

# Latest Features in Simulink Fixed Point

September 2009

**R2009b**

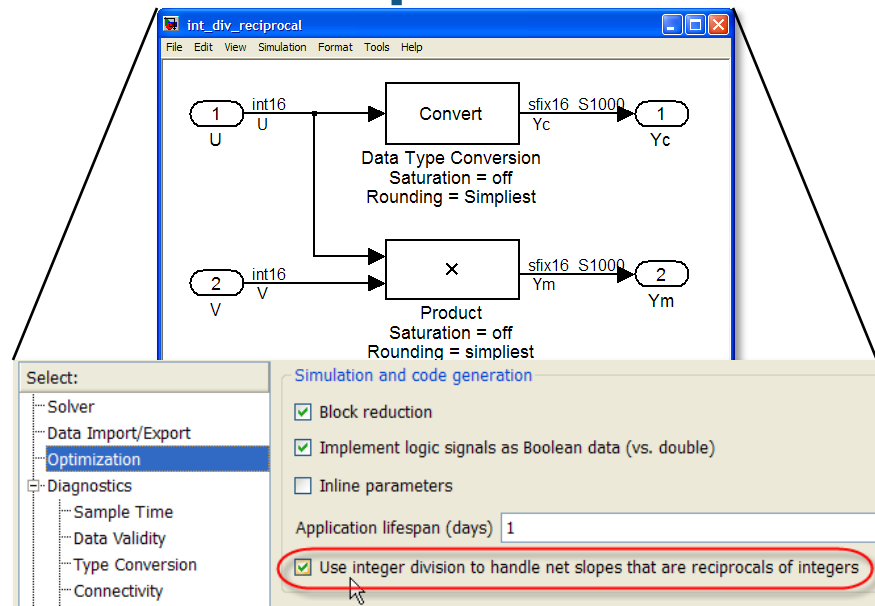
## Demo Download

- The demos presented in this document can be found in the shipping version of the product or at the following page:
  - [www.mathworks.com/support/solutions/en/data/1-ALTSET/index.html](http://www.mathworks.com/support/solutions/en/data/1-ALTSET/index.html)

# Fixed-Point Net Slope Correction Optimization

## Speed up execution using integer division to handle net slopes

- Mismatched scaling of the input-output operations needs net scaling correction (multiplication followed by barrel shift).
- Optimization replaces this operation with integer division under certain simplicity and accuracy conditions.
- This approach is useful on processors where the cost of integer division is less than that of a combined multiplication and barrel shifting operation.
- Model Advisor check is available.



Conceptually, we want:

$$Y_c = 0.001 * U;$$

$$Y_m = 0.001 * U * V;$$

### Optimization Off (multiplication and barrel shift)

```
Yc =(int16)((int32)U * 16777 >> 24);
Ym =(int16)((int32)(int16)((int32)U * (int32)V >> 10) * 16777 >> 14);
```

### Optimization On (integer division)

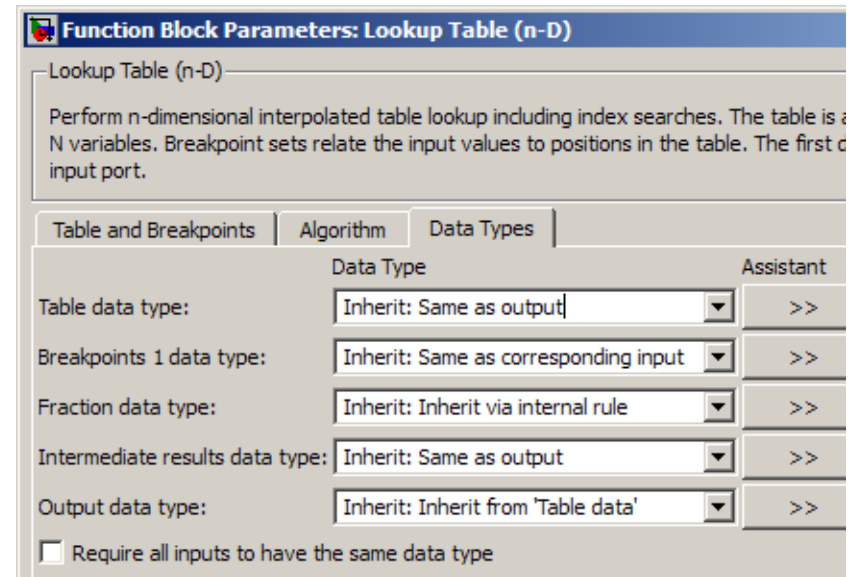
```
Yc = (int16)(U / 1000);
Ym = (int16)((int32)U * (int32)V / 1000);
```

```
>> int_div_reciprocal
```

# Independent Data Types in n-D Lookup Tables

**Lookup Table (n-D) block supports parameter data types different from signal data types, making it easier to model lookup tables and improving code efficiency**

- Independent data type specifications for
  - Table parameters
  - Breakpoint parameters
  - Intermediate results
- Less memory needed for storing table data that uses a smaller type than the output signal
- Shared prescaled table data between Lookup Table (n-D) blocks with different output data types
- Shared custom storage table data in generated code for blocks with different output data types



# Optimizations of Lookup Evenly Spaced

## Prelookup and Lookup Table (n-D) allocate less memory for evenly spaced breakpoint data

- Only the first two breakpoints are stored; offset and spacing of remaining points are calculated from these two data.
- Memory is reduced, and the execution speed of the generated code is increased.
- Breakpoint data cannot be tunable.

Table and Breakpoints (BP)

	Data
Table	tanh([0:4:40])
BP 1	[0:4:40] ← Breakpoint input

```

/* Constant parameters (auto storage) */
const ConstParam_lund_pow2_1D_lund_pow2_1D_ConstP = {
  /* Computed Parameter: Table
   * '<Root>/Lookup Table (n-D)'
   */
  { 0, 1023, 1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024 },

  /* Computed Parameter: BreakpointsForDimension1
   * '<Root>/Lookup Table (n-D)'
   */
  { 0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40 }
};
    
```

Breakpoint memory allocated in R2009a, which is removed in R2009b