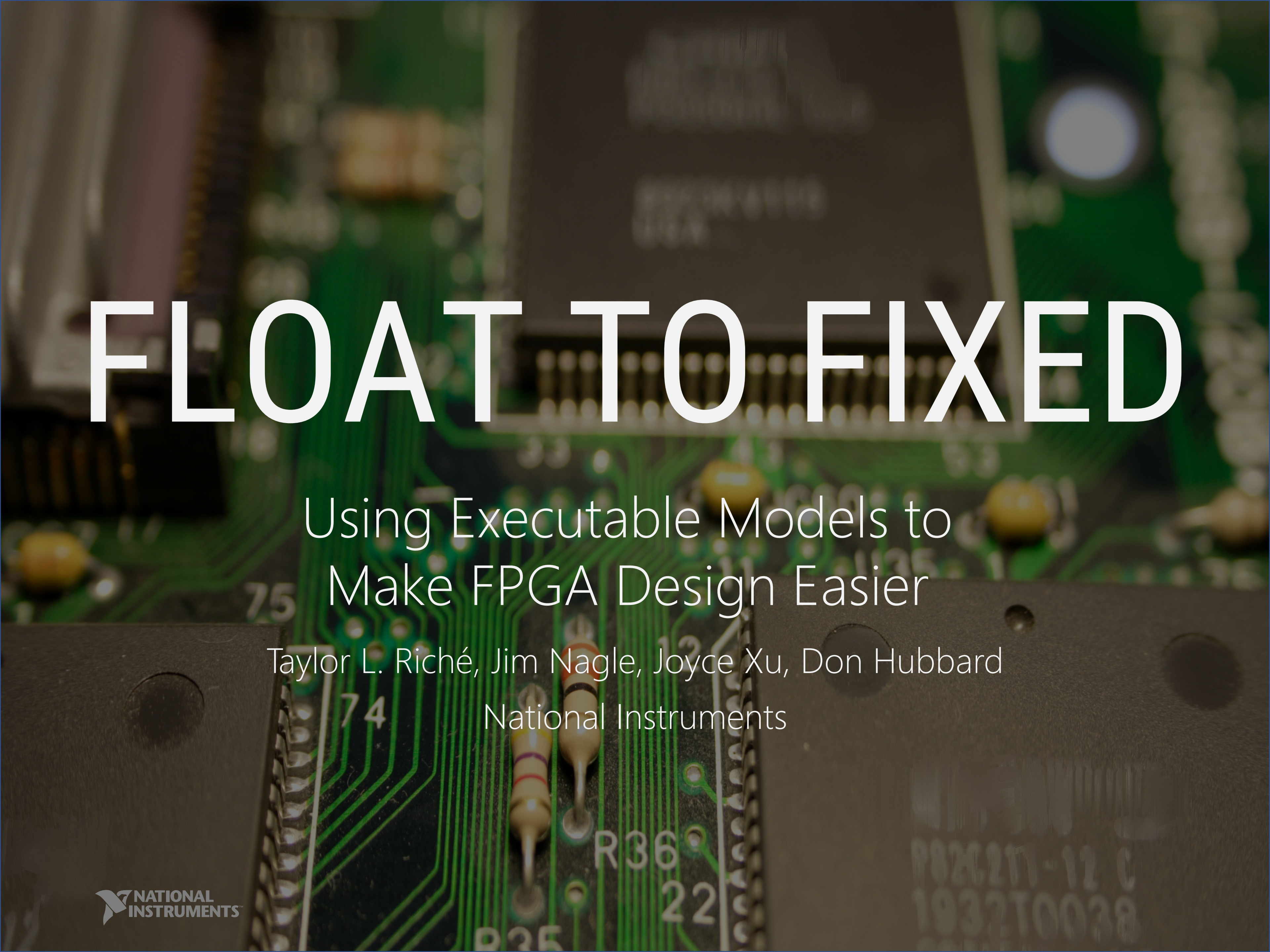


# FLOAT TO FIXED

## Using Executable Models to Make FPGA Design Easier

Taylor L. Riché, Jim Nagle, Joyce Xu, Don Hubbard

National Instruments

# WHO AM I?

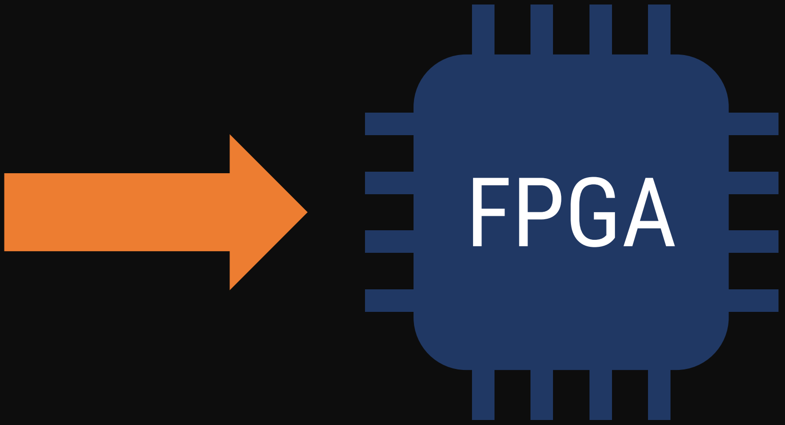
## Taylor L. Riché

Principal Product Owner
National Instruments

Manager
Local Arrangements Chair MODELS 2017
Developer
Postdoc (Riché et. al, MODELS 2010)
PhD Student

# GOAL:

$$f(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)$$



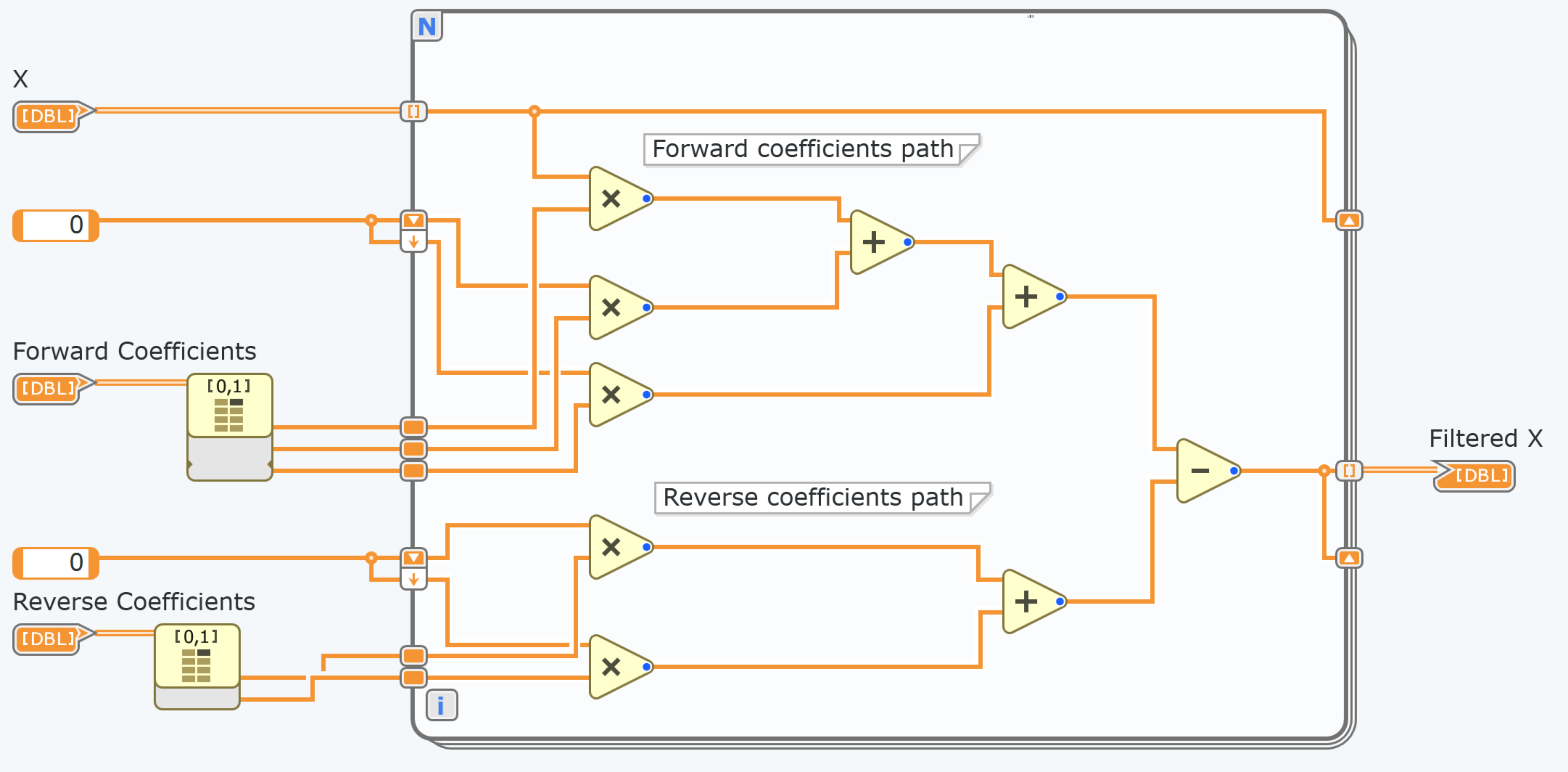
# PROBLEM: THIS IS HARD!

# SOLUTION:

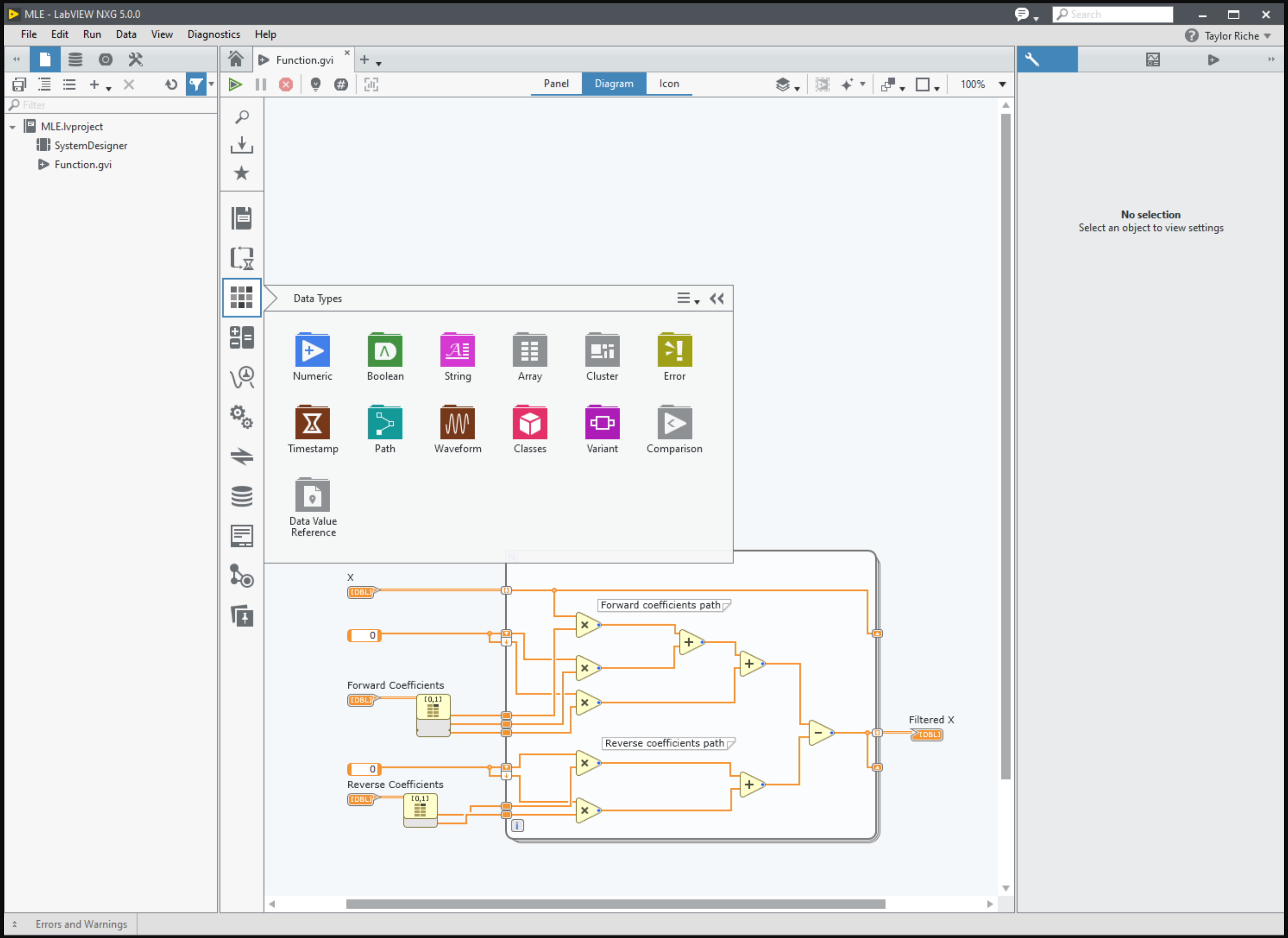
1. Have algorithm designers model their new algorithms in LabVIEW NXG
2. Use the executable nature of LabVIEW NXG to generate an initial model transform
3. Provide tools to help designers create and apply the remaining transforms
4. Along the way, executability gives constant feedback on “correct enough” by construction

# A QUICK INTRO TO G

# SO THIS IS G…


X
[DBL]
0
Forward Coefficients
[DBL]
[0,1]
0
Reverse Coefficients
[DBL]
[0,1]
N
i
Forward coefficients path
Reverse coefficients path
×
+
−
Filtered X
[DBL]

# AND THIS IS LABVIEW NXG…



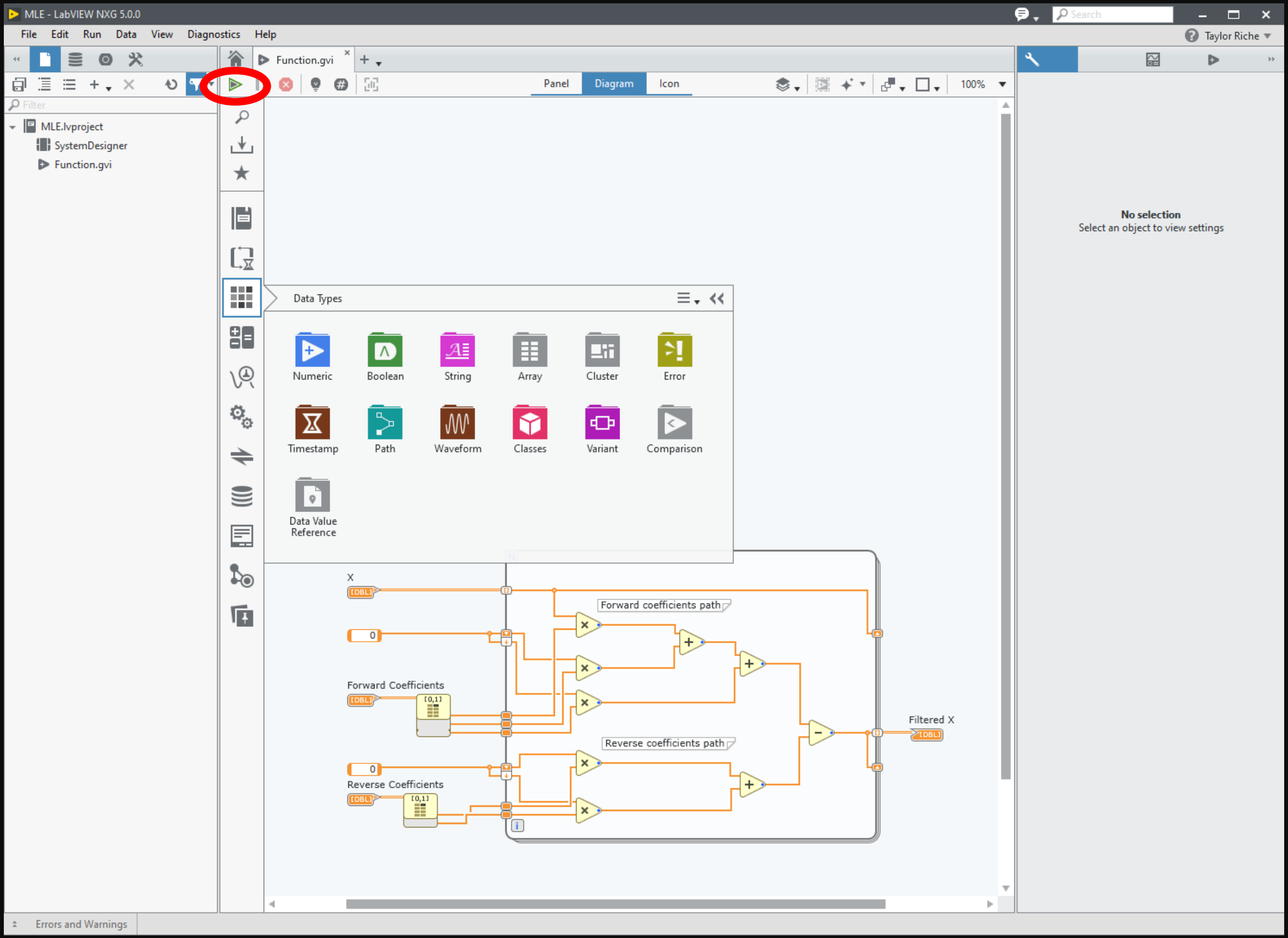
# A FEW DETAILS

1. Graphical dataflow language
2. Test and Measurement DSL
3. Allows you to model computation
4. Allows you to model hardware configuration
5. Different libraries of mathematical tools
6. Allows creation of EXEs and reusable IP
7. Maps computation to desktop, FPGA, and Realtime

http://www.ni.com/labviewnxg

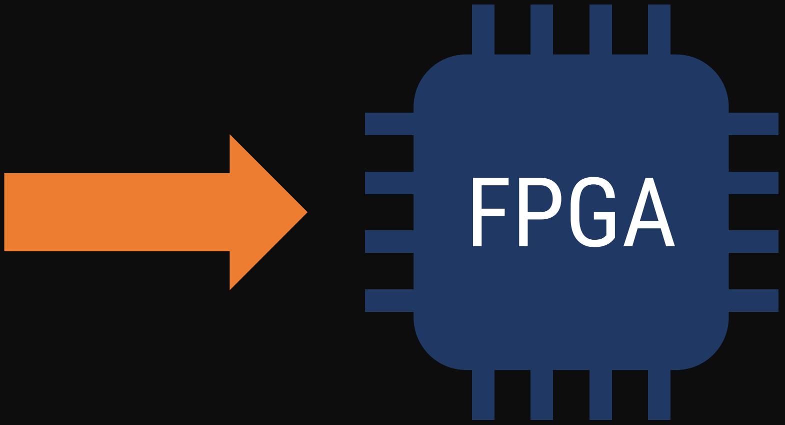
# EXECUTABILITY

# IT RUNS


MLE - LabVIEW NXG 5.0.0
Search
File
Edit
Run
Data
View
Diagnostics
Help
Taylor Riche
Function.gvi
Filter
MLE.lvproject
SystemDesigner
Function.gvi
Panel
Diagram
Icon
100%
No selection
Select an object to view settings
Data Types
Numeric
Boolean
String
Array
Cluster
Error
Timestamp
Path
Waveform
Classes
Variant
Comparison
Data Value Reference
X
0
Forward Coefficients
Reverse Coefficients
Forward coefficients path
Reverse coefficients path
Filtered X
Errors and Warnings

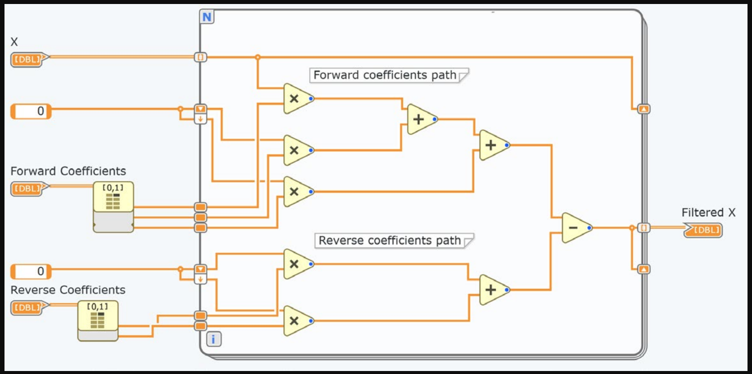
# GOAL:

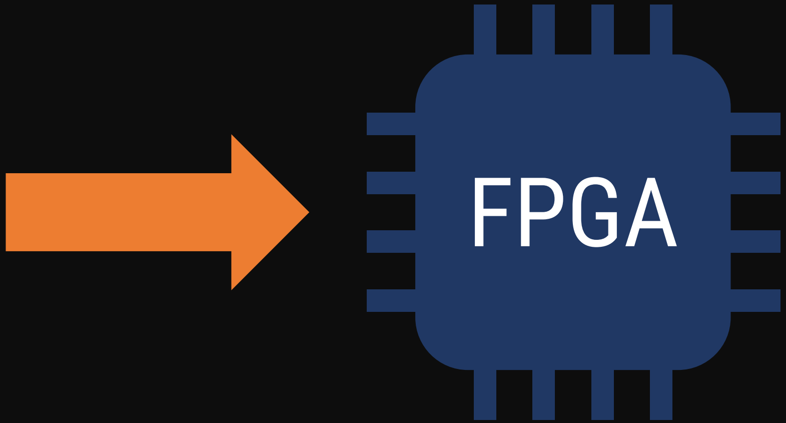
$$f(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)$$



FPGA

# GOAL:


N
X
[DBL]
0
Forward coefficients path
Forward Coefficients
[DBL]
[0,1]
Filtered X
[DBL]
Reverse coefficients path
0
Reverse Coefficients
[DBL]
[0,1]
i


FPGA

# THE PROBLEM:

- Throughput constraints require HW
- FPGAs have limited resources
- Floating point takes many resources
- Digital design experts are expensive

# THE RESOURCE SOLUTION: FIXED-POINT ARITHMETIC

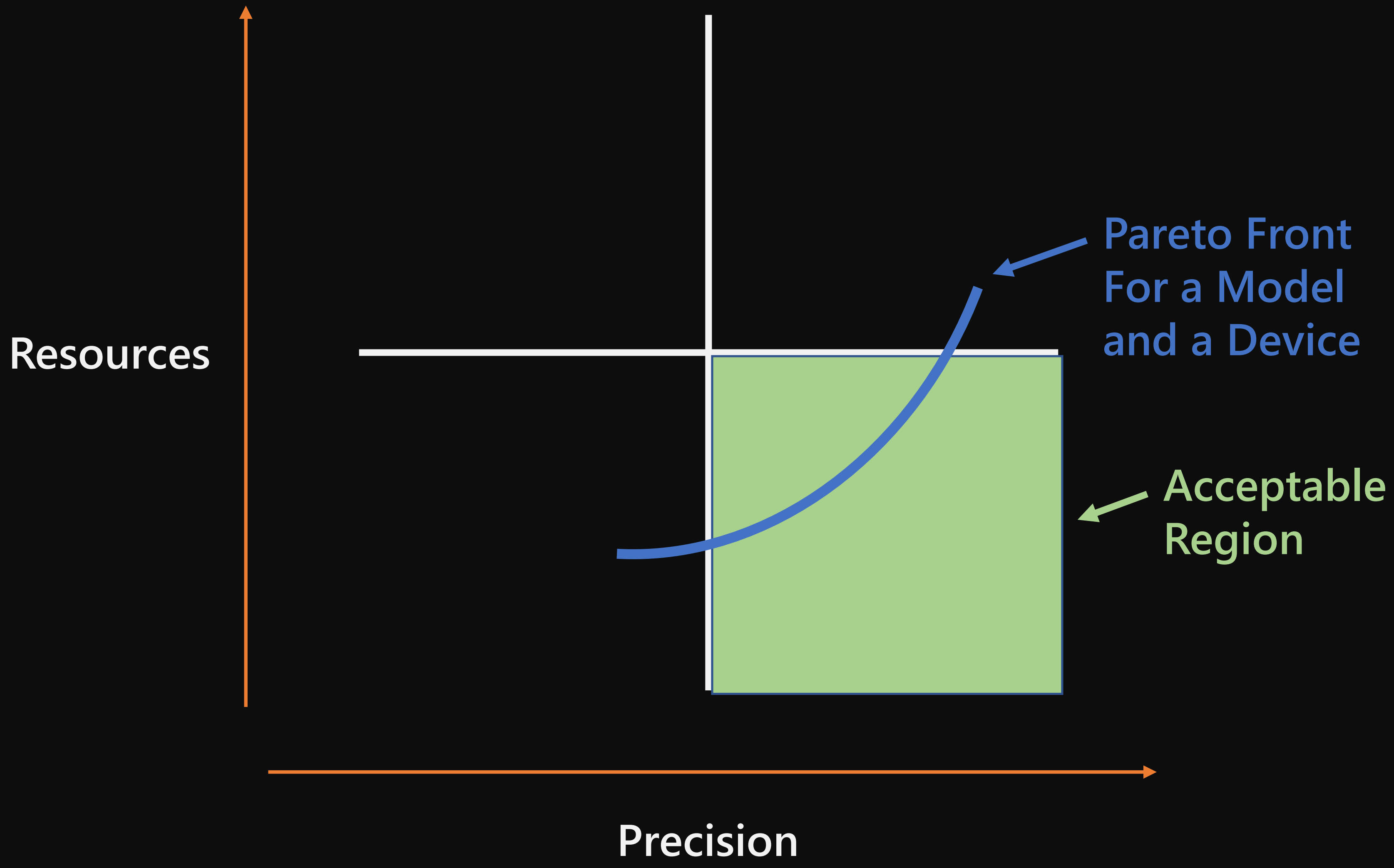
# FIXED POINT

$$\underbrace{XXXX}_{\text{4 integer bits}}.\underbrace{YYYYYYYYYYYY}_{\text{12 fractional bits}}$$

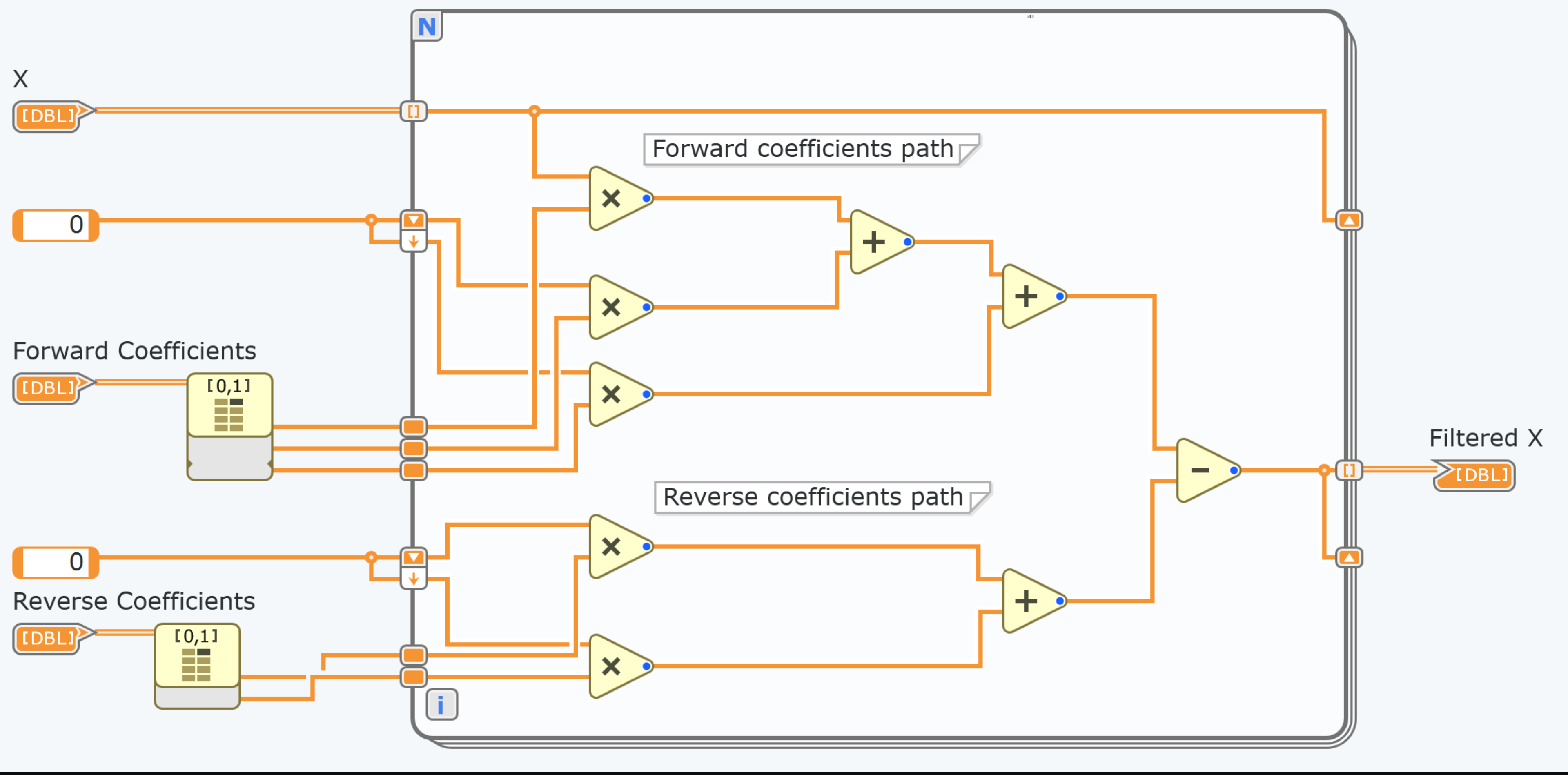
# FIXED POINT CHALLENGES

- Too few integer bits → overflow
- Too few fractional bits → loss of precision
- More bits use more FPGA resources

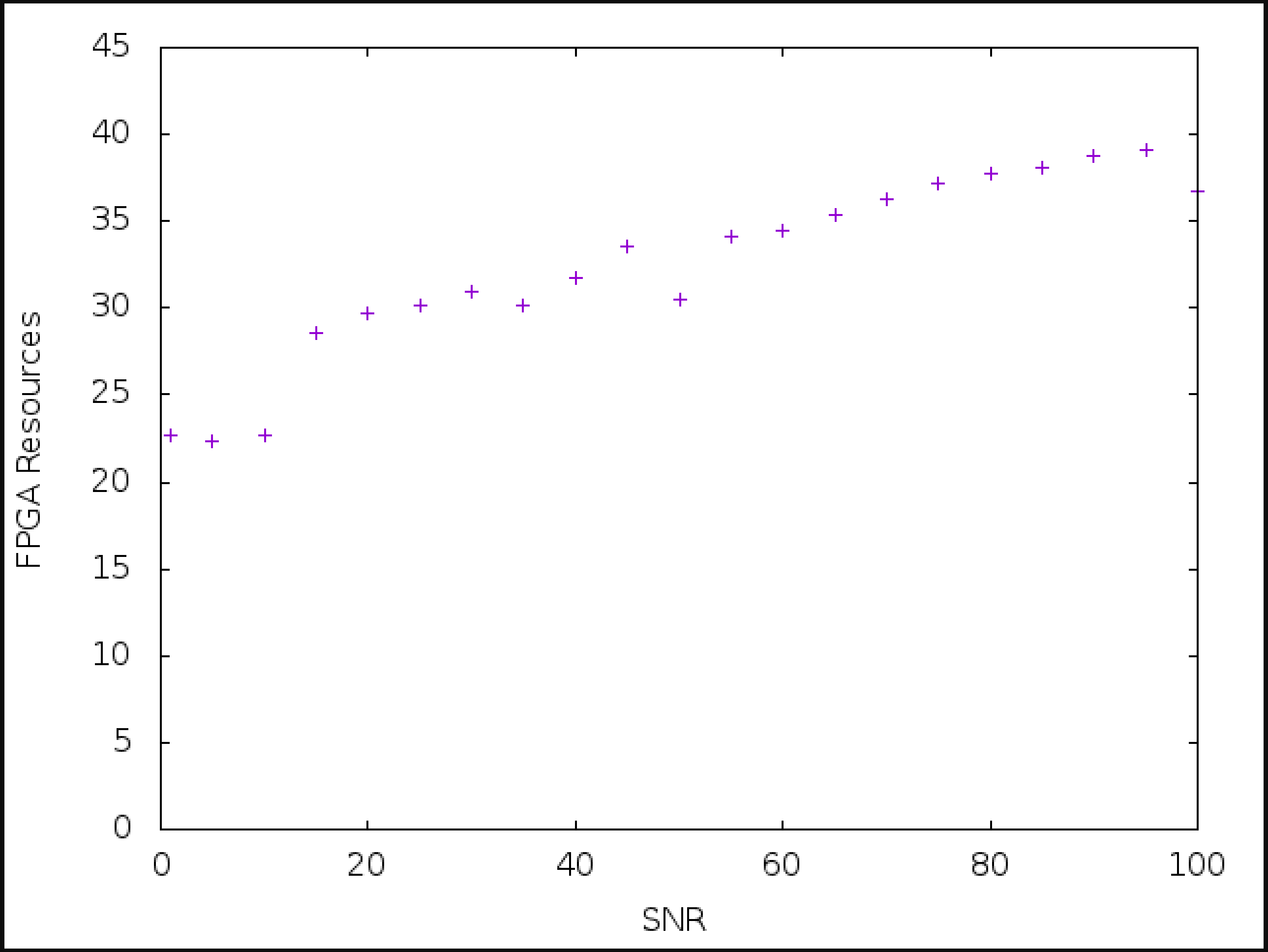
# RESOURCES V. PRECISION



# RESOURCE EXAMPLE



# RESOURCE EXAMPLE



# OUR SOLUTION: LABVIEW NXG F2F

# DESIGN TENANTS:

- Not trying to make the best F2F tool
- Usability was paramount
  - Don’t hire a FXP or DD expert
  - No spreadsheets!
- Don’t try to encode all constraints
  - Focus on Signal-to-Noise Ratio (SNR)
- Perfect is the enemy of shipping

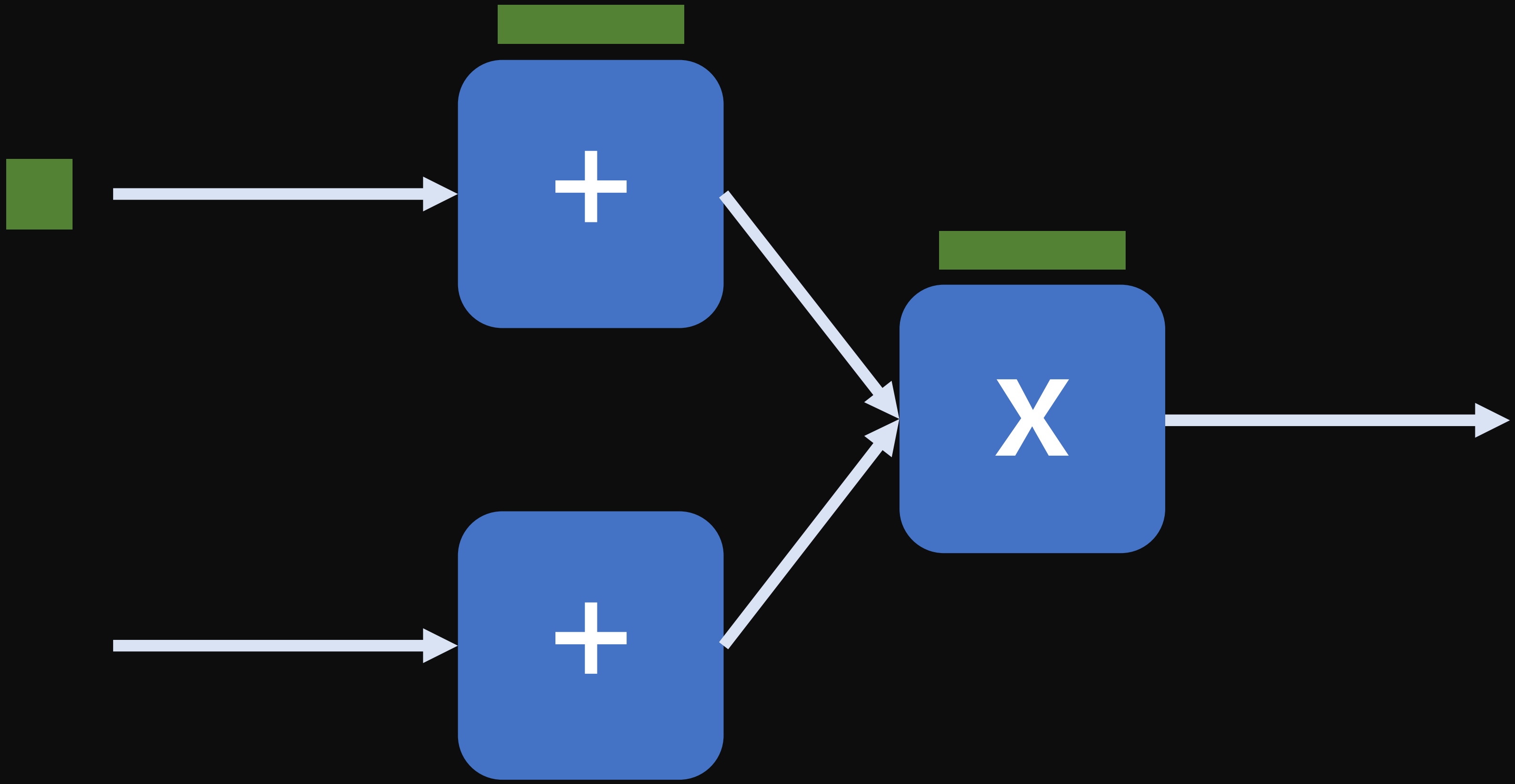
# BUT FIRST, SOME FORMALISMS

$$M_{golden} \rightarrow M_{FXP}$$

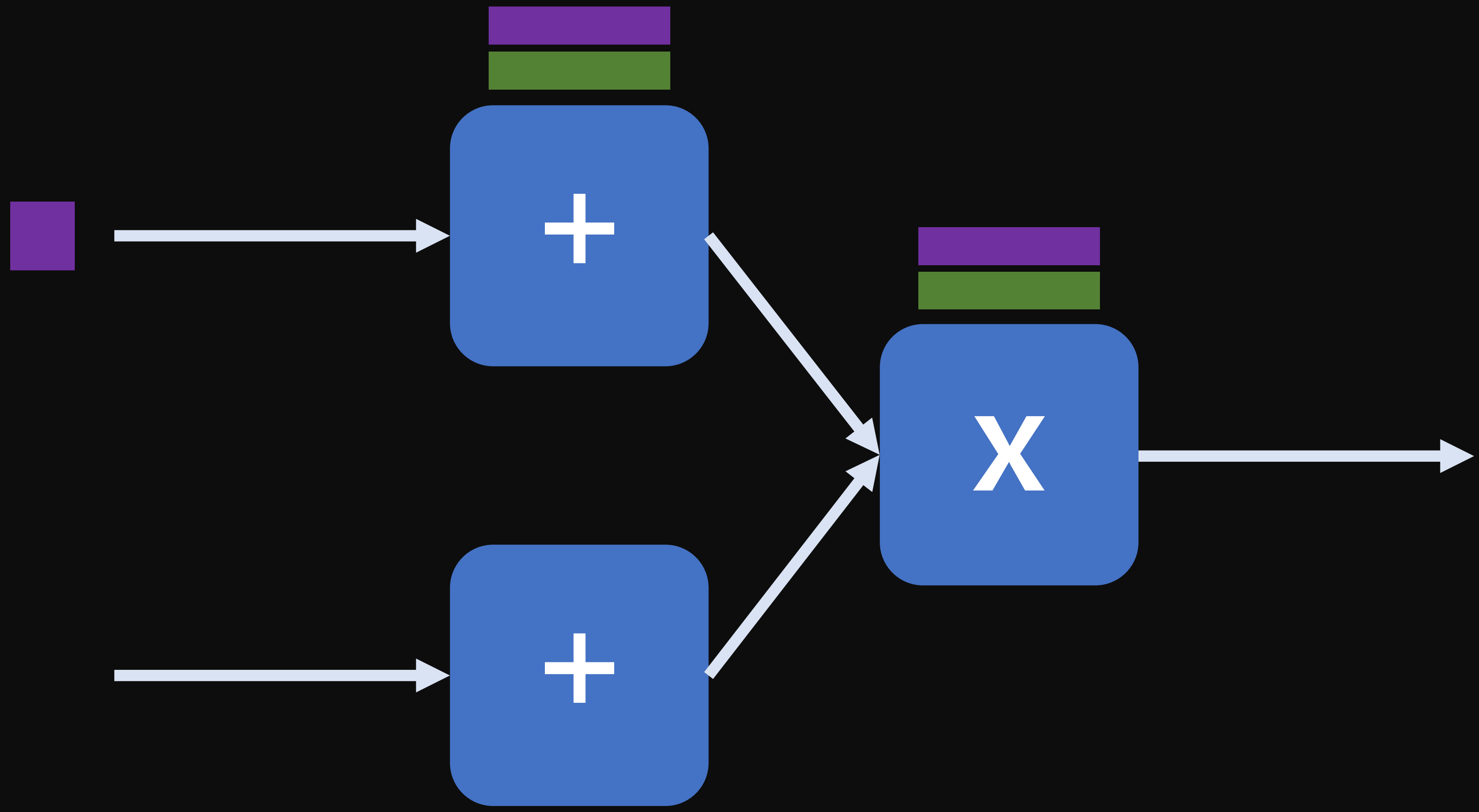
$$T = T_{FXP1} \ldots T_{FXPn}$$

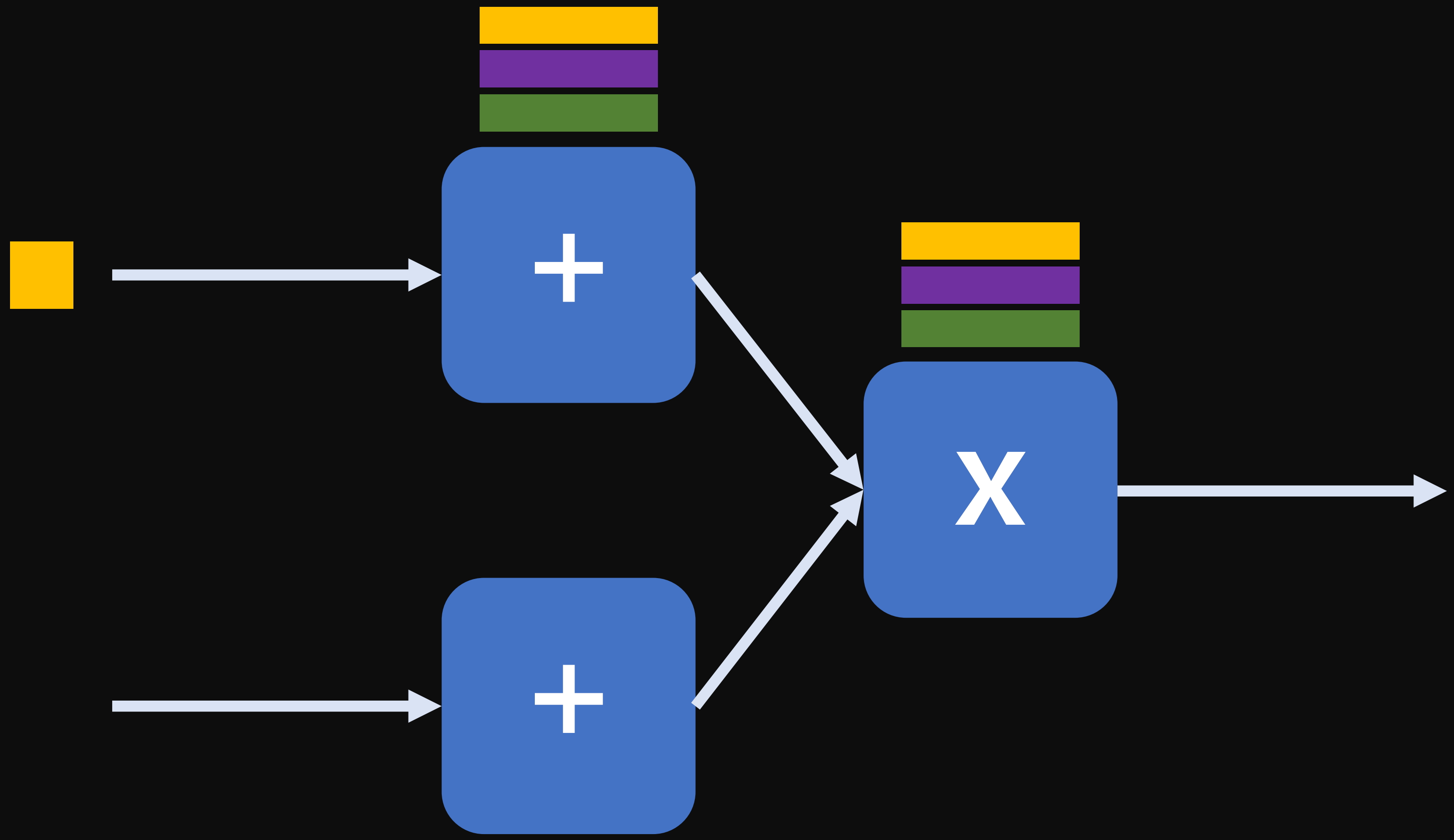
$$T_{FXPi} = \{t_1, t_2, \ldots t_k\}$$

$$t_j = [DBL \rightarrow (1.15)]$$



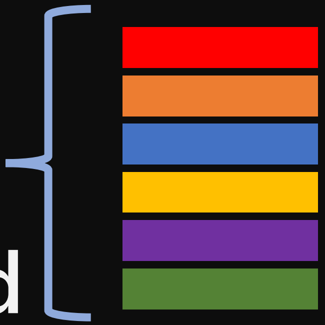
+
+
X

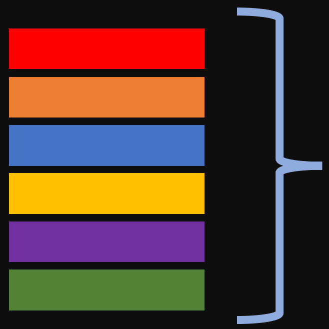

+
+
X


+
+
X

$$IntMax$$

integer bits required for no overflow





$FracMin$

fractional bits required to meet local SNR

$$t = [DBL \rightarrow (I.F)]$$

🟥🟧🟦🟨🟪🟩 $\vec{O}_i$

$$FXP_{(I.F)}(\vec{O}_i)$$

$$\|\vec{O}_i - FXP_{I.F}(\vec{O}_i)\|$$

$$\sqrt{\sum_{j=1}^{n}(o_j - FXP_{I.F}(o_j))^2}$$

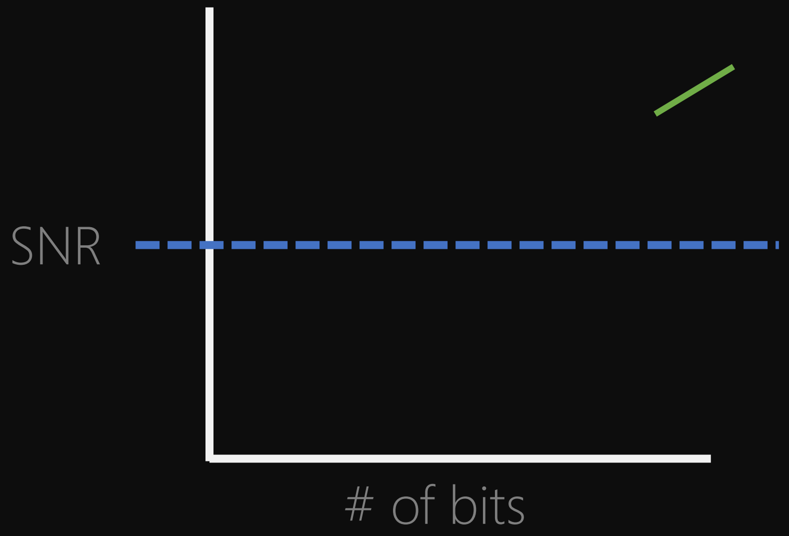
$$\|\vec{O}_i\| = \sqrt{\sum_{j=1}^{n} o_j^2}$$

$$SNR_i = \log_{10}\left(\frac{\|\vec{O}_i\|}{\|\vec{O}_i - FXP_{I.F}(\vec{O}_i)\|}\right)$$

NATIONAL INSTRUMENTS™

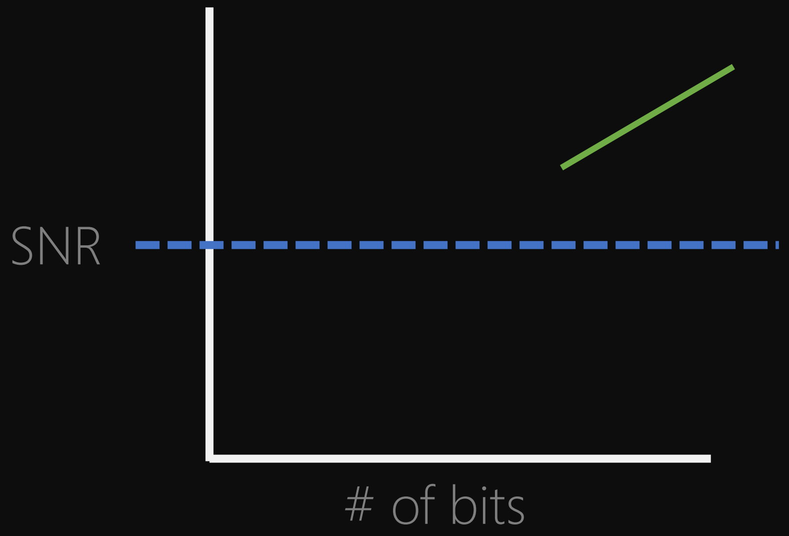
# FINDING FRACTIONAL BITS

XXXX.YYYYYYYYYYYY



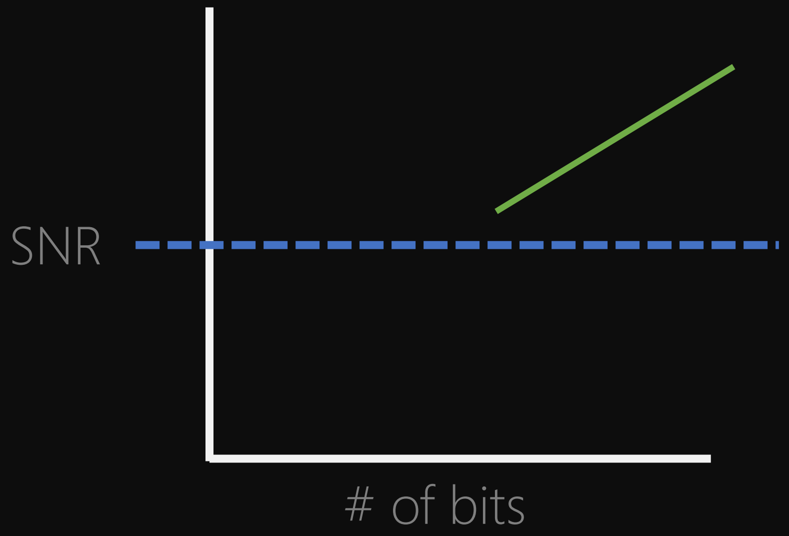
# FINDING FRACTIONAL BITS

XXXX.YYYYYYYYYYY



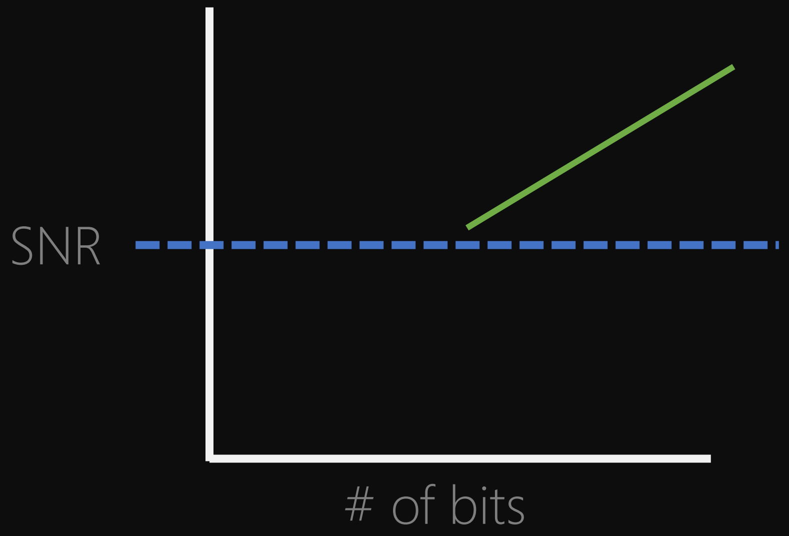
# FINDING FRACTIONAL BITS

XXXX.YYYYYYYYYY



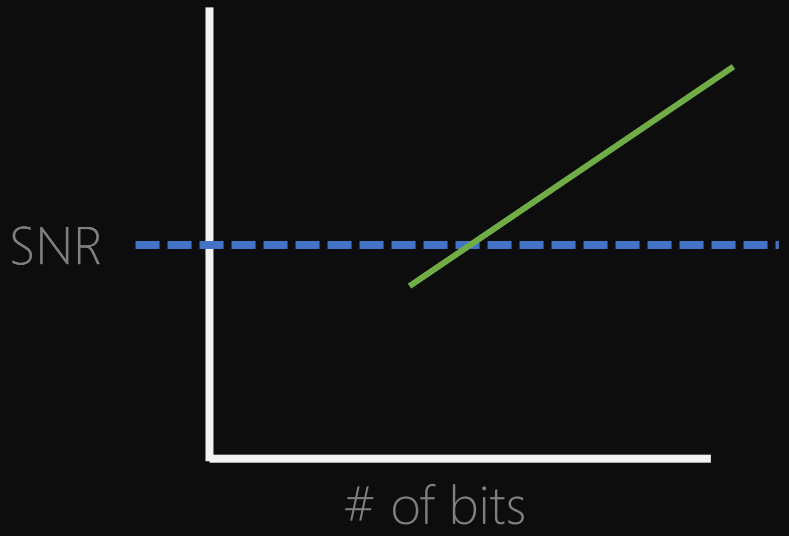
# FINDING FRACTIONAL BITS

XXXX.YYYYYYYYY



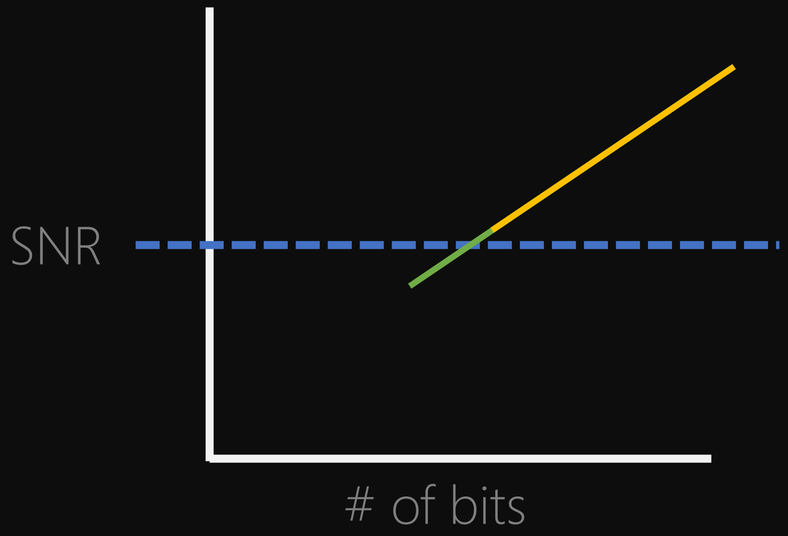
# FINDING FRACTIONAL BITS

XXXX.YYYYYYYY



# FINDING FRACTIONAL BITS

$$\text{XXXX.}\overbrace{\text{YYYYYYYYY}}^{\text{9 fractional bits}}$$



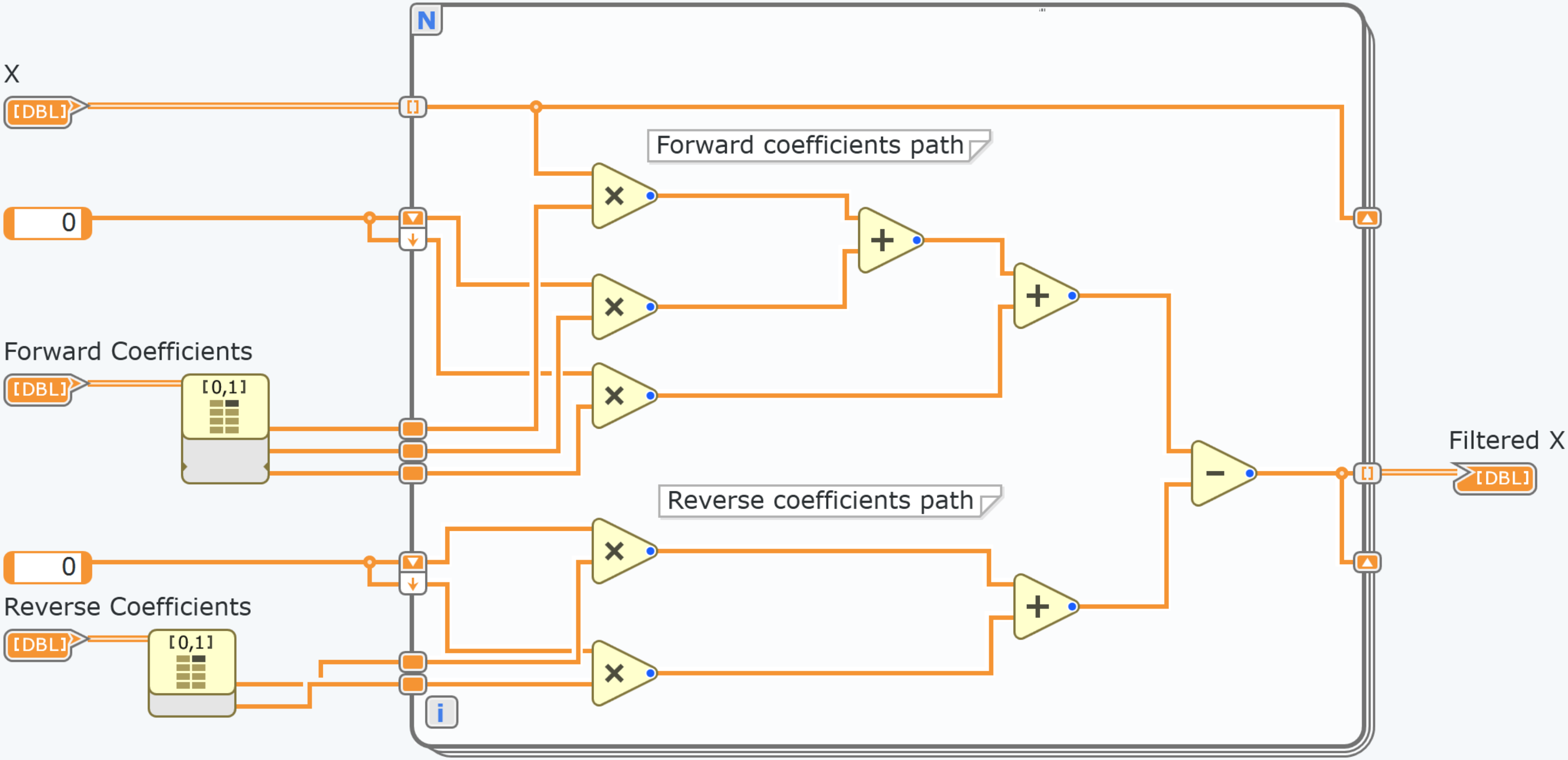
$$T = T_{FXP1} \ldots T_{FXPn}$$

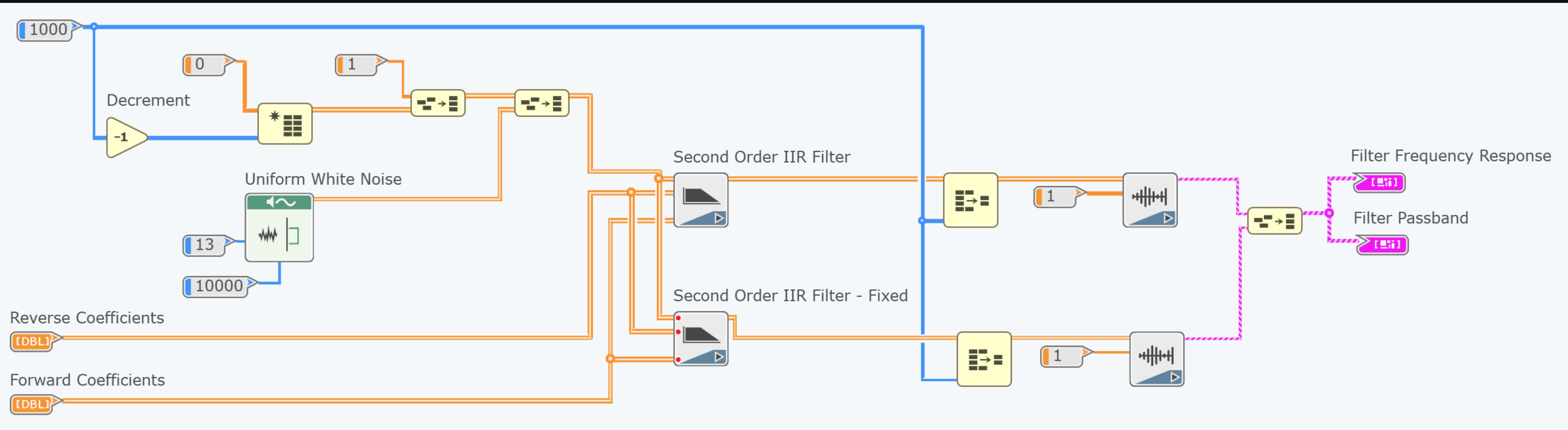
$$T_{FXP1} = \{t_1, t_2, \ldots t_k\}$$

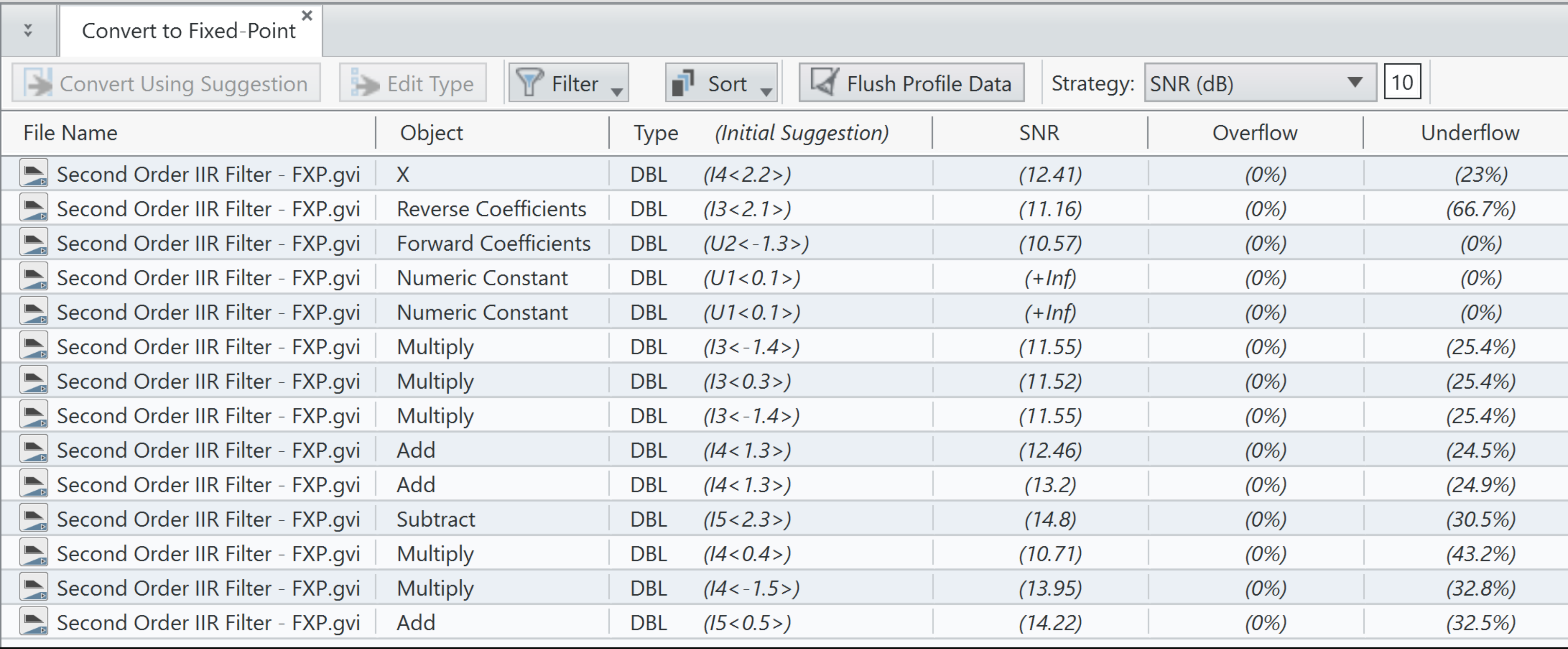
$$t_j = [DBL \rightarrow (IntMax_j.FracMin_j)]$$

NATIONAL INSTRUMENTS™

# THE WORKFLOW

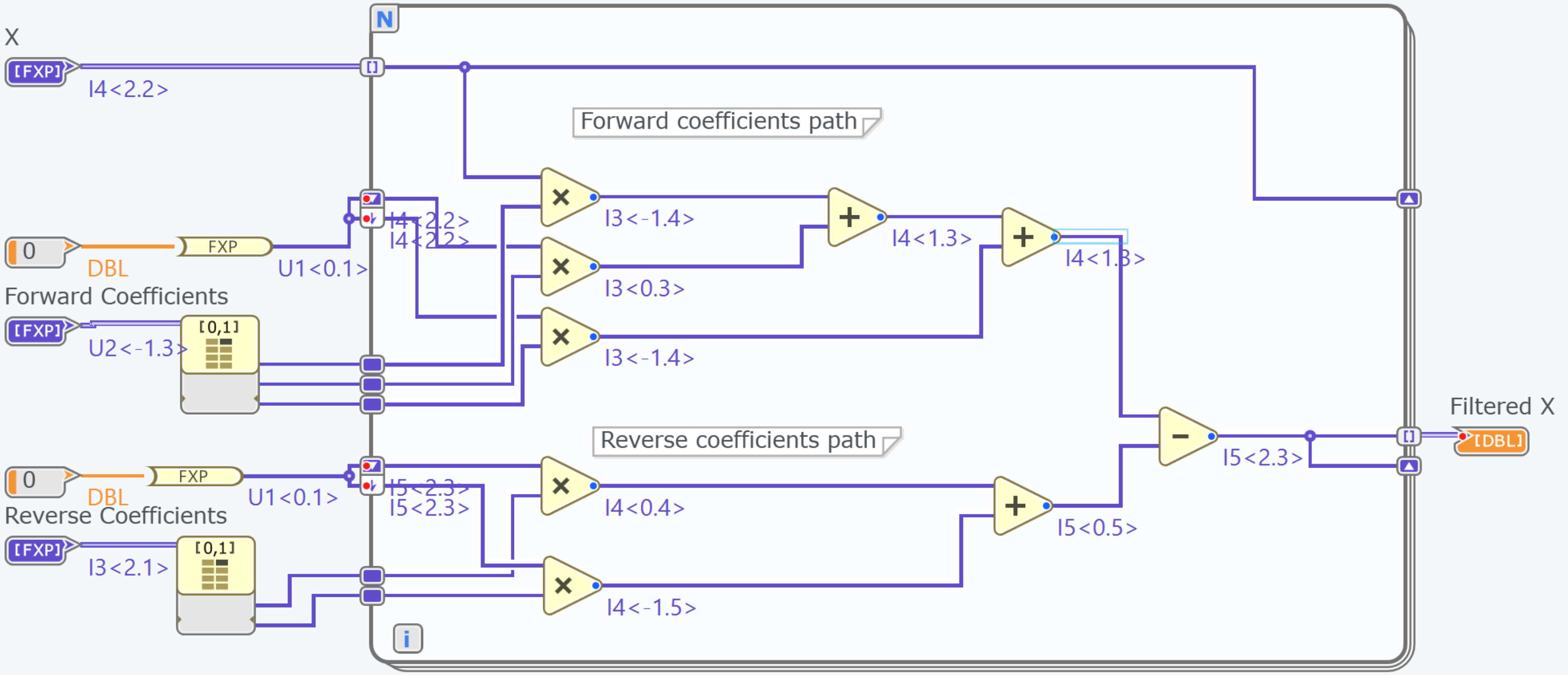

X
[DBL]
N
Forward coefficients path
0
Forward Coefficients
[DBL]
[0,1]
Filtered X
[DBL]
Reverse coefficients path
0
Reverse Coefficients
[DBL]
[0,1]
i


1000
0
1
Decrement
-1
Uniform White Noise
13
10000
Reverse Coefficients
[DBL]
Forward Coefficients
[DBL]
Second Order IIR Filter
Second Order IIR Filter - Fixed
1
1
Filter Frequency Response
Filter Passband

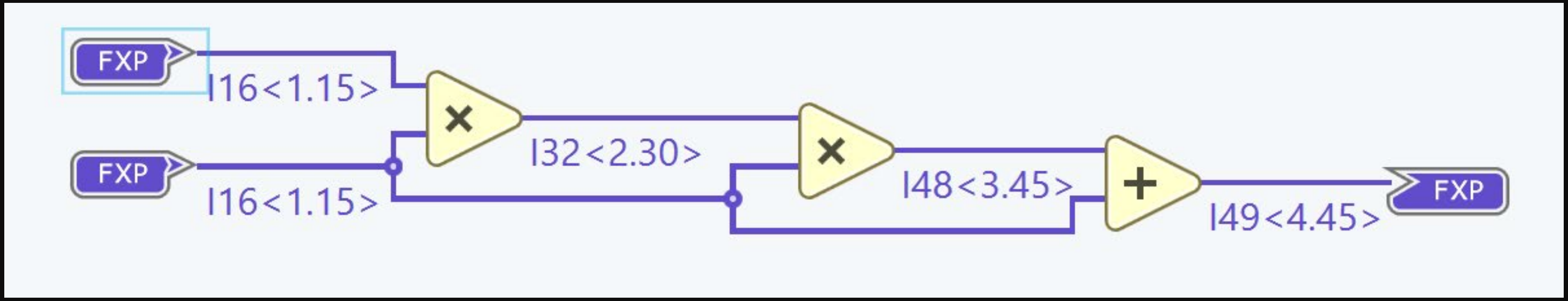


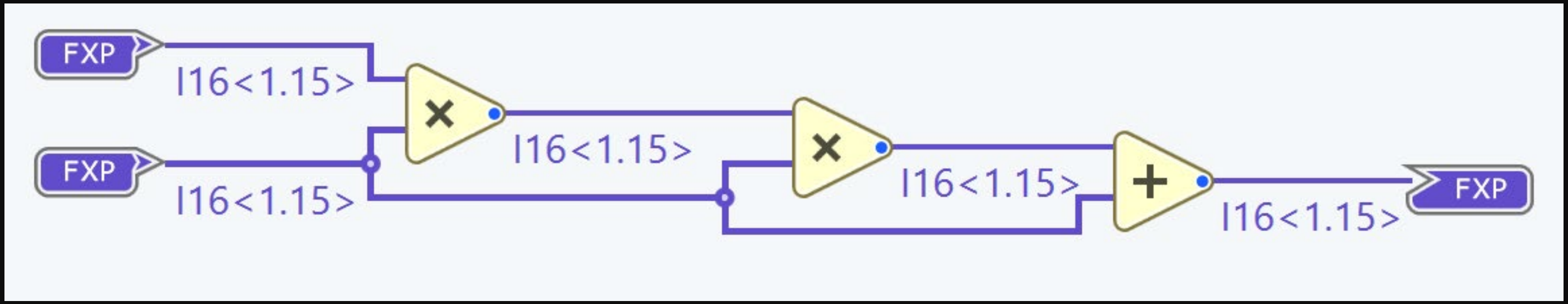
Strategy: SNR (dB) 10

| File Name | Object | Type | *(Initial Suggestion)* | SNR | Overflow | Underflow |
|---|---|---|---|---|---|---|
| Second Order IIR Filter - FXP.gvi | X | DBL | *(I4<2.2>)* | *(12.41)* | *(0%)* | *(23%)* |
| Second Order IIR Filter - FXP.gvi | Reverse Coefficients | DBL | *(I3<2.1>)* | *(11.16)* | *(0%)* | *(66.7%)* |
| Second Order IIR Filter - FXP.gvi | Forward Coefficients | DBL | *(U2<-1.3>)* | *(10.57)* | *(0%)* | *(0%)* |
| Second Order IIR Filter - FXP.gvi | Numeric Constant | DBL | *(U1<0.1>)* | *(+Inf)* | *(0%)* | *(0%)* |
| Second Order IIR Filter - FXP.gvi | Numeric Constant | DBL | *(U1<0.1>)* | *(+Inf)* | *(0%)* | *(0%)* |
| Second Order IIR Filter - FXP.gvi | Multiply | DBL | *(I3<-1.4>)* | *(11.55)* | *(0%)* | *(25.4%)* |
| Second Order IIR Filter - FXP.gvi | Multiply | DBL | *(I3<0.3>)* | *(11.52)* | *(0%)* | *(25.4%)* |
| Second Order IIR Filter - FXP.gvi | Multiply | DBL | *(I3<-1.4>)* | *(11.55)* | *(0%)* | *(25.4%)* |
| Second Order IIR Filter - FXP.gvi | Add | DBL | *(I4<1.3>)* | *(12.46)* | *(0%)* | *(24.5%)* |
| Second Order IIR Filter - FXP.gvi | Add | DBL | *(I4<1.3>)* | *(13.2)* | *(0%)* | *(24.9%)* |
| Second Order IIR Filter - FXP.gvi | Subtract | DBL | *(I5<2.3>)* | *(14.8)* | *(0%)* | *(30.5%)* |
| Second Order IIR Filter - FXP.gvi | Multiply | DBL | *(I4<0.4>)* | *(10.71)* | *(0%)* | *(43.2%)* |
| Second Order IIR Filter - FXP.gvi | Multiply | DBL | *(I4<-1.5>)* | *(13.95)* | *(0%)* | *(32.8%)* |
| Second Order IIR Filter - FXP.gvi | Add | DBL | *(I5<0.5>)* | *(14.22)* | *(0%)* | *(32.5%)* |

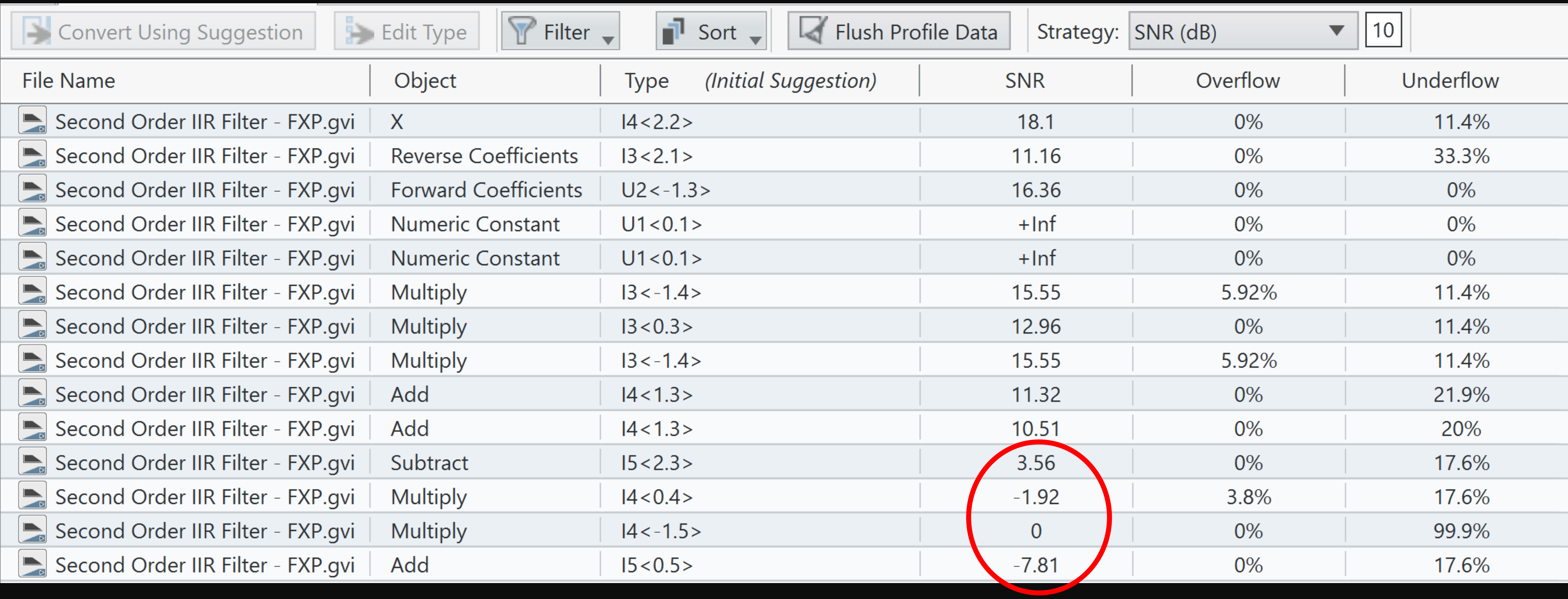

X
[FXP]
I4<2.2>
N
Forward coefficients path
0
DBL
FXP
U1<0.1>
Forward Coefficients
[FXP]
U2<-1.3>
[0,1]
I4<2.2>
I4<2.2>
I3<-1.4>
I3<0.3>
I3<-1.4>
I4<1.3>
I4<1.3>
Reverse coefficients path
0
DBL
FXP
U1<0.1>
Reverse Coefficients
[FXP]
I3<2.1>
[0,1]
I5<2.3>
I5<2.3>
I4<0.4>
I4<-1.5>
I5<0.5>
I5<2.3>
Filtered X
[DBL]
i

# LOCAL ERROR V. REAL ERROR


FXP
I16<1.15>
FXP
I16<1.15>
I32<2.30>
I48<3.45>
I49<4.45>
FXP

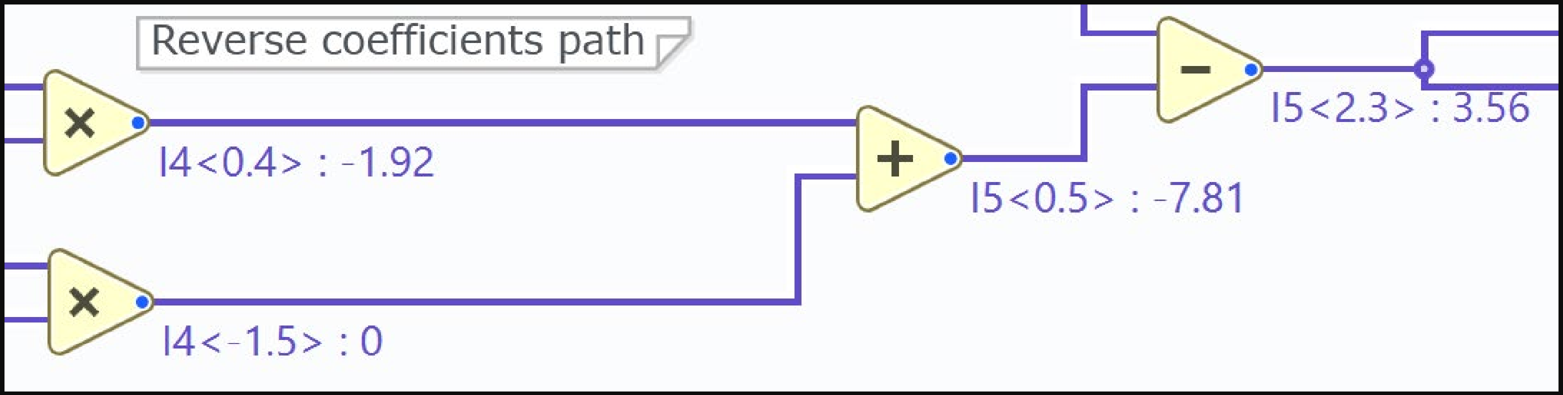

FXP
I16<1.15>
FXP
I16<1.15>
I16<1.15>
I16<1.15>
I16<1.15>
FXP

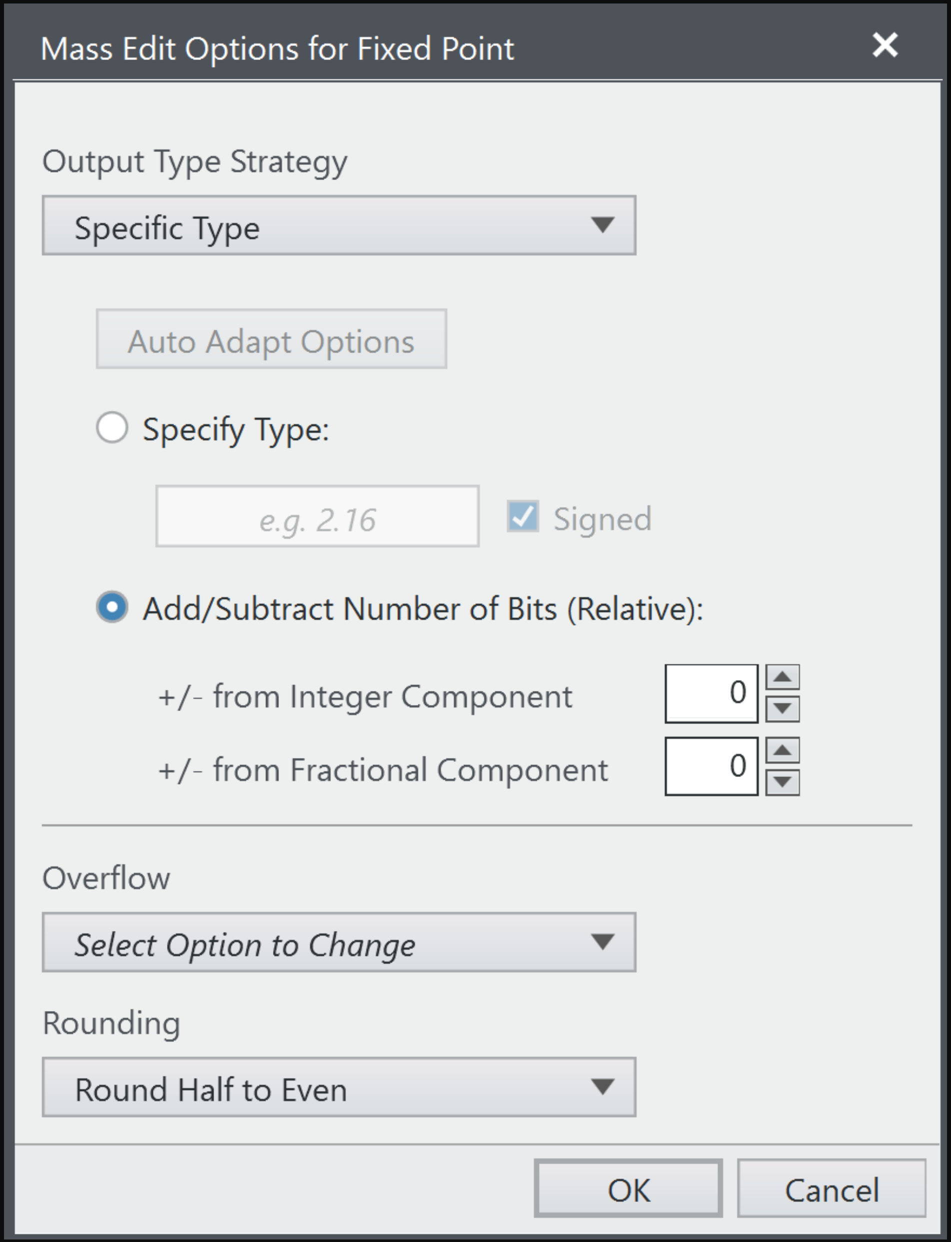
# CORRECT ENOUGH BY CONSTRUCTION



Strategy: SNR (dB) 10

| File Name | Object | Type *(Initial Suggestion)* | SNR | Overflow | Underflow |
|---|---|---|---|---|---|
| Second Order IIR Filter - FXP.gvi | X | I4<2.2> | 18.1 | 0% | 11.4% |
| Second Order IIR Filter - FXP.gvi | Reverse Coefficients | I3<2.1> | 11.16 | 0% | 33.3% |
| Second Order IIR Filter - FXP.gvi | Forward Coefficients | U2<-1.3> | 16.36 | 0% | 0% |
| Second Order IIR Filter - FXP.gvi | Numeric Constant | U1<0.1> | +Inf | 0% | 0% |
| Second Order IIR Filter - FXP.gvi | Numeric Constant | U1<0.1> | +Inf | 0% | 0% |
| Second Order IIR Filter - FXP.gvi | Multiply | I3<-1.4> | 15.55 | 5.92% | 11.4% |
| Second Order IIR Filter - FXP.gvi | Multiply | I3<0.3> | 12.96 | 0% | 11.4% |
| Second Order IIR Filter - FXP.gvi | Multiply | I3<-1.4> | 15.55 | 5.92% | 11.4% |
| Second Order IIR Filter - FXP.gvi | Add | I4<1.3> | 11.32 | 0% | 21.9% |
| Second Order IIR Filter - FXP.gvi | Add | I4<1.3> | 10.51 | 0% | 20% |
| Second Order IIR Filter - FXP.gvi | Subtract | I5<2.3> | 3.56 | 0% | 17.6% |
| Second Order IIR Filter - FXP.gvi | Multiply | I4<0.4> | -1.92 | 3.8% | 17.6% |
| Second Order IIR Filter - FXP.gvi | Multiply | I4<-1.5> | 0 | 0% | 99.9% |
| Second Order IIR Filter - FXP.gvi | Add | I5<0.5> | -7.81 | 0% | 17.6% |


Reverse coefficients path
×
l4<0.4> : -1.92
×
l4<-1.5> : 0
+
l5<0.5> : -7.81
−
l5<2.3> : 3.56



Mass Edit Options for Fixed Point

Output Type Strategy

Specific Type ▼

○ Specify Type:

e.g. 2.16 ☑ Signed

● Add/Subtract Number of Bits (Relative):

+/- from Integer Component: 0

+/- from Fractional Component: 0

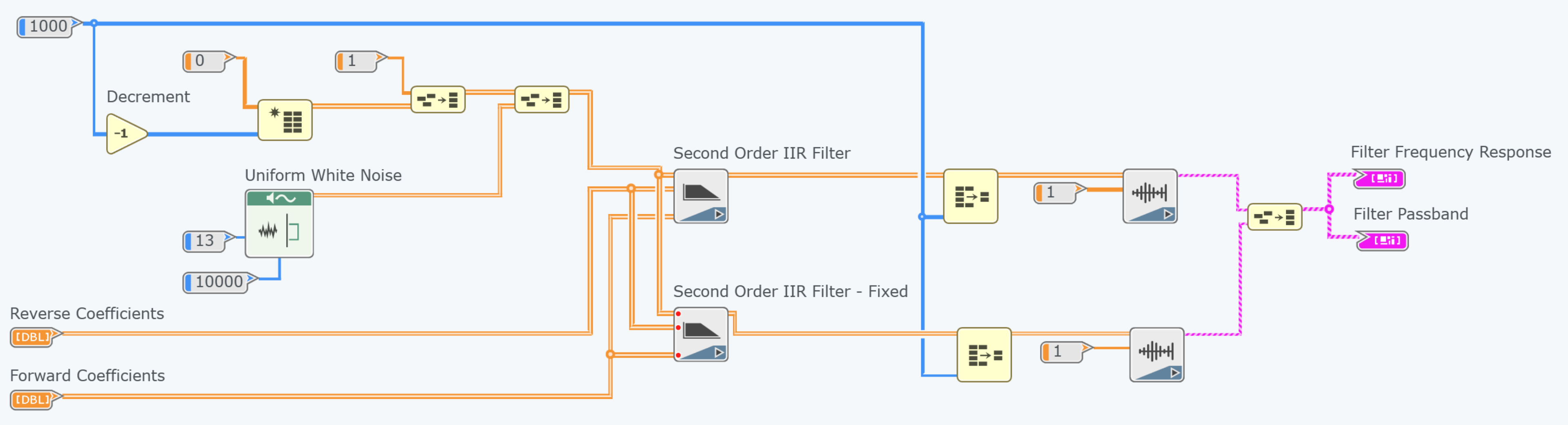
Overflow

*Select Option to Change* ▼

Rounding

Round Half to Even ▼

$$T = T_{FXP1}, T_{FXP2} \ldots T_{FXPn}$$



# EXECUTABILITY ALLOWS FOR CONTINUAL FEEDBACK

# CONCLUSION

## Our F2F tool in LabVIEW NXG FPGA:

1. Lets algorithm designers model their new algorithms in LabVIEW
2. Uses the executable nature of LabVIEW to generate an initial model transform
3. Provides tools to help designers create and apply the remaining transforms
4. Gives constant feedback on “correct enough” by construction