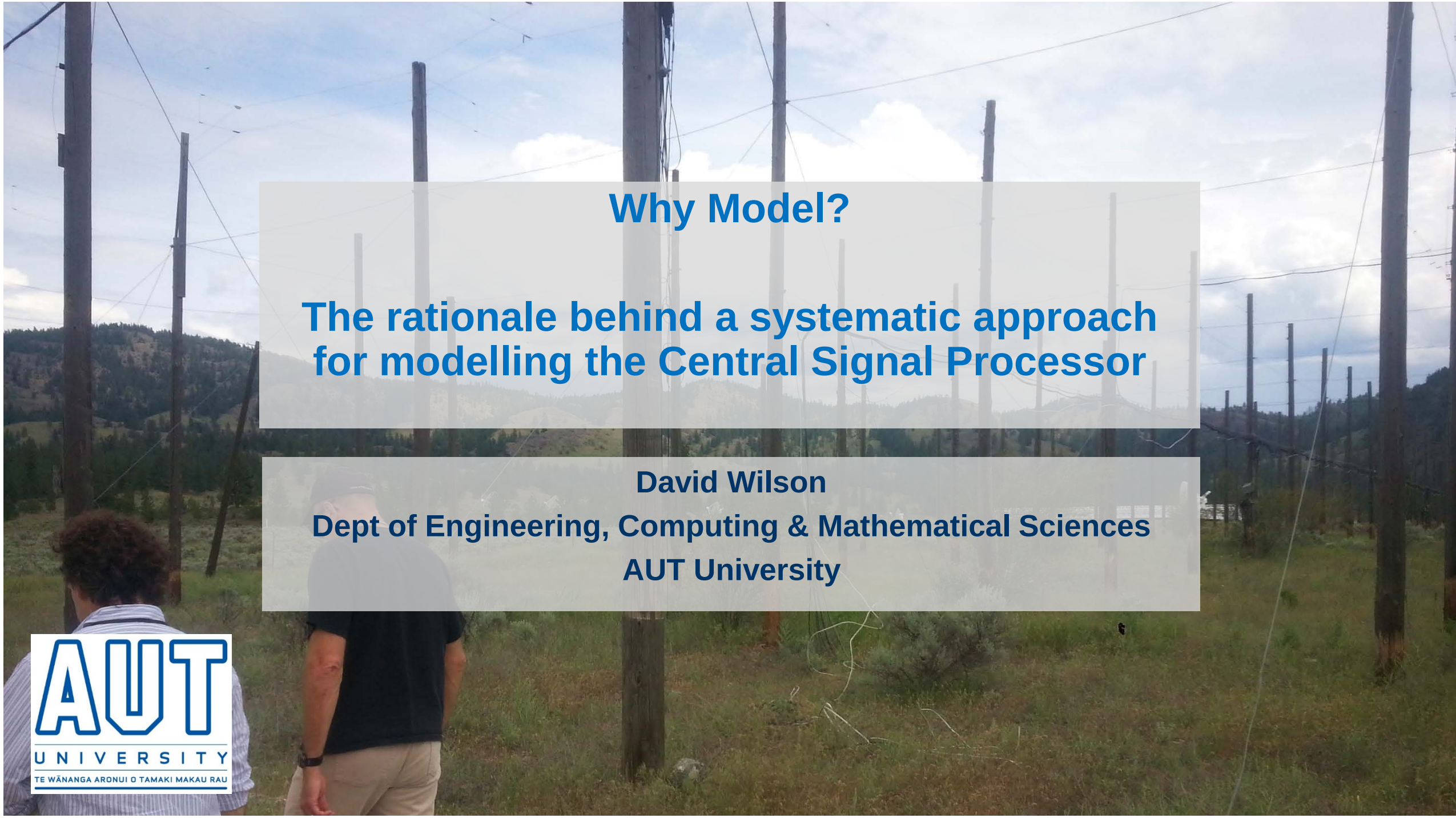


Why Model?

The rationale behind a systematic approach for modelling the Central Signal Processor

David Wilson

Dept of Engineering, Computing & Mathematical Sciences

AUT University

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Why we need a model

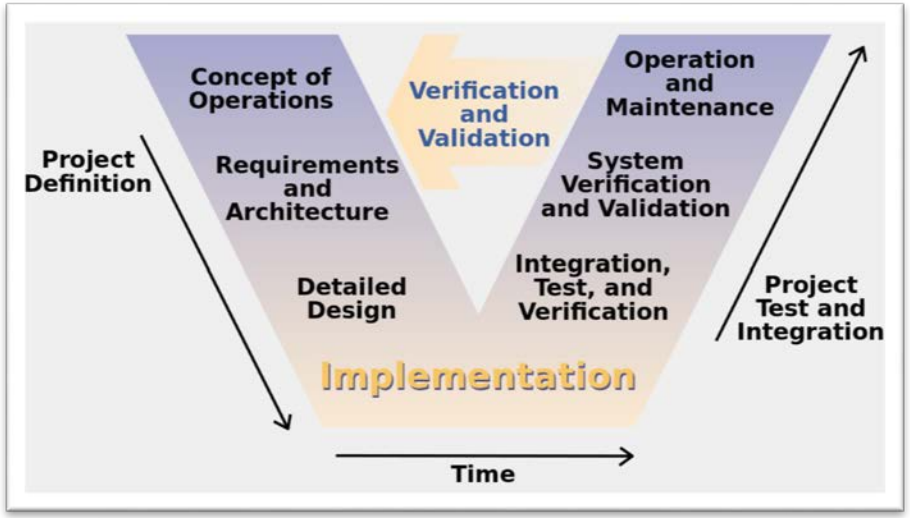
To show the algorithm works

To justify that the simplifications are reasonable

Assist the low-level hardware designers

Assist in debugging

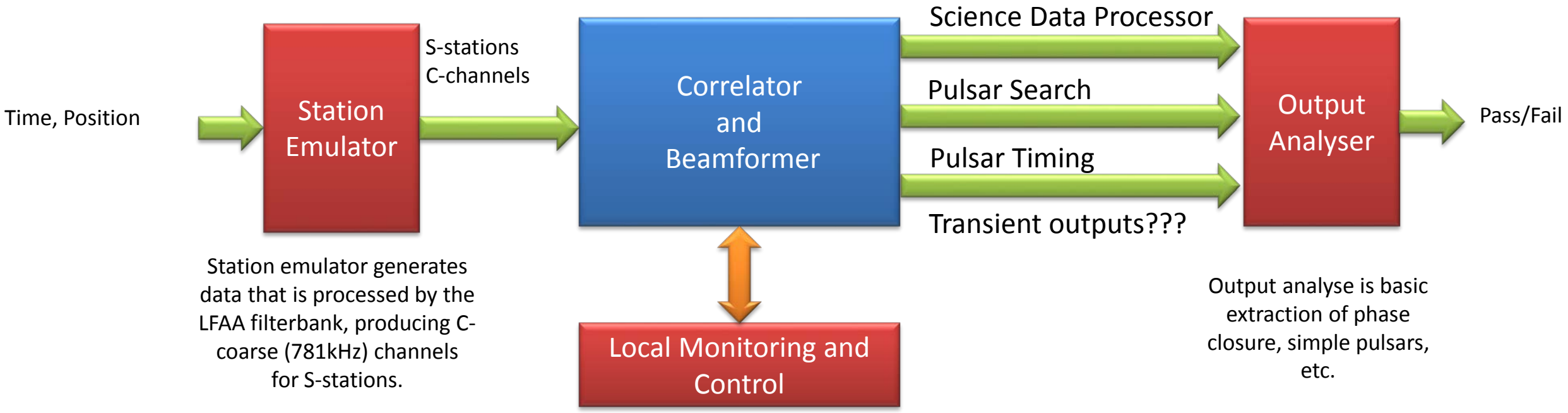
Check cooling & thermal stresses



Key considerations

- **Models at a suitably high-level abstraction**
- **Models are parameterised**
  - **Easy to adjust as req'mts change**
  - **Proof of concept**
- **Using Simulink/Matlab**
- **Built from components**
  - **Allow "plug & play"**
  - **"Model variants" in Simulink & projects**

Basic block diagram of the Central Signal Processor

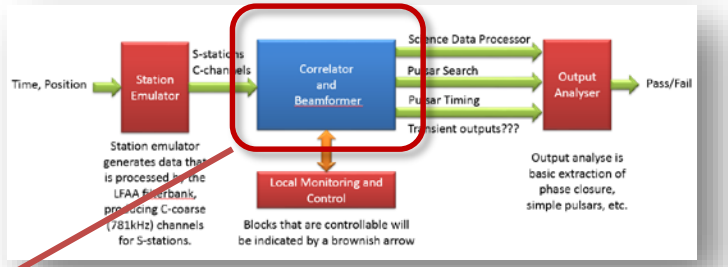


Station emulator generates data that is processed by the LFAA filterbank, producing C-coarse (781kHz) channels for S-stations.

Output analyse is basic extraction of phase closure, simple pulsars, etc.

Blocks that are controllable will be indicated by a brownish arrow

Level 1 – Major Blocks of LOW CBF



Each station cross connect output is all Stations for one frequency channel.

S-stations
C-channels

Fine Filterbanks (Cor,PSS,PST)

N-stations
Fc-channels
Fs-channels
Ft-channels

Station Cross Connect

Correlator and Beamformer

Frequency Aggregator

Science Data Processor

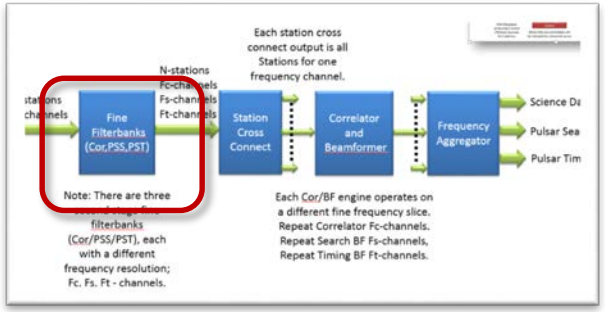
Pulsar Search

Pulsar Timing

Note: There are three second stage fine filterbanks (Cor/PSS/PST), each with a different frequency resolution;
Fc, Fs, Ft - channels.

Each Cor/BF engine operates on a different fine frequency slice.
Repeat Correlator Fc-channels.
Repeat Search BF Fs-channels,
Repeat Timing BF Ft-channels.

Level 2a – Filterbanks



Re-arrangement of multi-station/frequency data into long strings of the same data for FFB's

All filterbanks operate on the same input data, output CS or OS

Fine time delays of the order of ~1ps

Fixed Fine blanking

S-stations
C-channels

Doppler Correction

Corner Turner

Coarse Time Delay

Correlator Fine Filterbank

Search Fine Filterbank

Timing Fine Filterbank

Fine Time Delay

Fine Time Delay

Fine Time Delay

RFI Flagging

N-stations
Fc-channels
Fs-channels
Ft-channels

Realtime RFI Detection

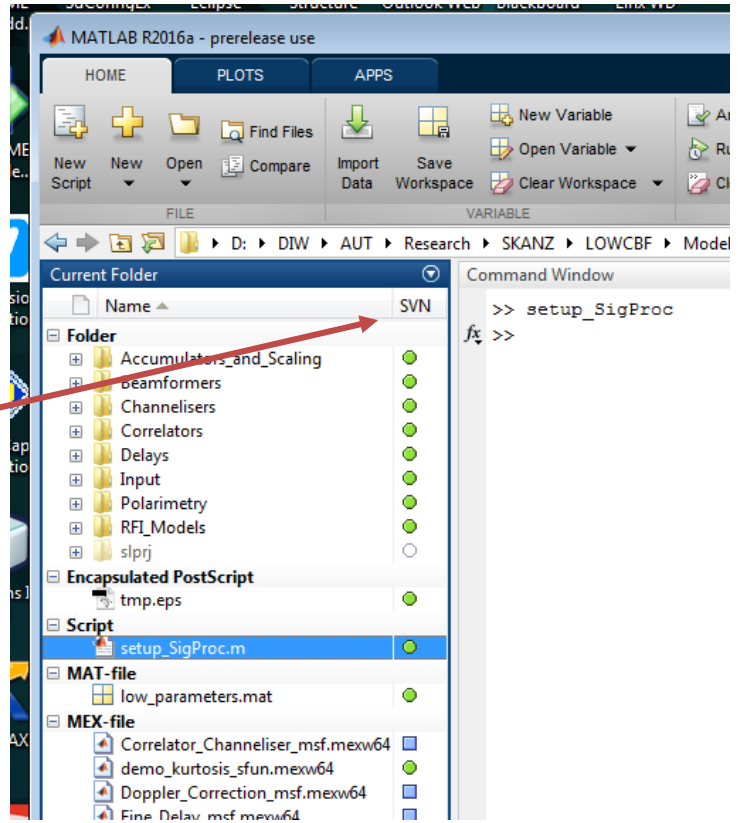
Shift ~1MHz wide frequency channels up to several Hz

Coarse time delays of the order ~1us

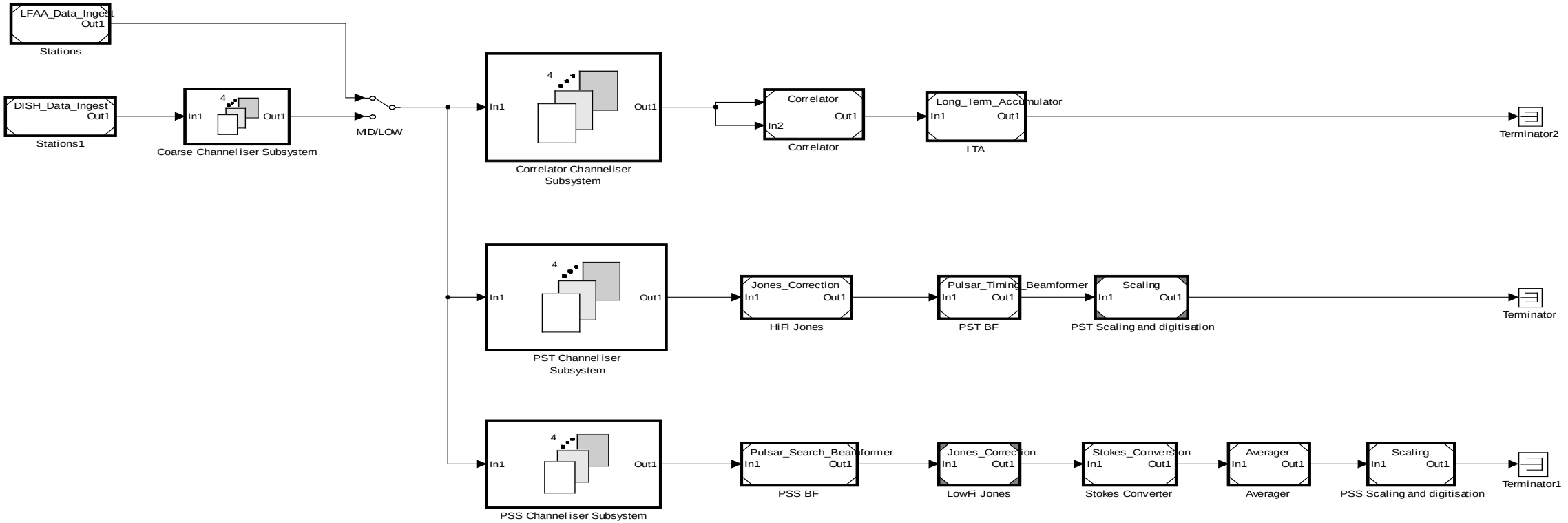
Signal flow model

- Constructed in Simulink & Matlab
- Well suited to a block diagram approach
- Simulink well suited for operating in “frames”
- Possible to setup for group working (version & control)

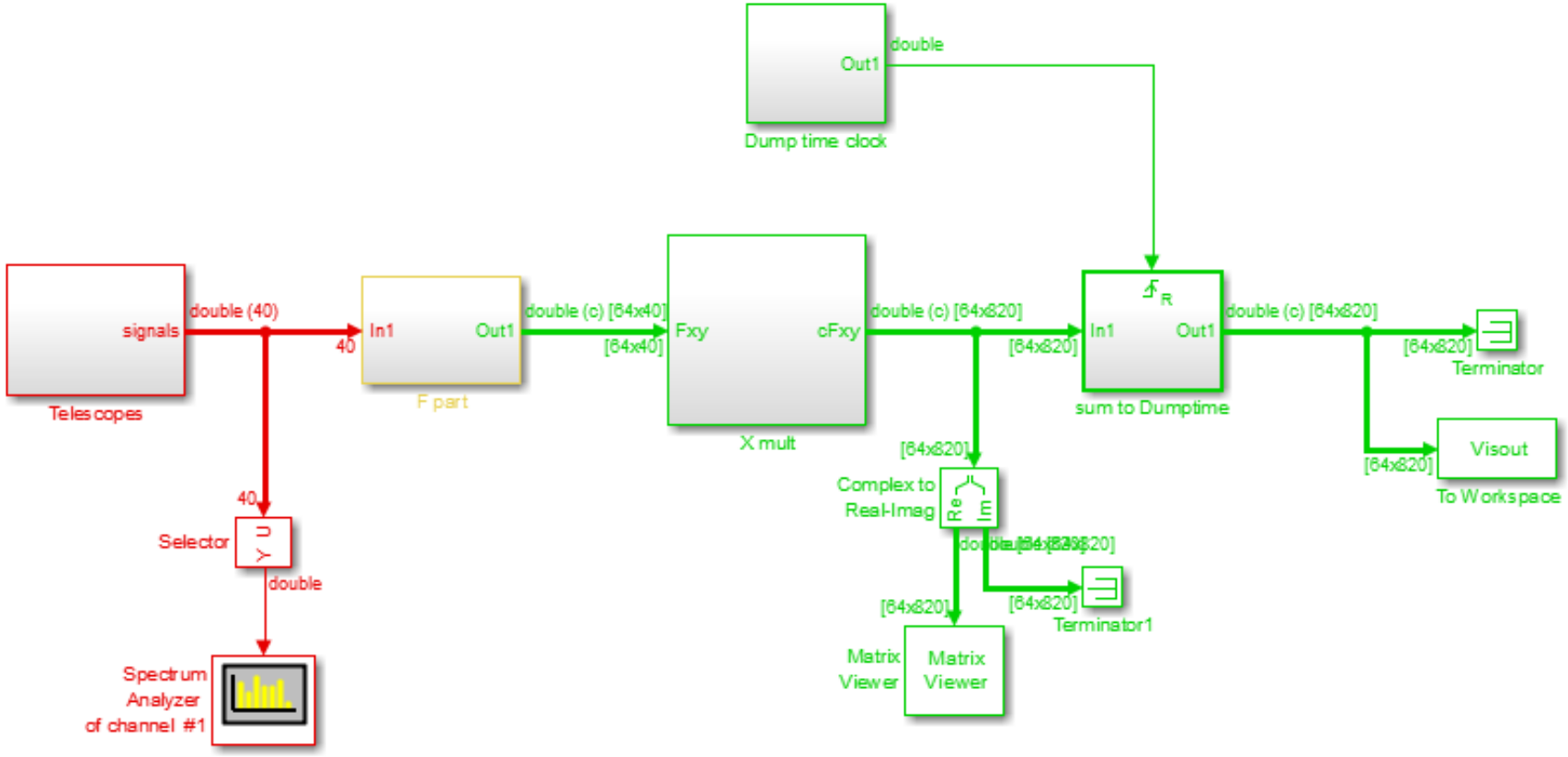
- Expensive
- Awkward with multiple software versions



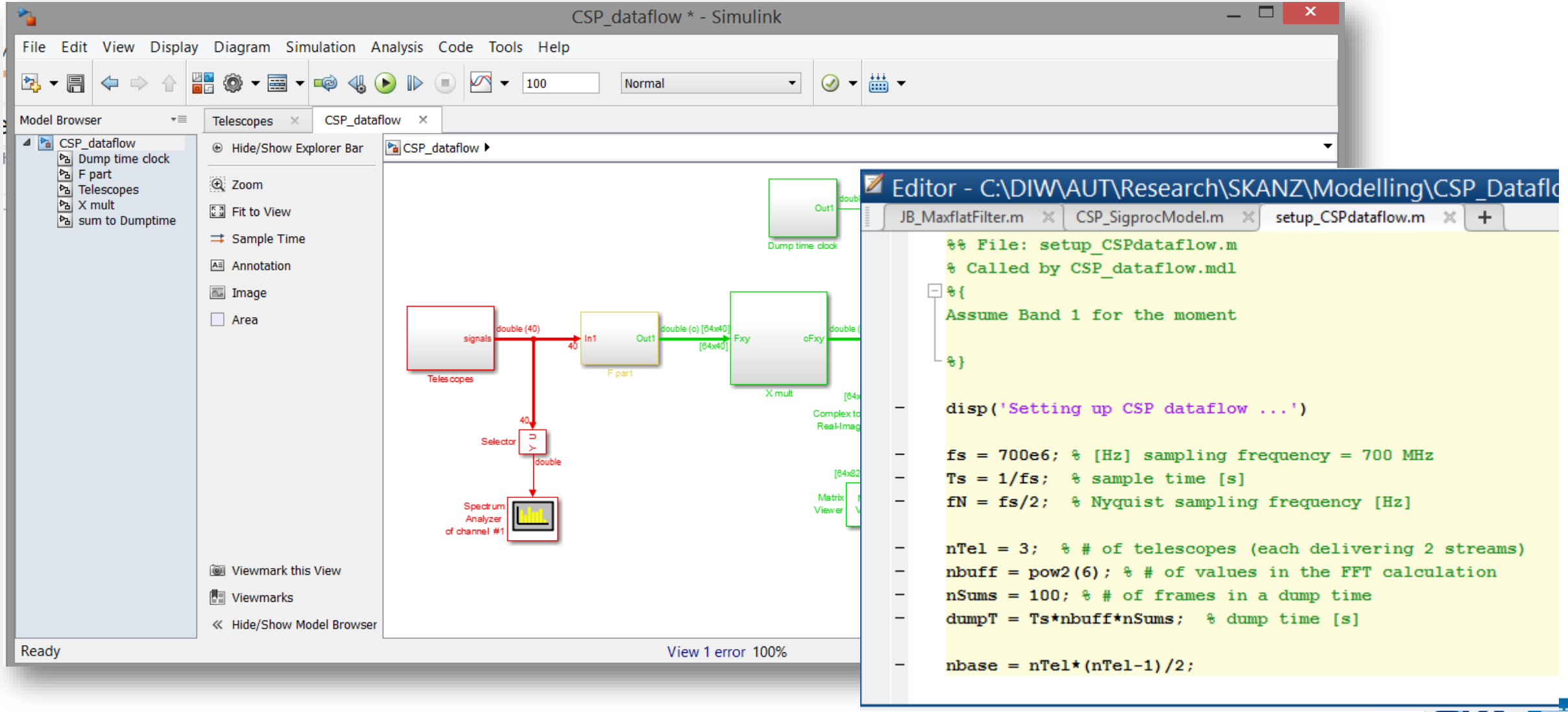
CSP Signal flow model



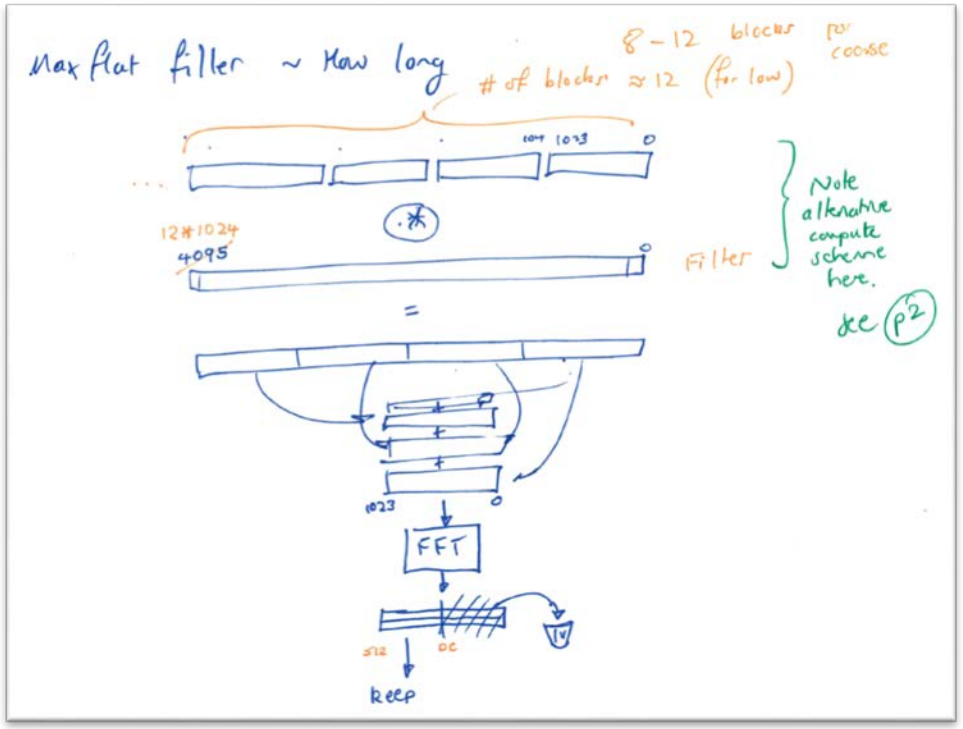
CSP Signal processing model

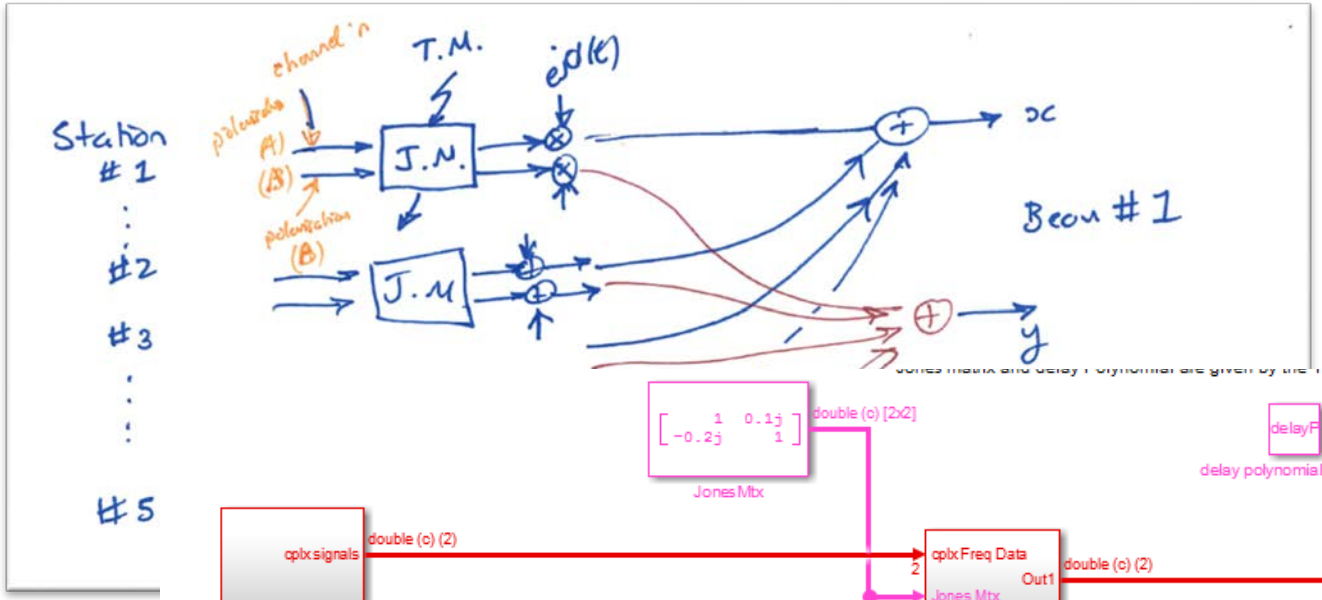


Components


CSP_dataflow * - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
Model Browser
Telescopes
CSP_dataflow
CSP_dataflow
Dump time clock
F part
Telescopes
X mult
sum to Dumptime
Hide/Show Explorer Bar
Zoom
Fit to View
Sample Time
Annotation
Image
Area
Viewmark this View
Viewmarks
Hide/Show Model Browser
signals
double (40)
In1
Out1
Fxy
cFxy
Selector
Spectrum Analyzer of channel #1
Ready
View 1 error 100%
Editor - C:\DIW\AUT\Research\SKANZ\Modelling\CSP_Dataflo
JB_MaxflatFilter.m
CSP_SigprocModel.m
setup_CSPdataflow.m
%% File: setup_CSPdataflow.m
% Called by CSP_dataflow.mdl
%{
Assume Band 1 for the moment
%}
disp('Setting up CSP dataflow ...')
fs = 700e6; % [Hz] sampling frequency = 700 MHz
Ts = 1/fs; % sample time [s]
fN = fs/2; % Nyquist sampling frequency [Hz]
nTel = 3; % # of telescopes (each delivering 2 streams)
nbuff = pow2(6); % # of values in the FFT calculation
nSums = 100; % # of frames in a dump time
dumpT = Ts*nbuff*nSums; % dump time [s]
nbase = nTel*(nTel-1)/2;

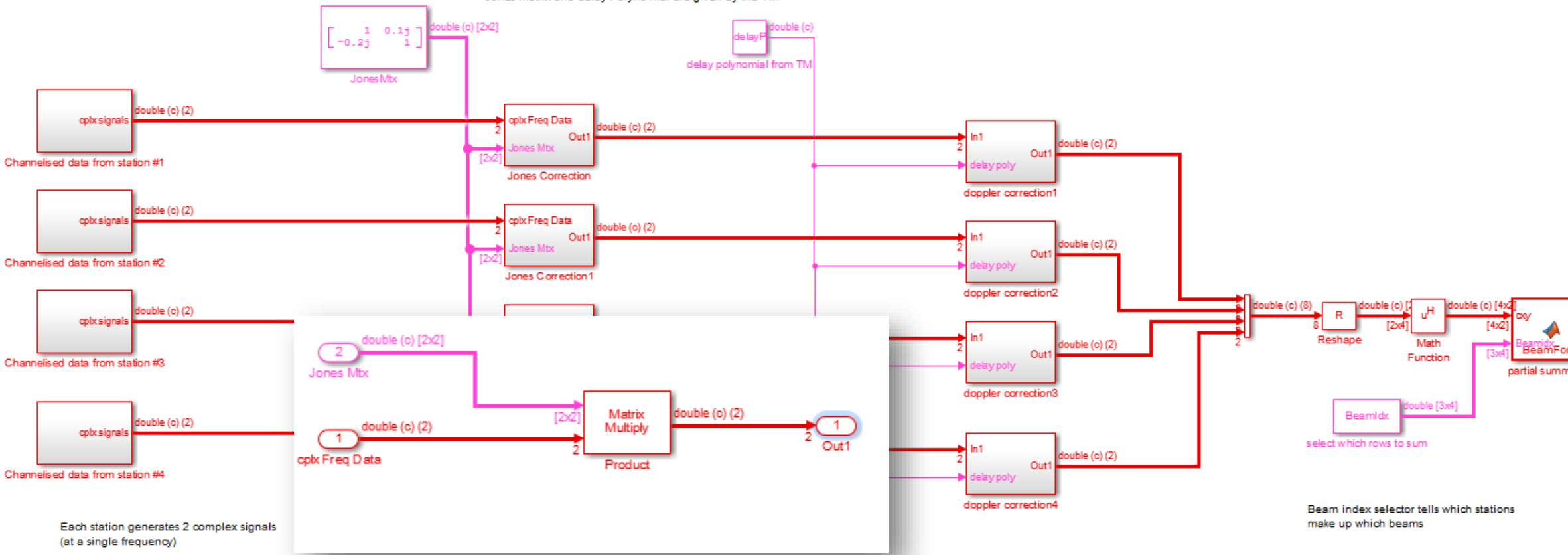
Prototyping: From sketch to block diagrams



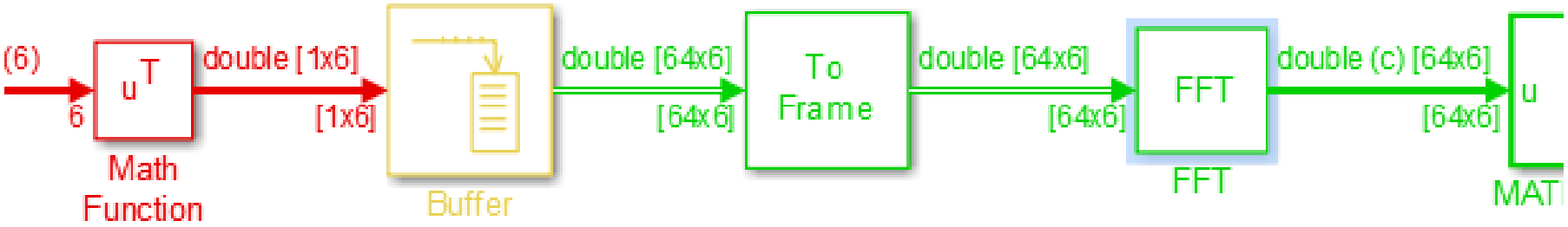


Proto-typing the *Jones Matrix* correction & Doppler correction for the Pulsar timing beamformer.

Crude implementation so far



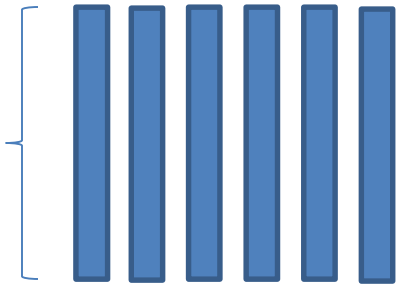
Frame-based processing



6 independent signals

Can have overlap (oversampled)

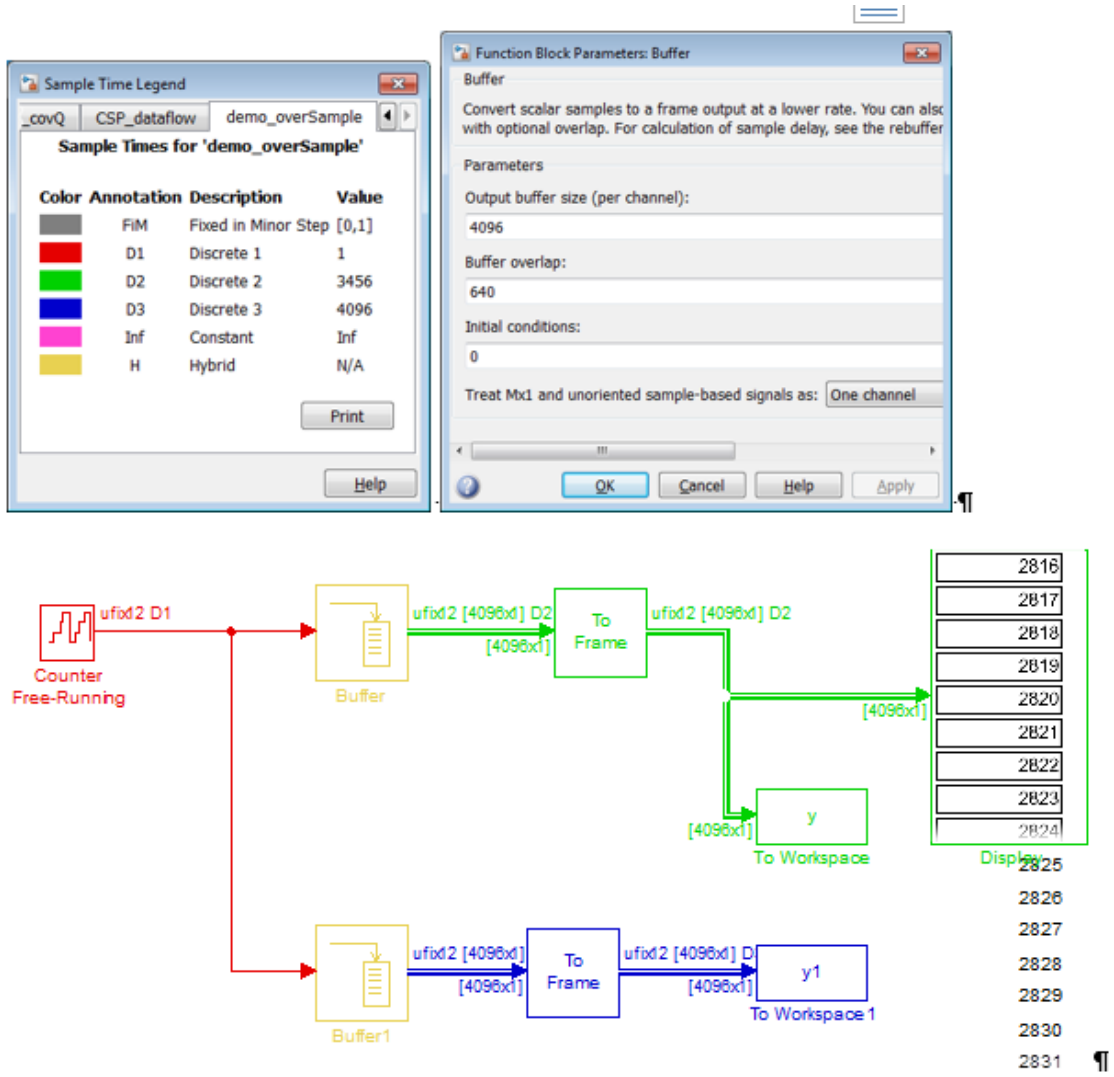
FFT operates all 6 columns



Oversampling

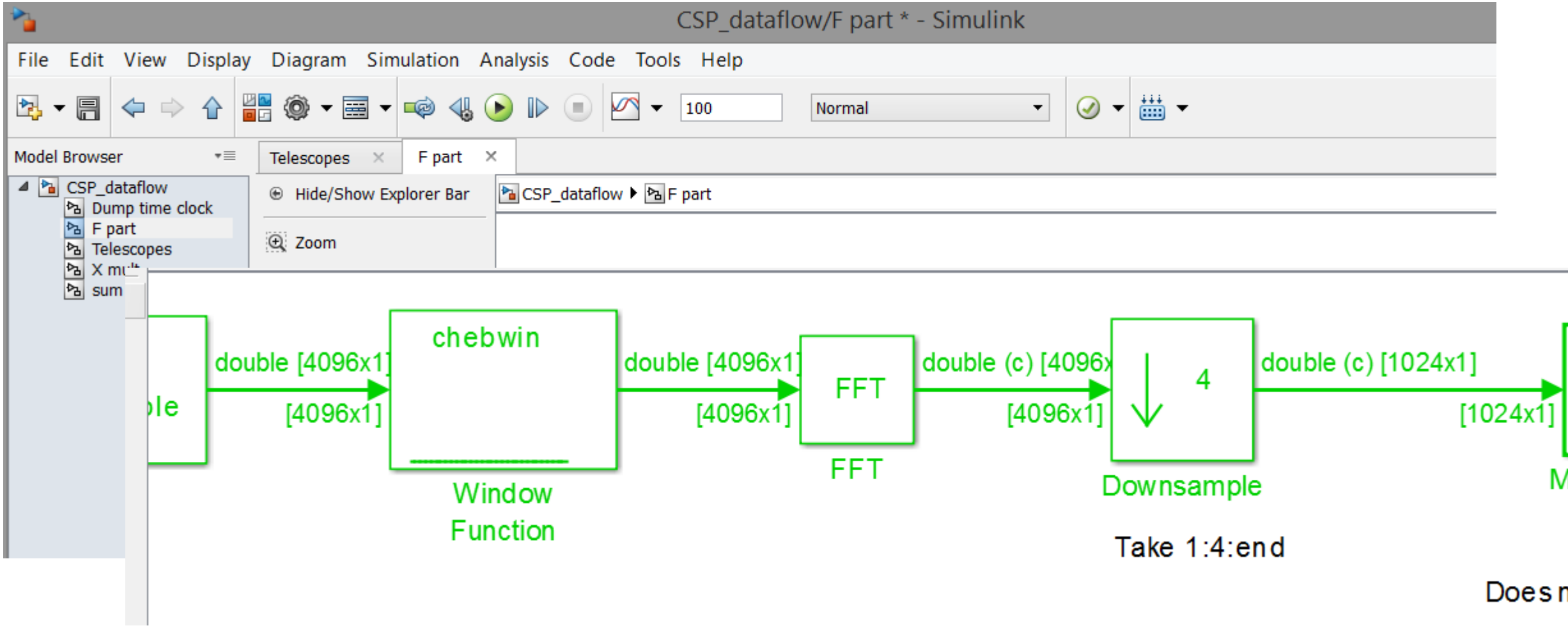
Oversampling is a key concept in the CSP.
Necessary to obtain the required freq. response.
We oversample at 27/32

Elegant & “mistake-free” using frames in Simulink

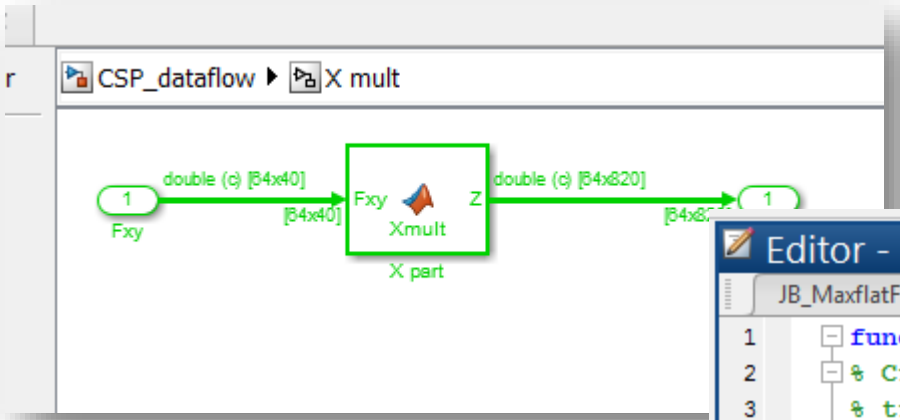


Figure·11:·Oversampling·at·27/32¶

F-part with “pure” FFT



X-part (embedded Matlab code)



Editor - Block: CSP_dataflow/X mult/X part

JB_MaxflatFilter.m | CSP_SigprocModel.m | setup_CSPdataflow.m | X mult/X part

```
1  function Z   = Xmult(Fxy)
2  % Cross multiply all columns in Fxy with each other, only keep upper
3  % triangular result.
4  % Output must be connected to something that is complex.
5
6
7  [nr,nc] = size(Fxy);
8
9  % Now cross correlate every pair (n*(n-1)/2)
10 no = nc*(nc+1)/2; % include diagonals
11 Z = complex(zeros(nr,no),0); % initialise as complex
12
13 k=0; % counter (count along rows on top triangle)
14 for i=1:nc
15     a = Fxy(:,i);
16     for j=i:nc
17         k = k+1; % increment counter
18         b = Fxy(:,j);
19         Z(:,k) = a.*b ; % complex multiply
20     end % for j
```

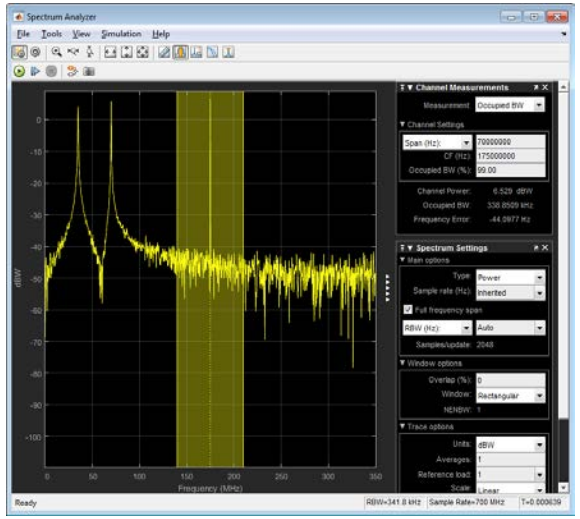
Command Window

Previously, this was inefficient & limited

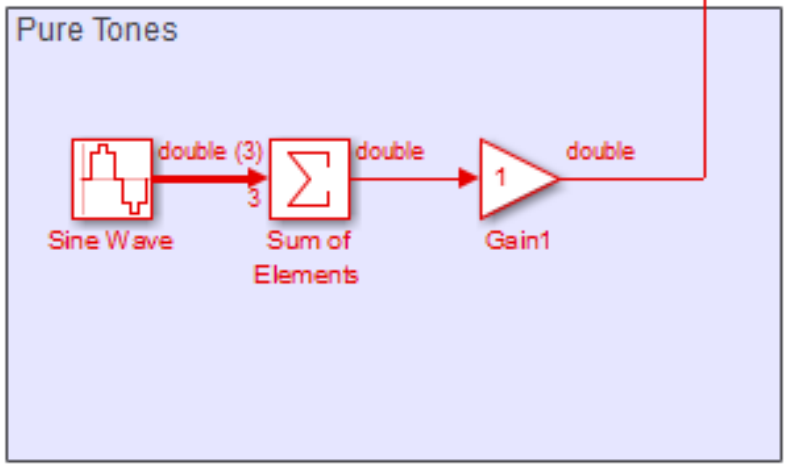
Now, just awkward and subtly different from (raw) Matlab

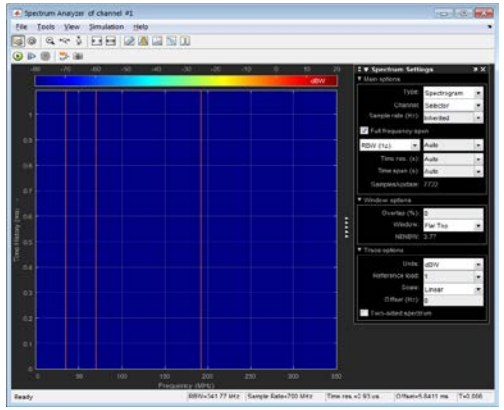
Incoming signal generators

Uncorrelated Gaussian random noise (sampled)

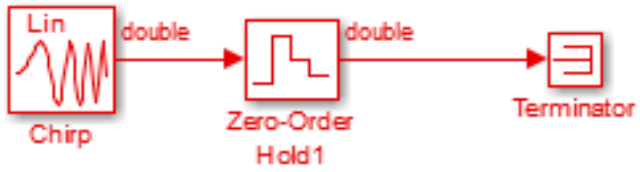


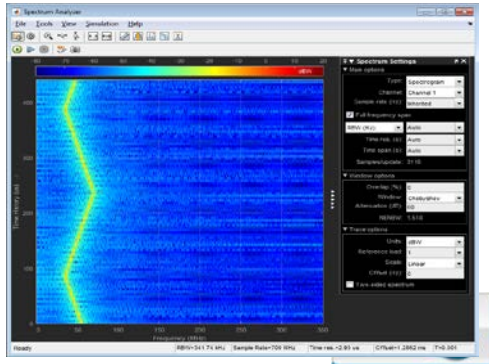
Various pure tones (summed)





Time varying frequencies (disabled)





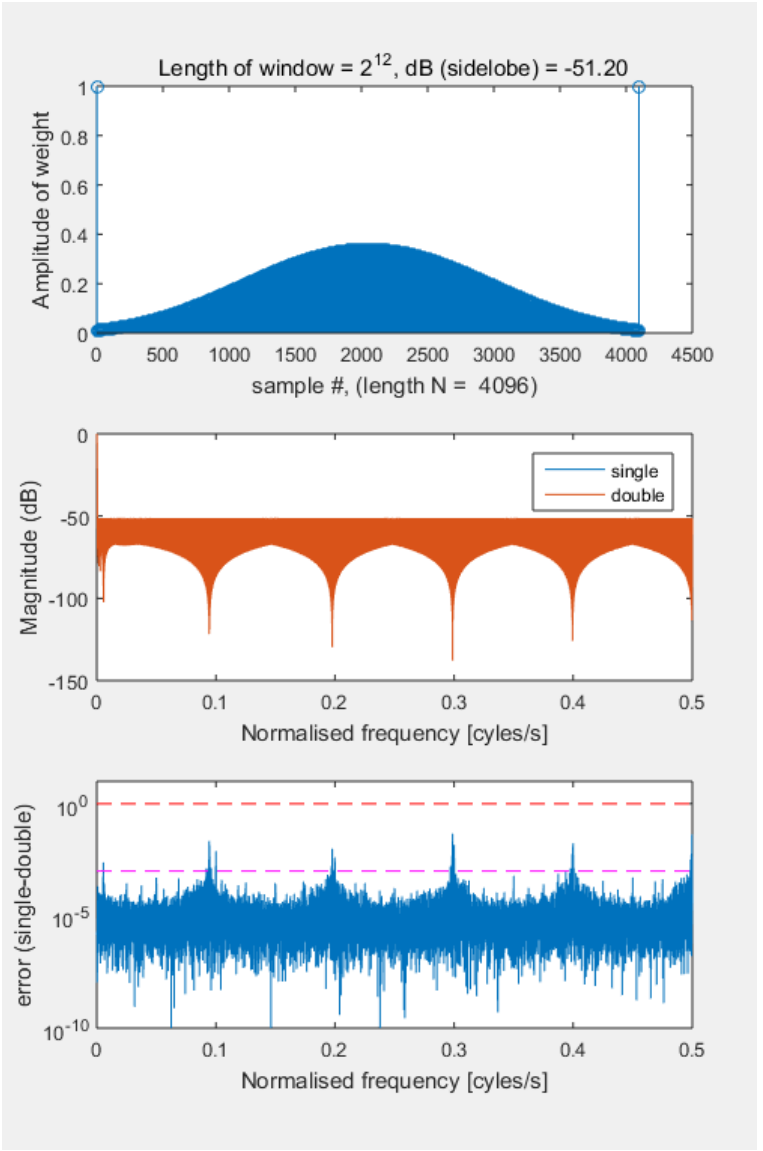
Numerical Precision testing

How gracefully does the algorithm degrade with limited precision?

Double vs single

```
%% Now try fixed point
L = pow2(9); % length of windows
alpha = 2.56; % value from Gianni
W = chebwin(L, 20*alpha);
sW = single(W); % convert offline designed va
fW = fi(W,1,8); % convert to fixed pt

N = pow2(7);
freqz(single(fW),1,N)
```



Numerical Precision testing

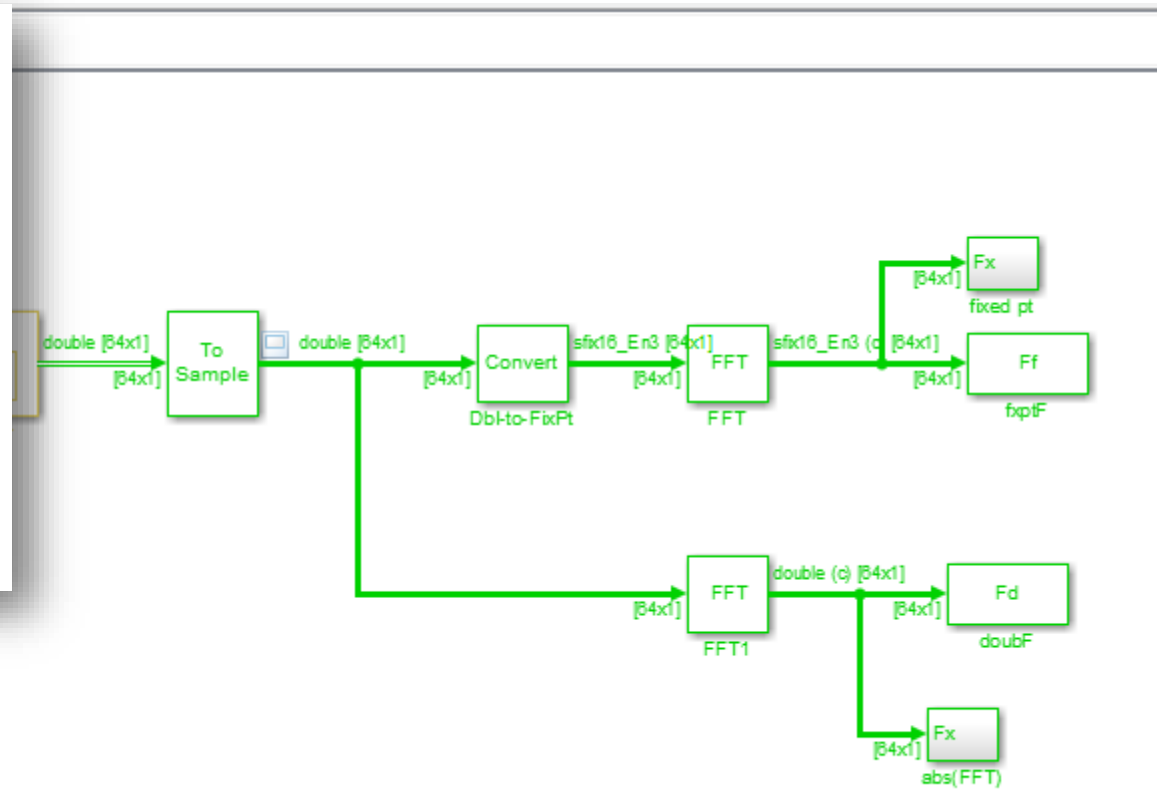
Fixed point simulations: Requires care & iterative design

```
>> help fixedpoint
  Fixed-Point Designer
  Version 5.0 (R2015a) 09-Feb-2015

  MATLAB Functionality
    Data and Data Types      - Fixed-Point Data and Data Types
    Math                     - Fixed-Point Math
    Float-to-Fixed Workflow  - Float-to-Fixed Workflow
    Code Acceleration        - Fixed-Point on Simulink Platform

  Simulink Functionality
    Data Types and Math      - Fixed-Point Data Types and Math
    Float-to-Fixed Workflow  - Float-to-Fixed Workflow

    Other functions named fixedpoint
```

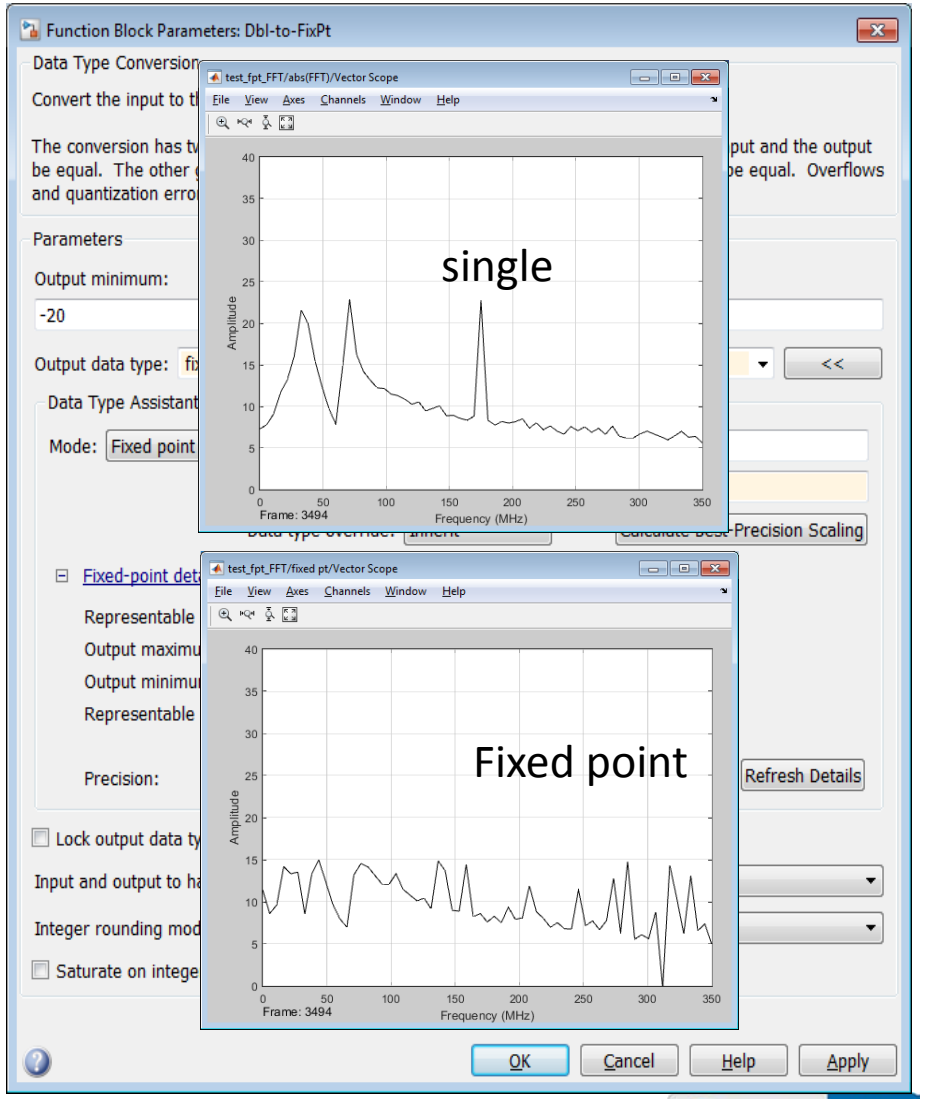


Comparison of single & fixed point

1. Double “Golden model”
2. Single (good for testing)
3. Fixed point (various)

Issues with:

- Saturation, RFI flagging



Our models

- **Easy to spawn variants**
  - **Add “convert” blocks early**
  - **Propagate types**
- **Easy to concatenate models**
  - **ICDs hardly necessary**
- **Convenient for block processing**
  - **But hard for exception handling**

IP spinoffs

- Taylor the “FFT”
  - faster, more efficient … but …
- Fixed pt & shorten the FPGA development
- Develop our local capability
- Thermal modelling

Wrap up

Models demonstrate/validate the algorithm

Assist with hardware programming/debugging

Allows scenario testing

“Golden model” encapsulates unambiguously the algorithm

But

Many FPGA issues not modelled

Corner turners, memory limitations, IP blocks


NRC·CNRC
Canada
SKA
MINEX
NRC·CNRC