

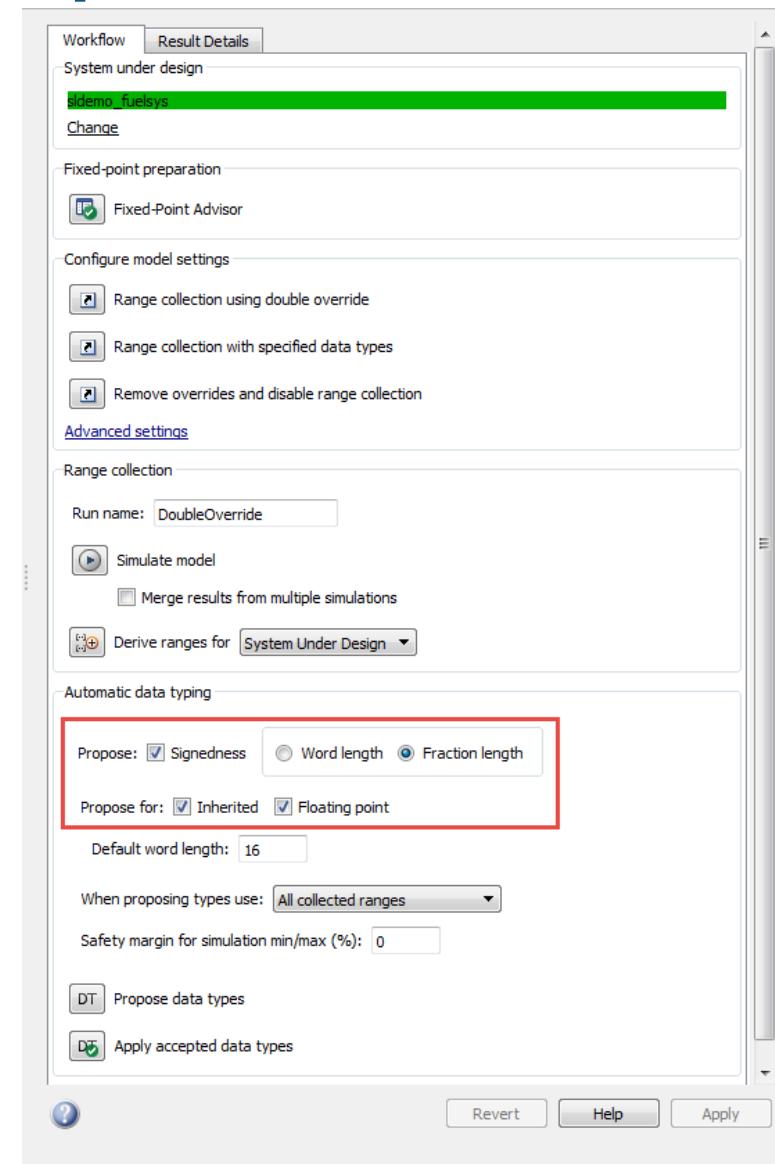
Latest Features in Fixed-Point Designer

September 2015

Simulink Fixed-Point Tool workflow simplification

Propose signedness and data types for inherited and floating-point types

- Propose signedness for blocks in the system under design
- Propose fixed-point data types for objects that use floating-point or inherited data types
- Two-way traceability between Simulink blocks and corresponding results in the Fixed-Point Tool



Double-precision to single-precision conversion

Convert double-precision MATLAB code to single-precision MATLAB code using the command line

- `convertToSingle` function to convert double-precision MATLAB code to single-precision MATLAB code
- Verify single-precision version without modifying original algorithm
- Using MATLAB Coder, generate single-precision C code using `-singleC` option of `codegen` command

```
%% convertToSingle command
% Create a SingleConfig via coder.config
singleCfg = coder.config('single');

% Configure SingleConfig
singleCfg.TestBenchName = 'test_heart_rate_detector';
singleCfg.TestNumerics = true;
singleCfg.LogIOForComparisonPlotting = true;

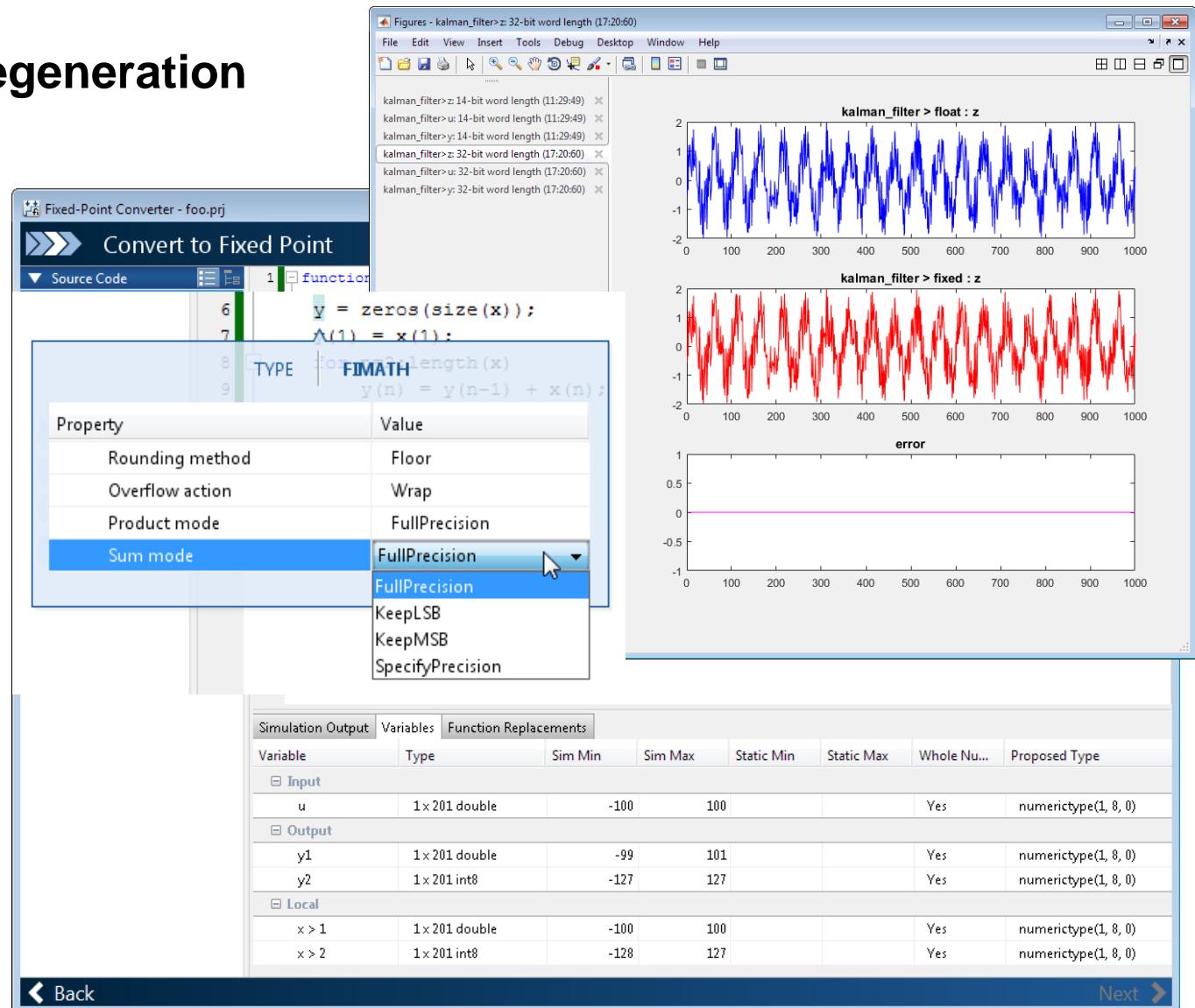
% Pass it to convertToSingle command
convertToSingle -args {0, true} heart_rate_detector -config singleCfg

%% -singleC codegen option
% Specify -singleC to codegen along with other options
libCfg = coder.config('lib');
libCfg.TargetLangStandard = 'C99 (ISO)';
codegen -args {0, true} heart_rate_detector -report -singleC -config libCfg
```

MATLAB Fixed-Point Converter app streamlined workflow

Restore project state and minimize regeneration of MEX files

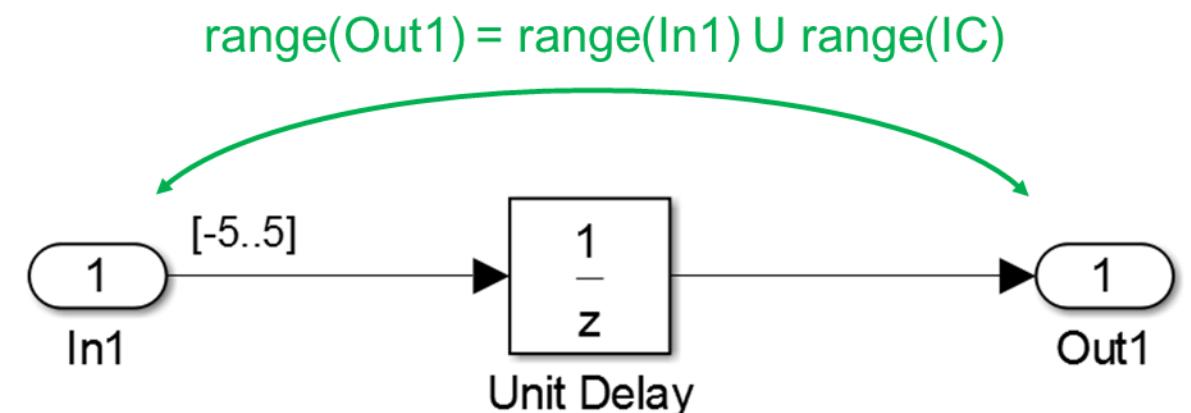
- Save state of project between sessions
- Rebuild MEX files only when required by changes in the code
- Control `fimath` properties within Fixed-Point Converter app editor
- Improved management of comparison plots by docking generated test plots into separate tabs of one figure window
- View variable specializations in the Variables table of the app



Range analysis for Delay blocks

Improve accuracy and speed of range analysis on models using Delay blocks

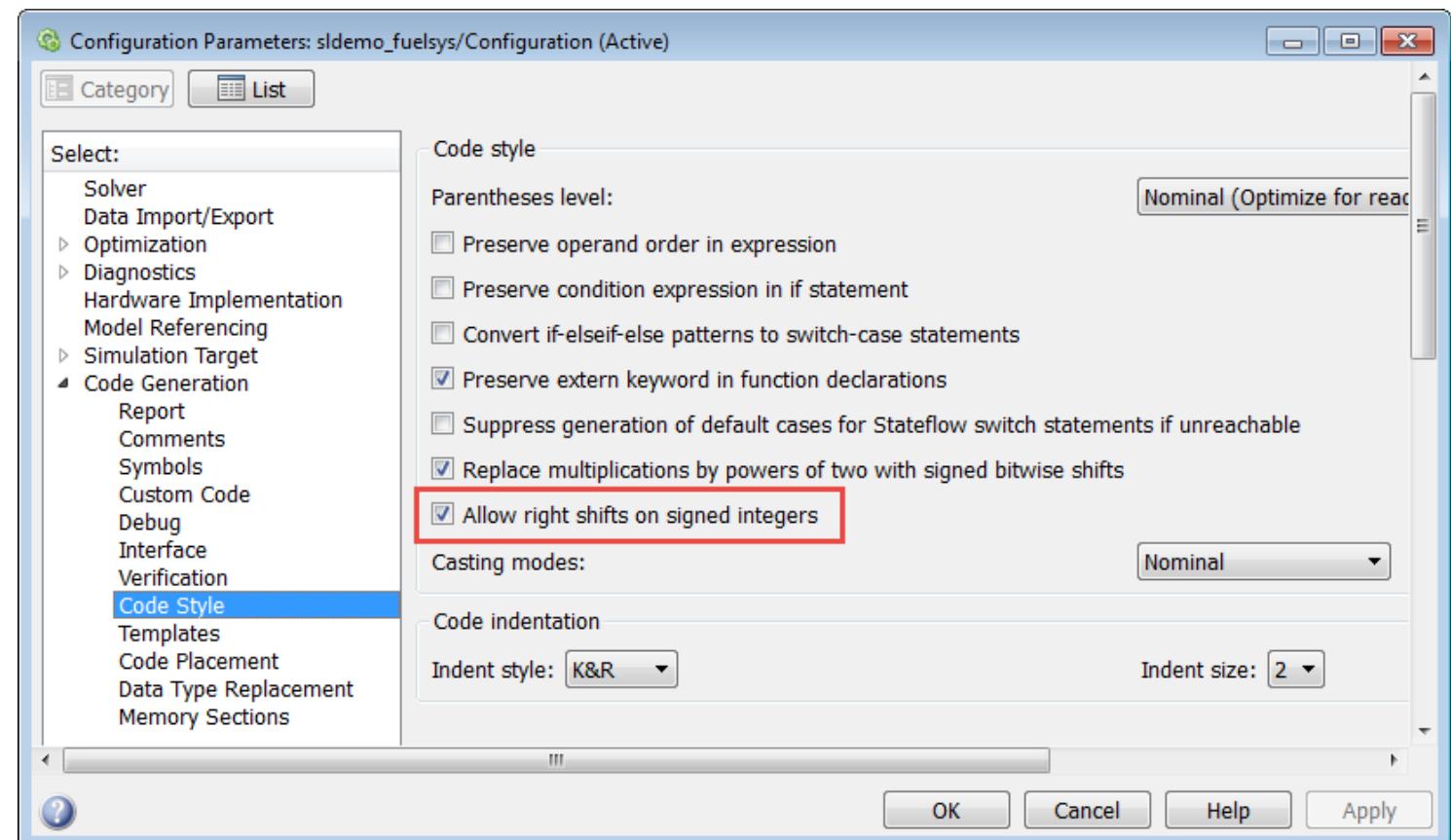
- Derive ranges with greater precision for models that use Delay blocks
- Precise output range for all Delay blocks with bounded input range
- Greater theoretical accuracy and speed in deriving ranges for certain configurations of cascading Delay blocks



Control of signed shifts in fixed-point scaling operations

Control the use of signed shifts in generated code

- Replace signed shifts with a function call that performs the operation without the use of signed shifts
- Assist in compliance with certain coding standards (e.g., MISRA)



Full-precision value property for fi object

Easy access to exact values in decimal and "to string" methods for fi object

- Set and get full-precision fixed-point values in "real-world value"
- Use `tostring` for `fi` objects to enable cut-and-paste into MATLAB function
- Use `mat2str` "matrix to string" for `fi` objects to display full precision without going through `double`

```
Command Window

>> X = fi([],1,128,127);
>> X.Value = '[0.25, 0.1]'

X =
0.2500    0.1000

    DataTypeMode: Fixed-point: binary point scaling
    Signedness: Signed
    WordLength: 128
    FractionLength: 127
>> X.Value

ans =
[0.25 0.100000000000000000000000000000000000000000000000000000000000000011754943508222875079687365372222456778186655567720875215

>> tostring(X)

ans =
fi('numerictype',numerictype(1,128,127),'Value','[0.25 0.100000000000000000000000000000000000000000000000000000000000000012]')

>> mat2str(X)

ans =
[0.25 0.100000000000000000000000000000000000000000000000000000000000000012]

>> mat2str(X,'class')

ans =
fi('numerictype',numerictype(1,128,127),'Value','[0.25 0.100000000000000000000000000000000000000000000000000000000000000012]')

>> mat2str(X,4,'class')

ans =
fi('numerictype',numerictype(1,128,127),'Value','[0.25 0.1]')
```

Detection of multiword operations for Simulink models

Detect multiword operations using Model Advisor checks

- Available in “Identify questionable fixed-point operations” check
- Detect blocks that generate multiword operations
- Provide highlighted links to trace back to questionable blocks

Identify questionable fixed-point operations
Analysis (^Triggers Update Diagram)
These operations can lead to non-optimal results
Run This Check

Result:  Warning

Check for multiword operations
Data types larger than the largest word size of your processor are handled in software with multiword operations. For guidelines on how to avoid multiword code, see [Fixed-Point Multiword Operations In Generated Code](#). The following blocks generate multiword code:

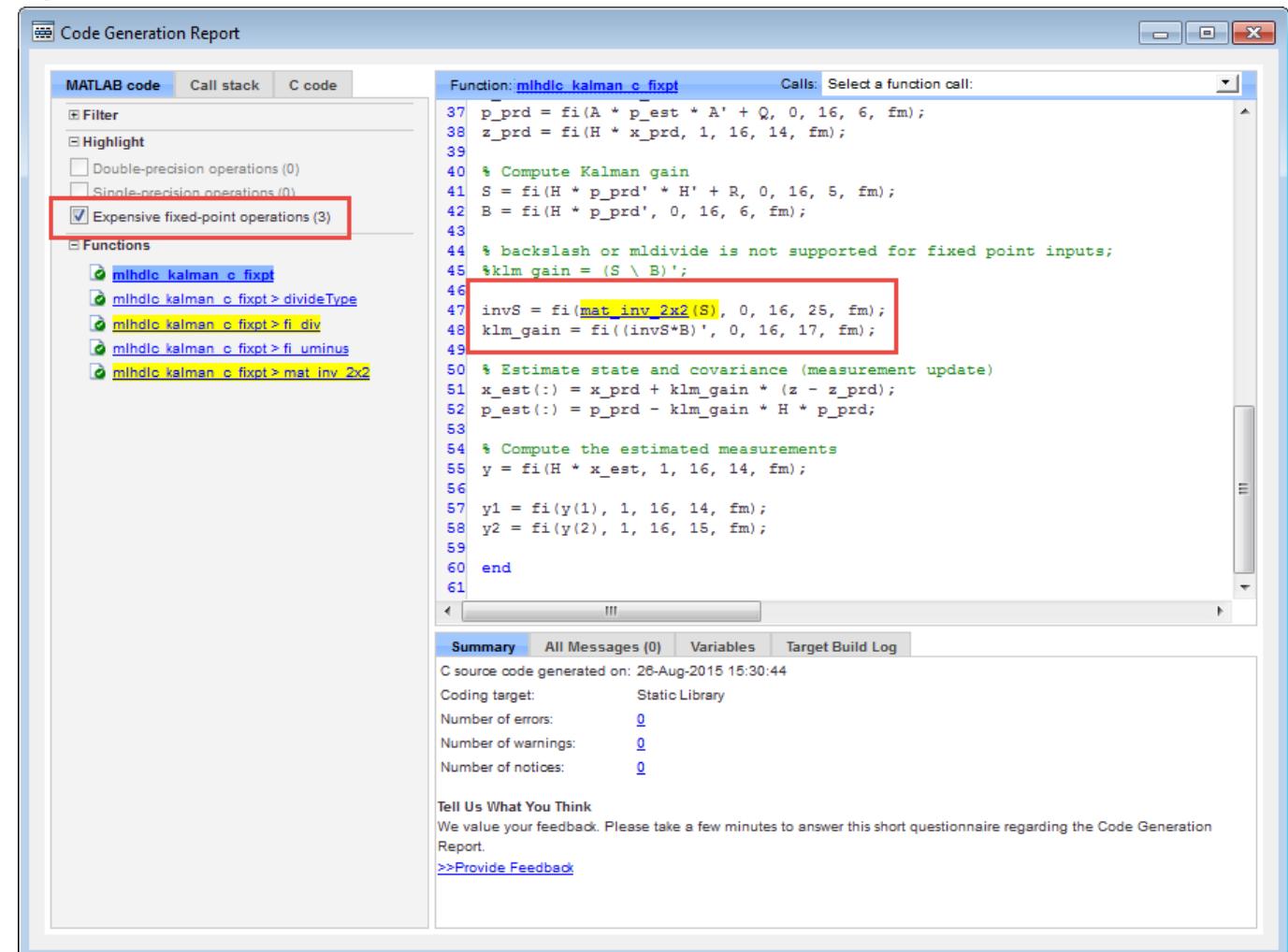
Warning

ID	Expensive fixed-point helper function
fxpdemo_multiword_example1/Gain	sMultiWordMul sMultiWordShl sMultiWord2MultiWord
fxpdemo_multiword_example1/Gain1	sMultiWordMul sMultiWordShr sMultiWord2MultiWord
fxpdemo_multiword_example1/Sum .../Data Type Conversion	MultiWordAdd sMultiWord2Double

Detection of multiword operations in MATLAB

Detect multiword operations using Fixed-Point Converter and MATLAB Coder apps

- Available in “Highlight Potential Data Type Issues” check
- Detect MATLAB code that generates multiword operations
- Highlight multiword operations in MATLAB code in the code generation report



Code Generation Report

MATLAB code Call stack C code

Filter

Highlight

Double-precision operations (0)

Single-precision operations (0)

Expensive fixed-point operations (3)

Functions

- mlhdic_kalman_c_fixpt
- mlhdic_kalman_c_fixpt>divideType
- mlhdic_kalman_c_fixpt>fi_div
- mlhdic_kalman_c_fixpt>fi_uminus
- mlhdic_kalman_c_fixpt>mat_inv_2x2

Function: `mlhdic_kalman_c_fixpt` Calls: Select a function call:

```
37 p_prd = fi(A * p_est * A' + Q, 0, 16, 6, fm);
38 z_prd = fi(H * x_prd, 1, 16, 14, fm);
39
40 % Compute Kalman gain
41 S = fi(H * p_prd' * H' + R, 0, 16, 5, fm);
42 B = fi(H * p_prd', 0, 16, 6, fm);
43
44 % backslash or mldivide is not supported for fixed point inputs;
45 %klm_gain = (S \ B)';
46
47 invS = fi(mat_inv_2x2(S), 0, 16, 25, fm);
48 klm_gain = fi((invS*B)', 0, 16, 17, fm);
49
50 % Estimate state and covariance (measurement update)
51 x_est(:) = x_prd + klm_gain * (z - z_prd);
52 p_est(:) = p_prd - klm_gain * H * p_prd;
53
54 % Compute the estimated measurements
55 y = fi(H * x_est, 1, 16, 14, fm);
56
57 y1 = fi(y(1), 1, 16, 14, fm);
58 y2 = fi(y(2), 1, 16, 15, fm);
59
60 end
61
```

Summary All Messages (0) Variables Target Build Log

C source code generated on: 26-Aug-2015 15:30:44
Coding target: Static Library
Number of errors: 0
Number of warnings: 0
Number of notices: 0

Tell Us What You Think
We value your feedback. Please take a few minutes to answer this short questionnaire regarding the Code Generation Report.
[>>Provide Feedback](#)

System object instrumentation in the Fixed-Point Tool

Collect simulation ranges and propose data types for select System objects used inside a MATLAB function block

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

- `dsp.ArrayVectorAdder`
- `dsp.BiquadFilter`
- `dsp.FIRRateConverter`
- `dsp.LowerTriangularSolver`
- `dsp.UpperTriangularSolver`
- `dsp.FIRFilter` (DF, TDF structure)
- `dsp.FIRDecimator`
- `dsp.FIRInterpolator`
- `dsp.VariableFractionalDelay`
- `dsp.Window`
- `dsp.LUFactor`