Latest Features in Fixed-Point Designer

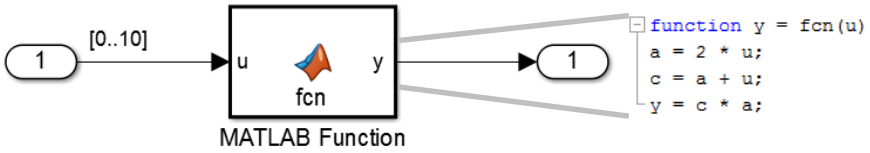
March 2015

R2015a

Derived Ranges for MATLAB Function Blocks in Simulink

Improved MATLAB Function Block Support for Fixed-Point Tool range analysis

- The Fixed-Point Tool uses design ranges to derive ranges for MATLAB variables in a MATLAB Function block.
- The tool can also propose data types for the variables based on the derived range data.



Name	DerivedMin	DerivedMax
In1	0	10
MATLAB Function.y	0	600
Out1	0	600
MATLAB Function/fcn : u	0	10
MATLAB Function/fcn : a	0	20
MATLAB Function/fcn : c	0	30
MATLAB Function/fcn : y	0	600

Output of block and MATLAB variables are derived and displayed

```
» ex_range_matlab_function_block
```

Automated Fixed-Point Conversion for Additional DSP System objects

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

You can now convert the following DSP System Toolbox System objects to fixed-point using the Fixed-Point Converter app:

- `dsp.FIRDecimator`
- `dsp.FIRInterpolator`
- `dsp.FIRFilter`
- `dsp.LUFactor`
- `dsp.VariableFractionalDelay`
- `dsp.Window`

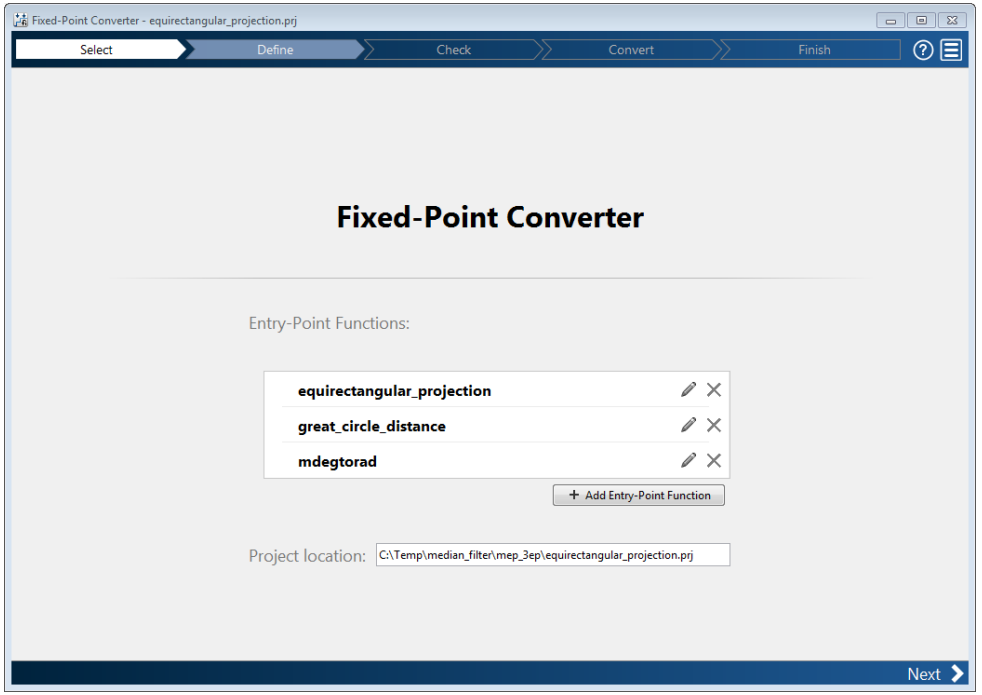
Fixed-Point Converter App Enhancements for MATLAB Fixed-Point Users

R2015a

Support for Projects with Multiple Entry-Point Functions

Generate fixed-point code for multiple entry point functions

- Specify multiple entry-point functions in a Fixed-Point Converter app project.
- Generate fixed-point C/C++ libraries using MATLAB Coder.
- Perform conversion with multiple entry-point functions, which facilitates integration with larger applications.

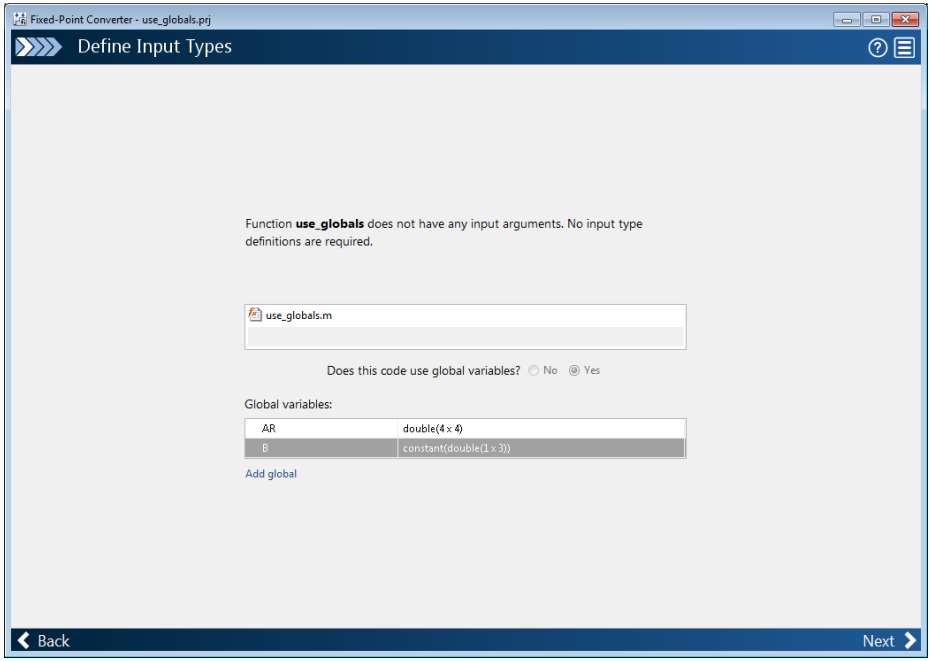


Support for Global Variables

Specify global variables in the Fixed-Point Converter app workflow

- Algorithms containing global variables can be converted without modifying your code.
- Ranges for globals are synchronized across functions.
- Constant globals used instead of passing constants to functions.
- Synchronize globals between the testbench and generated fixed-point code during numerical verification.

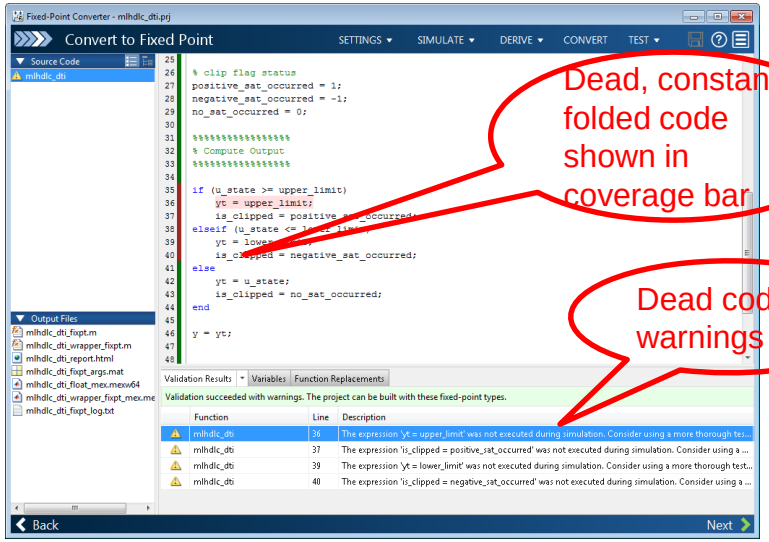
```
function y = use_globals()
%#codegen
% Turn off inlining to make
% generated code easier to read
coder.inline('never');
% Declare AR and B as global variables
global AR;
global B;
AR(1) = B(1);
y = AR * 2;
```



Smart Conversion of Dead and Constant Folded Code

Fixed-Point Converter app detects constant folded and dead code to reduce translation errors

- Augments test files to exercise the algorithm adequately.
- Inline comments in fixed-point MATLAB code to mark dead and untranslated regions.
- Displays code execution information in generated conversion report and as color-code bars in editor view.
- Supports command-line workflow.



Generated on 2015-02-24 18:54:50

The following table shows fixed point instrumentation results

Fixed-Point Report *mlhdlc_dti*

Simulation Coverage	Code
100%	function [y, is_clipped] = mlhdlc_dti(u_in, init_val, gain_val, upper_limit, lower_limit) % Discrete Time Integrator in MATLAB Function block % % Forward Euler method, also known as Forward Rectangular, % or left-hand approximation. The resulting expression for the % output of the block at step n is % % y(n) = y(n-1) + K * u(n-1) % %%%%%%%%%%% % Setup %%%%%%%%%%% % numeric type to clip the accumulator value after each addition % variable to hold state between consecutive calls to this block persistent u_state;
Once	if isempty(u_state) u_state = init_val; end

100%	% clip flag status
	positive_sat_occurred = 1;
	negative_sat_occurred = -1;
	no_sat_occurred = 0;

Code execution serialized to html report

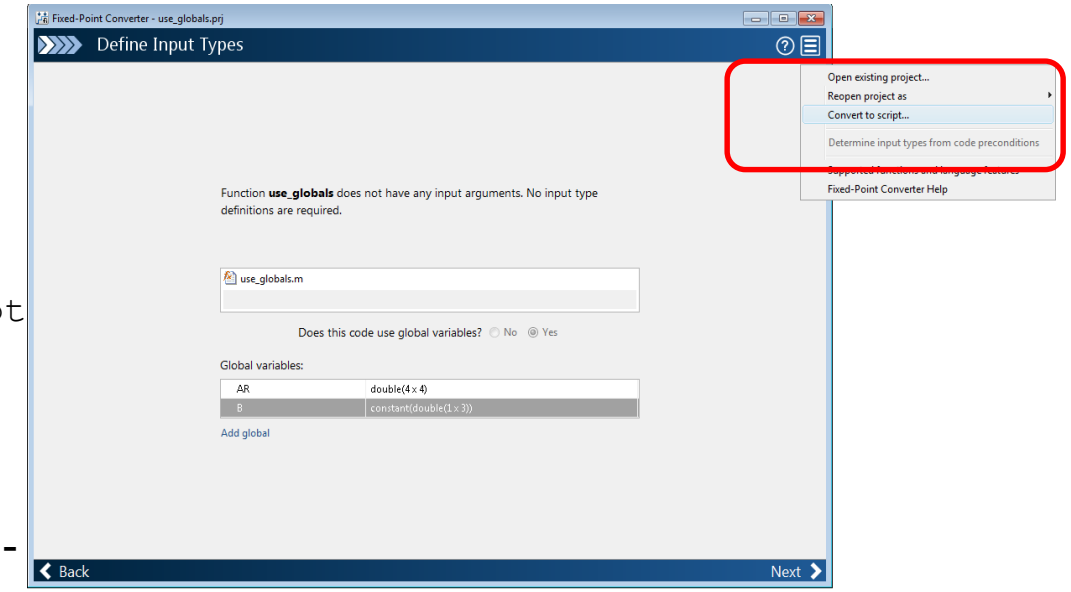
Project to Script Conversion

Convert fixed-point conversion project to equivalent MATLAB code in a MATLAB script

- Use *-tocode* option of the fixedPointConverter command.

```
>> fixedPointConverter -script
myScript -tocode myProject
```

- Use the script to repeat the project workflow in a command-line workflow for easy scripting.



Generated Fixed-Point Code Enhancements

Improving readability of generated fixed-point MATLAB code

The generated fixed-point MATLAB code now:

- Uses colon syntax for multi-output assignments, reducing the number of fi casts in the generated fixed-point code
- Preserves the indentation and formatting of your original algorithm, improving the readability of the generated fixed-point code

Floating Point MATLAB Code

```
function [g1, g2, g3] = gaussian_filter(gh, gv)

%g=fspecial('gaussian',[5 5],1.5);
g = [0.0144    0.0281    0.0351    0.0281    0.0144
     0.0281    0.0547    0.0683    0.0547    0.0281
     0.0351    0.0683    0.0853    0.0683    0.0351
     0.0281    0.0547    0.0683    0.0547    0.0281
     0.0144    0.0281    0.0351    0.0281    0.0144];

g1 = (gh .* gh) .* g(:)';
g2 = (gh .* gv) .* g(:)';
g3 = (gv .* gv) .* g(:)';

end
```

Fixed-Point MATLAB Code

```
function [g1, g2, g3] = gaussian_filter(gh, gv)

%g=fspecial('gaussian',[5 5],1.5);
fm = fimath('RoundingMethod', 'Floor', 'OverflowAction', 'Wrap', 'Prod

g = fi([0.0144    0.0281    0.0351    0.0281    0.0144
     0.0281    0.0547    0.0683    0.0547    0.0281
     0.0351    0.0683    0.0853    0.0683    0.0351
     0.0281    0.0547    0.0683    0.0547    0.0281
     0.0144    0.0281    0.0351    0.0281    0.0144], 0, 16, 19, fm);

g1 = fi((gh .* gh) .* g(:)', 0, 16, 15, fm);
g2 = fi((gh .* gv) .* g(:)', 1, 16, 15, fm);
g3 = fi((gv .* gv) .* g(:)', 0, 16, 15, fm);

end
```

Floating Point MATLAB Code

```
[y,z,p] = original_fir_filter(b,x,z,p);
```

Fixed-Point MATLAB Code

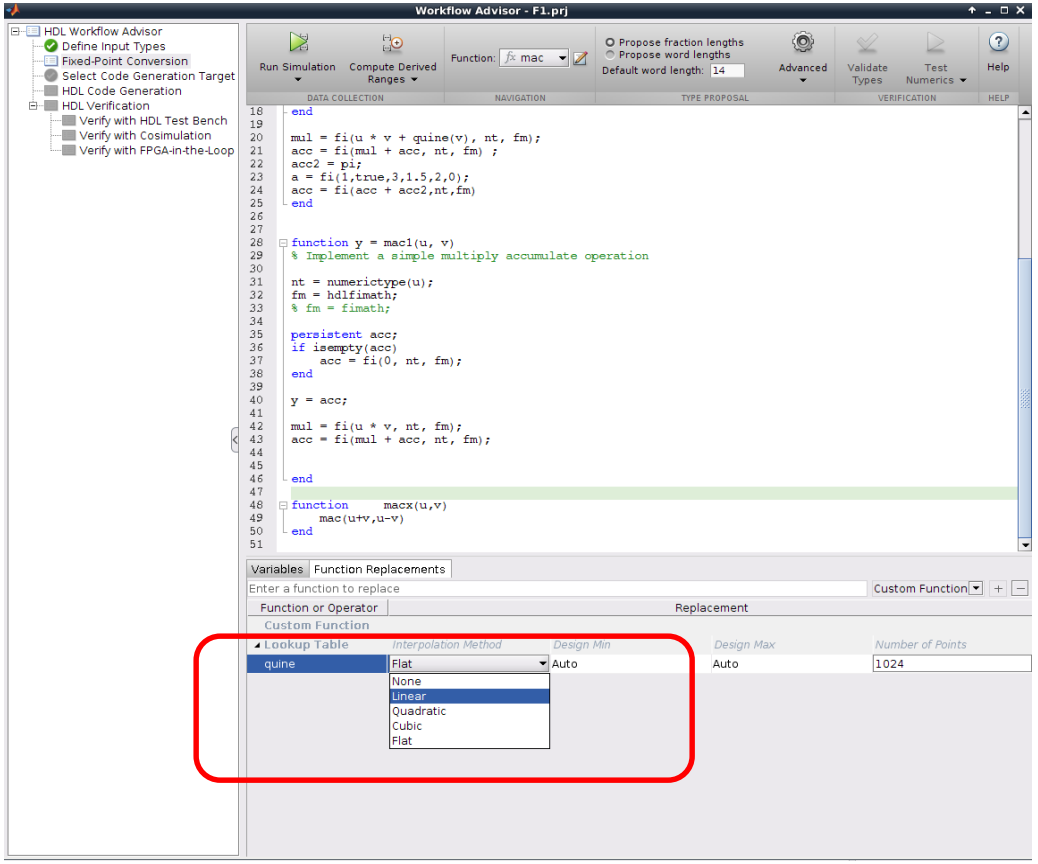
```
[y(:),z(:),p(:)] = original_fir_filter(b,x,z,p);
```

Flat Mode for Lookup Tables

New interpolation method for generating lookup table MATLAB function replacements in generated C code

You can now:

- Look up tables without index calculations
- Allows for faster code by discarding the prelookup step, and it is hardware efficient by reducing the use of multipliers in the data path
- Find it from both the command-line workflow and the Function Replacements tab of the Fixed-Point Converter app



Seamless Transition to MATLAB Coder

Integration with MATLAB Coder app interface

Smoother conversion process from floating-point MATLAB code to fixed-point C/C++ code