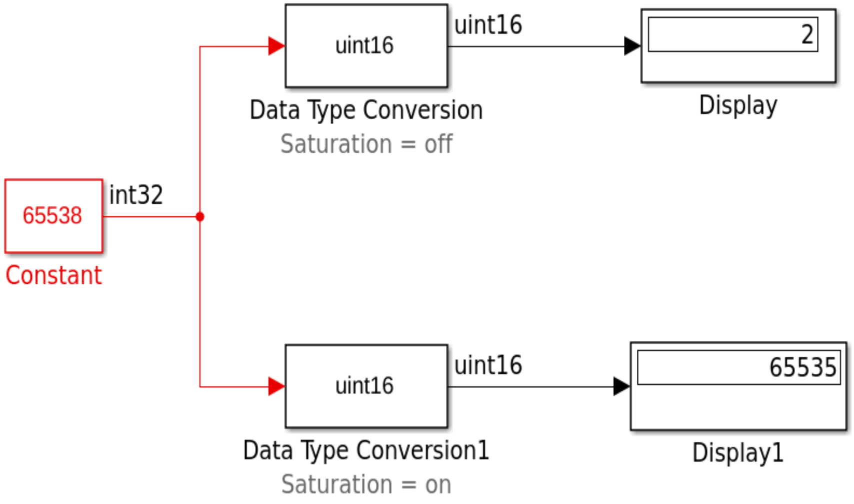
Property	Data Type	Minimum	Maximum	Precision
CompiledDT	double	-9.223372036854776e+18	9.223372036854776e+18	2.220446049250313e-16
ProposedDT	numerictype(1,8,-2)	-512	508	4

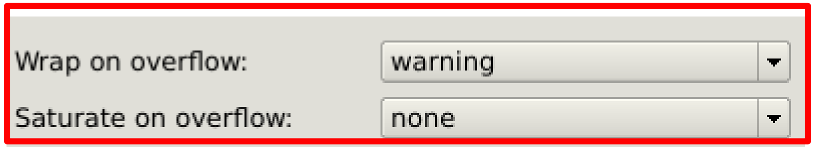
Range Information

Property	Minimum	Maximum
Simulation	0	255

Overflow Diagnostics to Distinguish Between Wrap and Saturation in Simulink

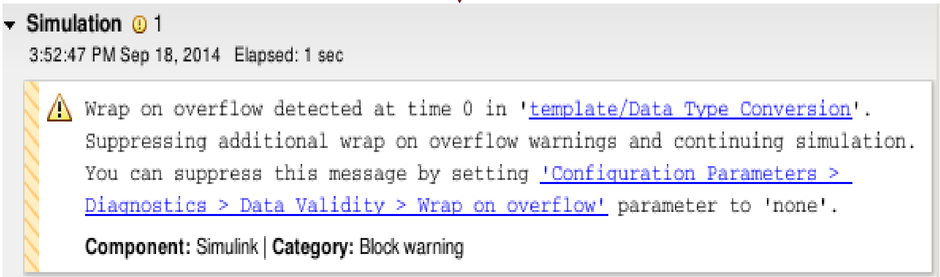
Separately control the diagnostics for overflows that wrap and overflows that saturate by setting each diagnostic to error, warning, or none





Wrap on overflow: warning

Saturate on overflow: none

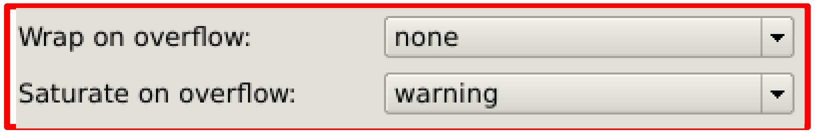


Simulation ⚠ 1

3:52:47 PM Sep 18, 2014 Elapsed: 1 sec

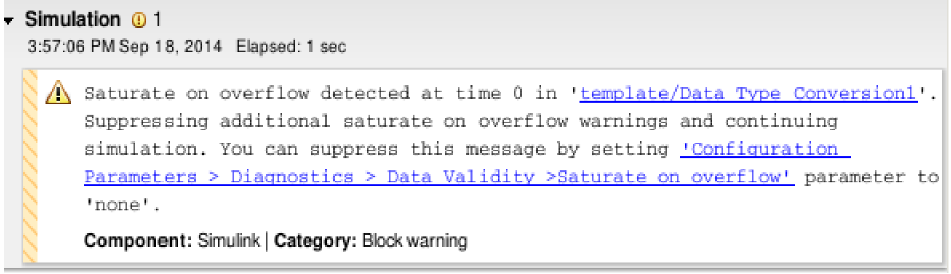
⚠ Wrap on overflow detected at time 0 in 'template/Data Type Conversion'. Suppressing additional wrap on overflow warnings and continuing simulation. You can suppress this message by setting 'Configuration Parameters > Diagnostics > Data Validity > Wrap on overflow' parameter to 'none'.

Component: Simulink | **Category:** Block warning



Wrap on overflow: none

Saturate on overflow: warning



Simulation ⚠ 1

3:57:06 PM Sep 18, 2014 Elapsed: 1 sec

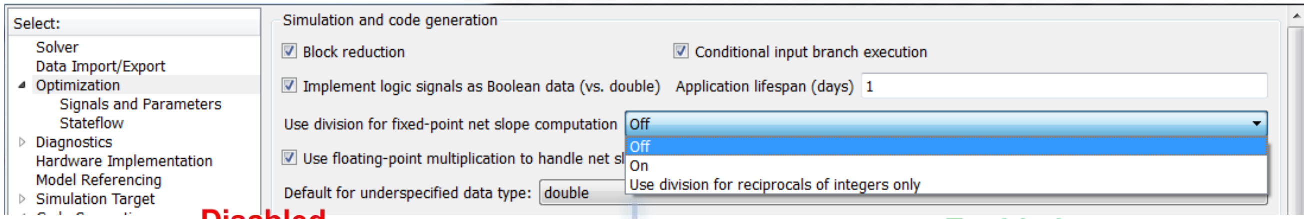
⚠ Saturate on overflow detected at time 0 in 'template/Data Type Conversion1'. Suppressing additional saturate on overflow warnings and continuing simulation. You can suppress this message by setting 'Configuration Parameters > Diagnostics > Data Validity >Saturate on overflow' parameter to 'none'.

Component: Simulink | **Category:** Block warning

MathWorks® R2014b

Cast Net Slope Computations Using Rational Numbers

Enable data type conversion using rational approximation for more accurate results and easier to read code


Select:
- Solver
- Data Import/Export
- Optimization
  - Signals and Parameters
  - Stateflow
- Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target

Simulation and code generation

☑ Block reduction

☑ Conditional input branch execution

☑ Implement logic signals as Boolean data (vs. double)

Application lifespan (days) 1

Use division for fixed-point net slope computation: Off
- Off
- On
- Use division for reciprocals of integers only

☑ Use floating-point multiplication to handle net sl

Default for underspecified data type: double

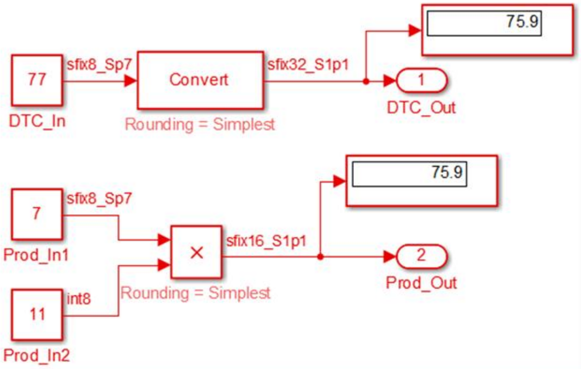
Disabled

Use division for fixed-point net slope computation: Off

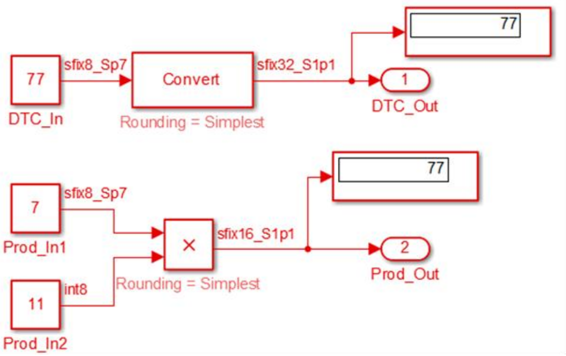
Enabled

Use division for fixed-point net slope computation: On

Simulation



More Accurate Results



Code Generation

Optimized for Speed Using Shift and Multiply

```
DTC_Out = DTC_In * 81 >> 7;
Prod_Out = (int16_T)((Prod_In1 * Prod_In2 >> 1) * 5213 >> 12);
```

Easier to Read

```
DTC_Out = DTC_In * 7 / 11;
Prod_Out = (int16_T)(Prod_In1 * Prod_In2 * 7 / 11);
```