Formal High-Level Model of a Radar Signal Processing System

George Ungureanu[1] *Anders Åhlander*[2] *Timmy Sundström*[3] *Ingo Sander*[1] *Ingemar Söderquist*[3]

[1]School of EECS, KTH Royal Institute of Technology, Stockholm, Sweden
[2]Business Area Surveillance, Saab AB, Gothenburg, Sweden
[2]Business Area Aeronautics, Saab AB, Linköping, Sweden

October 9, 2019

The Context of this Work

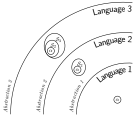
NFFP7: CORRECT

- Saab AB & KTH
- current industrial design flows do not have a clear path from the functional specification to product
- information on security, safety, behavior are important aspects to be *expressed* and *assessed*
  - identified by the industry
  - emphasized by certification authorities
- aim to decrease the need for late stage testing and related development costs.

Overview

Modeling with ForSyDe

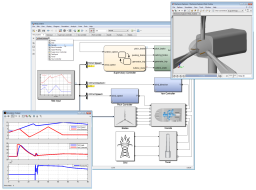
- push orthogonalization in modeling one step further
- harmoniously combine different DSLs capturing different aspects



The AESA Case Study

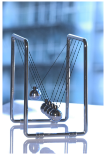
- Active Electronically Scanned Array Antenna
- typical high-performance component in avionics under real-time constraints
- *extremely* costly for civil applications *today*

Why We Use Models?

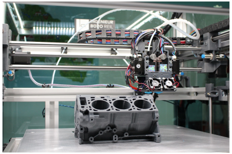


...to understand systems

$$v(t) = v(0) + \frac{1}{m}\int_0^t F(\tau)d\tau$$



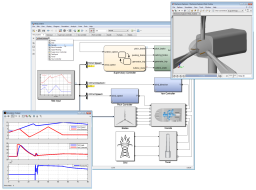
...to design systems



...to create systems

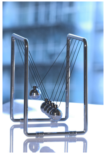
Why We Use Models?

To a scientist, the value of a model lies in how well its properties match those of a target (...). But to an engineer the value of an object lies in how well its properties match a model. A scientist asks, "Can I make a model for this thing?" An engineer asks, "Can I make a thing for this model?"[1]



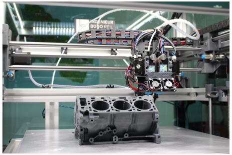
...to understand systems

$$v(t) = v(0) + \frac{1}{m}\int_0^t F(\tau)d\tau$$



...to design systems

...to create systems

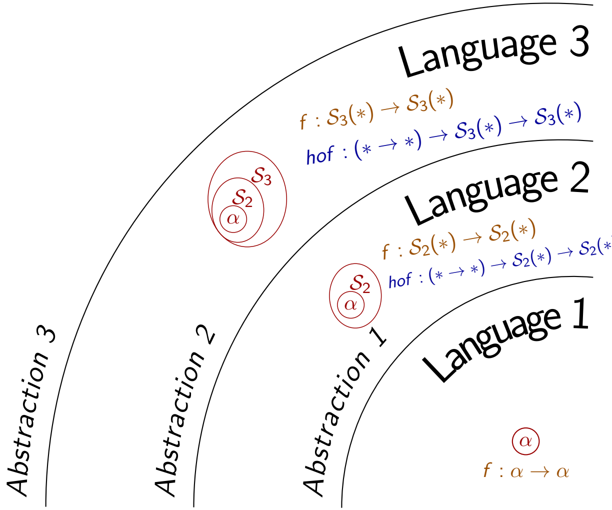


[1] Edward Ashford Lee. *Plato and the Nerd: The Creative Partnership of Humans and Technology*. MIT Press, 2017.

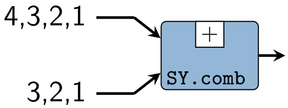
Layered Languages

Each layer consists of:

- a set of structured types
  $=$ *encode properties*
- a set of functions over these types
  $=$ *transformations, rules*
- a set of higher order functions (HOF)
  $=$ *conduits between layers*



A Language for Timed Behavior



SY process

Signals

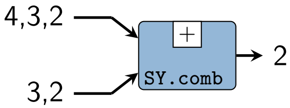
- encode temporal information
- define *tag systems*[2]

Processes

- act according to MoC semantics
- created with *process constructors* (HOF)

[2] E.A. Lee and A. Sangiovanni-Vincentelli. *A framework for comparing models of computation*. Dec. 1998.

A Language for Timed Behavior



SY process

Signals

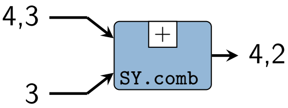
- encode temporal information
- define *tag systems*[2]

Processes

- act according to MoC semantics
- created with *process constructors* (HOF)

[2] Lee and Sangiovanni-Vincentelli, *A framework for comparing models of computation*.

A Language for Timed Behavior



SY process

Signals

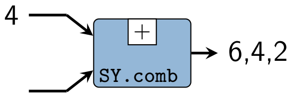
- encode temporal information
- define *tag systems*[2]

Processes

- act according to MoC semantics
- created with *process constructors* (HOF)

[2] Lee and Sangiovanni-Vincentelli, *A framework for comparing models of computation*.

A Language for Timed Behavior



SY process

Signals

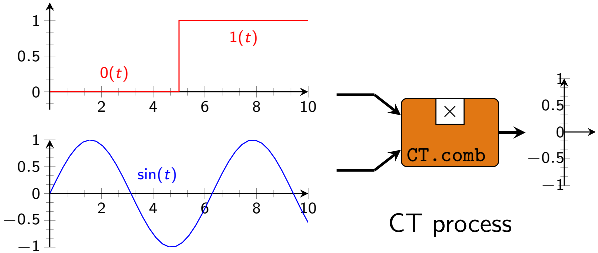
- encode temporal information
- define *tag systems*[2]

Processes

- act according to MoC semantics
- created with *process constructors* (HOF)

[2] Lee and Sangiovanni-Vincentelli, *A framework for comparing models of computation*.

A Language for Timed Behavior



Signals

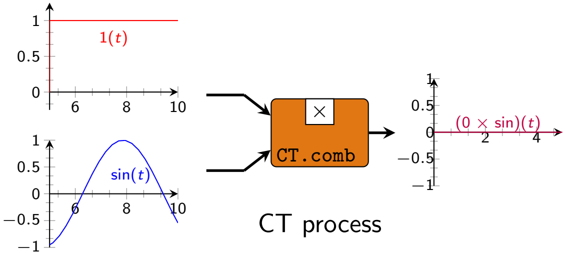
- encode temporal information
- define *tag systems*[2]

Processes

- act according to MoC semantics
- created with *process constructors* (HOF)

[2]Lee and Sangiovanni-Vincentelli, *A framework for comparing models of computation*.

A Language for Timed Behavior



Signals

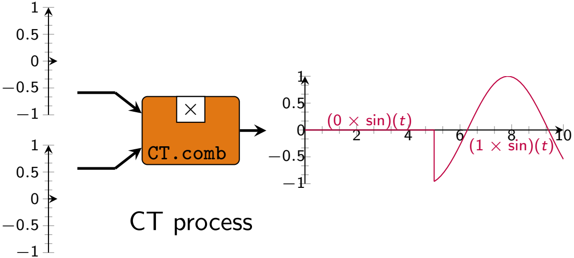
- encode temporal information
- define *tag systems*[2]

Processes

- act according to MoC semantics
- created with *process constructors* (HOF)

[2] Lee and Sangiovanni-Vincentelli, *A framework for comparing models of computation*.

A Language for Timed Behavior



Signals

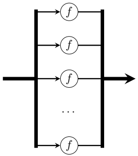
- encode temporal information
- define *tag systems*[2]

Processes

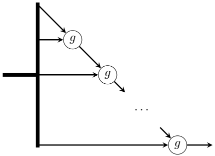
- act according to MoC semantics
- created with *process constructors* (HOF)

[2] Lee and Sangiovanni-Vincentelli, *A framework for comparing models of computation*.

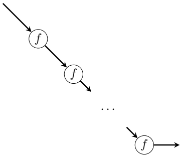
A Language for Structured Parallelism



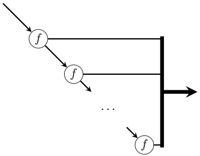
(a) `farm`



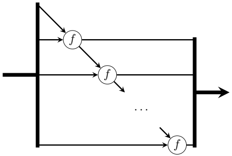
(b) `reduce`



(c) `pipe`



(d) `recur`



(e) `prefix`

Regular Structures (e.g. Vectors)

- encode spatial information
- enable *catamorphisms*[3]

Parallel Patterns

- functional relations between elements
- potential for parallel distribution
- created with *skeletons* (HOF)

[3] David B Skillicorn. *Foundations of parallel programming*. 6. Cambridge University Press, 2005.

A Language for Testing Properties

$$\textit{pre-condition} \Rightarrow \textit{statement}$$

Example:

$$\forall a \in List(\alpha) \Rightarrow \texttt{reverse}(\texttt{reverse}\ a) = a$$

Generators

- express pre-conditions
- abstract random data generators
- "smartened" by algebraic properties[4]

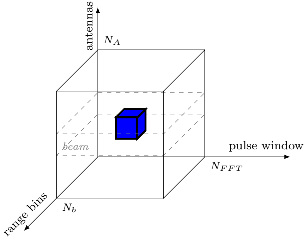
Properties & Combinators

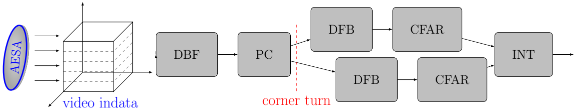
- test truth statements
- systematic composition of generators

[4] John Hughes. "QuickCheck testing for fun and profit". In: *International Symposium on Practical Aspects of Declarative Languages*. Springer. 2007.

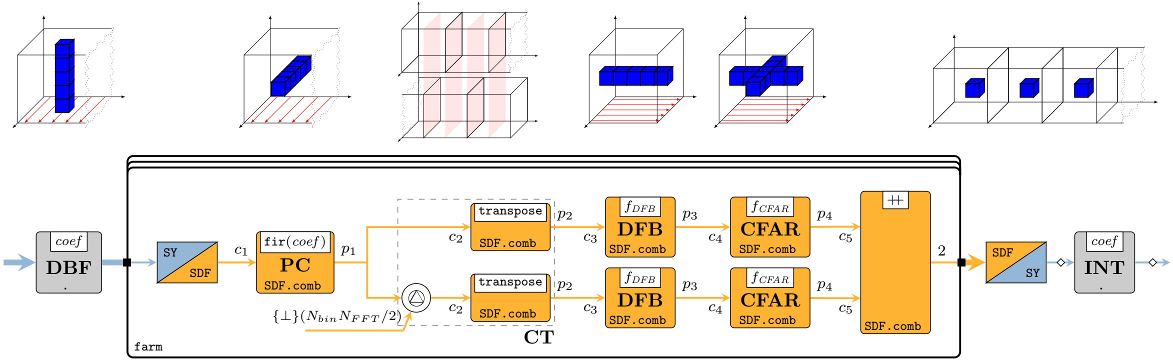
The AESA Radar

- typically thousands of antenna elements
- hundreds of GOPS
- when part of control loop, real-time aspect becomes critical





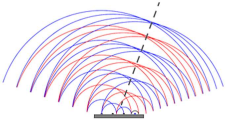
The AESA Signal Processing Chain as a Layered Model

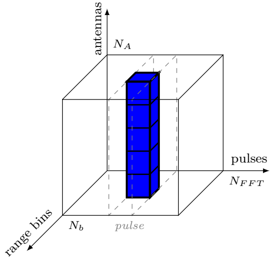


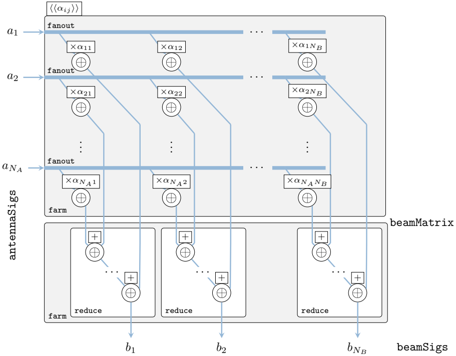
$c_1 = p_1 = N_b$; $c_2 = N_{FFT} \times N_b$; $p_2 = c_4 = p_4 = c_5 = N_b \times N_{FFT}$; $c_3 = p_3 = N_{FFT}$;

[4] ForSyDe-Atom code available at `https://github.com/forsyde/aesa-radar`

Digital Beamforming (DBF)





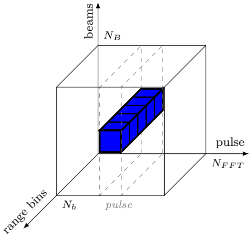


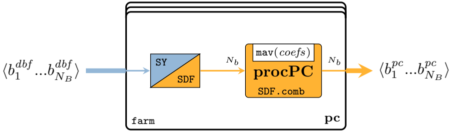
- forms "listening directions"
- vector-matrix multiplication

```haskell
dbf :: Vector (SY.Signal (Complex Float)) -> Vector (SY.Signal (Complex Float))
dbf antennaSigs = beamSigs
  where beamSigs   = reduce (farm21 (comb21 (+))) beamMatrix
        beamMatrix = farm21 (\c -> comb11 (*c)) beamConsts sigMatrix
        sigMatrix  = farm11 fanout antennaSigs
```

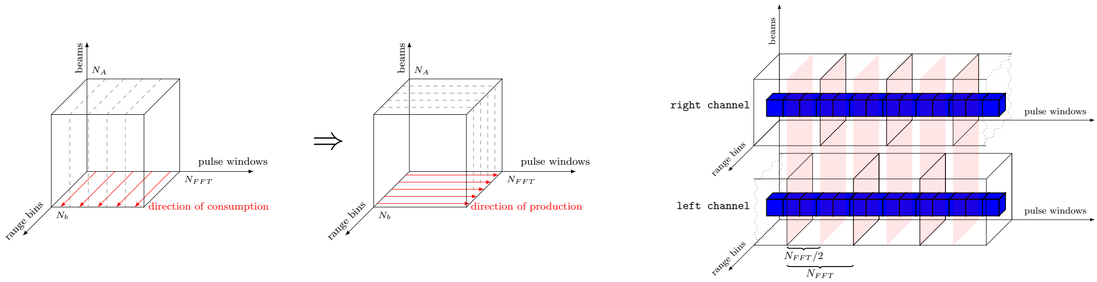
Pulse Compression (PC)

- echo is passed through a matched filter to decode its modulation
- basically a moving average/FIR filter

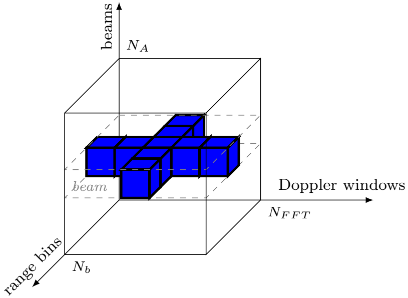




Corner Turn (CT) & Doppler Filter Bank (DFB)



Constant False Alarm Ratio (CFAR)

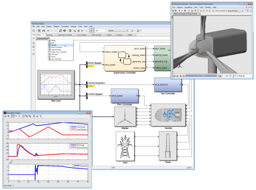


$$
\begin{cases}
CFAR(a_{ij}) = 2^{(5+\log_2 a_{ij}) - \max(EMV(a_{ij}), LMV(a_{ij}), MD(a_{ij}))} \\
EMV(a_{ij}) = \dfrac{1}{N} \displaystyle\sum_{k=0}^{N-1} \left(\log_2 \left(\frac{1}{4} \sum_{l=0}^{3} a_{(i-2-4k-l)j} \right) \right) \\
LMV(a_{ij}) = \dfrac{1}{N} \displaystyle\sum_{k=0}^{N-1} \left(\log_2 \left(\frac{1}{4} \sum_{l=0}^{3} a_{(i+2+4k+l)j} \right) \right) \\
MD(a_{ij}) = \log_2 \left(\min_{k=1}^{N} (a_{ik}) \right)
\end{cases}
$$

$$
\forall i \in [eb, lb], j \in [1, N] \text{ where } \begin{cases} N = N_{FFT} \\ eb = N_{FFT} + 1 \\ lb = N_b - N_{FFT} - 1 \end{cases}
$$

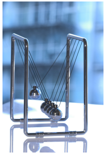
```haskell
fCFAR :: RealFloat r => Range (Window r) -> Range (Window r)
fCFAR rbins = V.farm41 (\ m -> V.farm31 (normCfa m)) md rbins lmv emv
  where
    md  = V.farm11 (logBase 2 . V.reduce min) rbins
    emv = (V.farm11 aritMean . V.stencil nFFT) rbins
    lmv = (V.drop 2 . V.farm11 aritMean . V.stencil nFFT) rbins
    normCfa m a l e = 2 ** (5 + logBase 2 a - maximum [l,e,m])
    aritMean = V.farm11 (/n) . V.reduce (+) . V.farm11 geomMean . V.group 4
    geomMean = V.farm11 (logBase 2 . (/4)) . V.reduce (+)
```

Why Have We Done This?



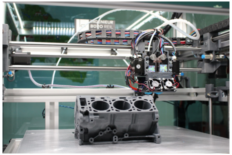
...to understand systems

$$v(t) = v(0) + \frac{1}{m}\int_0^t F(\tau)d\tau$$

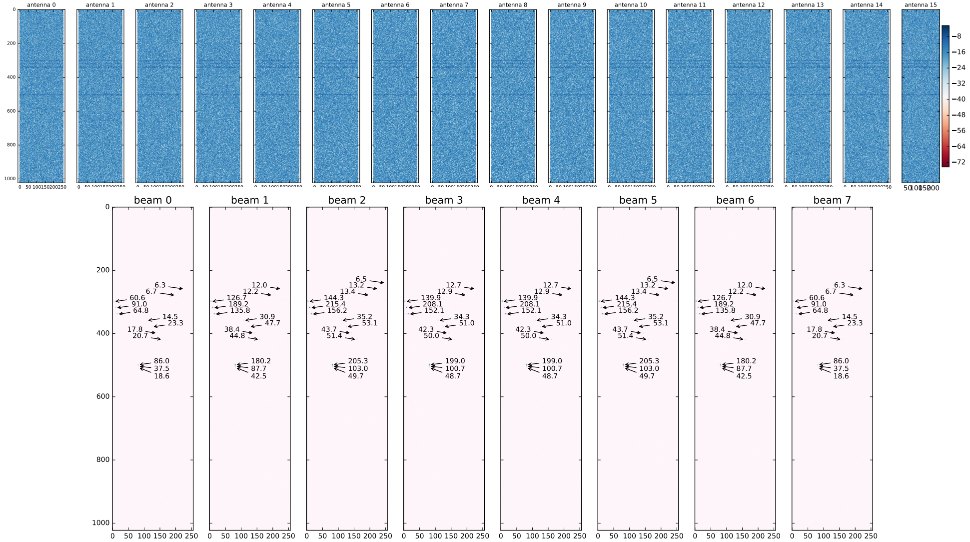


...to design systems

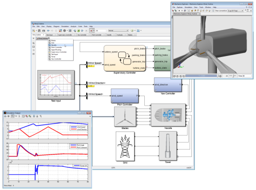
...to create systems



...To Understand the AESA Signal Processing System

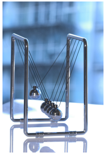


Why Have We Done This?



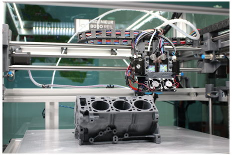
...to understand systems ✓

$$v(t) = v(0) + \frac{1}{m}\int_0^t F(\tau)d\tau$$



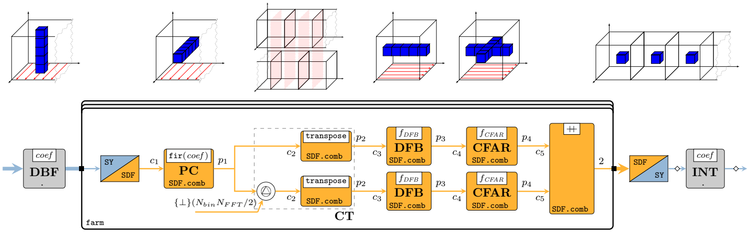
...to design systems

...to create systems



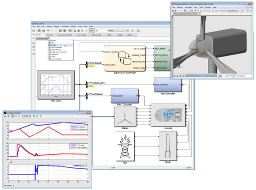
...To Design the AESA Signal Processing System

- system structure
- behavior particularities
- tested properties
- validated the design



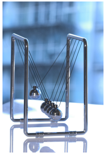
$c_1 = p_1 = N_b;\;\; c_2 = N_{FFT} \times N_b;\;\; p_2 = c_4 = p_4 = c_5 = N_b \times N_{FFT};\;\; c_3 = p_3 = N_{FFT};$

Why Have We Done This?



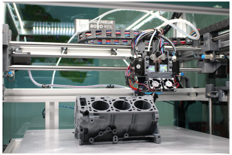
...to understand systems ✓

$$v(t) = v(0) + \frac{1}{m}\int_0^t F(\tau)d\tau$$

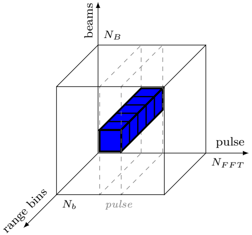


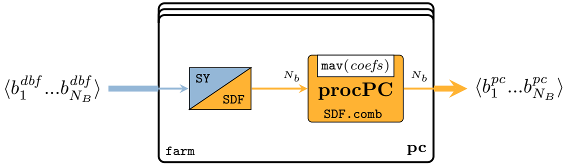
...to design systems ✓

...to create systems



Synthesizing the PC component to FPGA



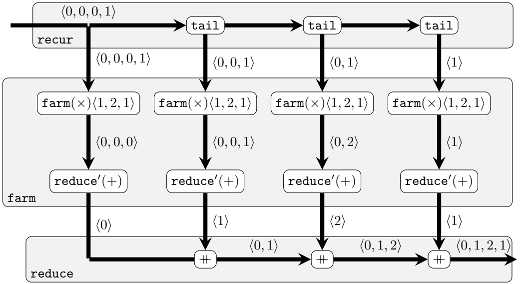


Given Specification

"For each antenna the data arrives *pulse by pulse*, and each pulse arrives *range bin by range bin*. This happens for all *antennas in parallel*, and all complex samples are synchronized with the *same sampling rate*, e.g. of the A/D converter.
(...)
The goal of the pulse compression is to collect all received energy from one target into a single range bin. The received echo of the modulated pulse is **passed through a matched filter**. Here, the matched filtering is done digitally. "

A Closer Look at the MAV Operation

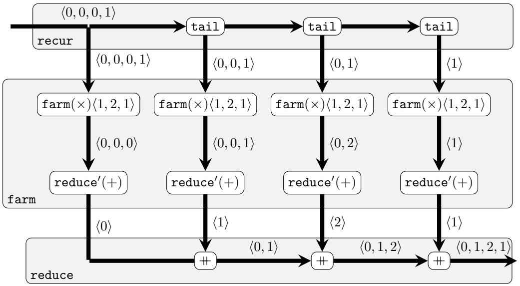
$$\texttt{reduce}(\ \mathbin{+\!\!+}\ ) \circ \texttt{farm}(\ \texttt{reduce}'(+) \circ \texttt{farm}(\times)\ c\ ) \circ \texttt{recur}(\ \texttt{tail}\ )$$



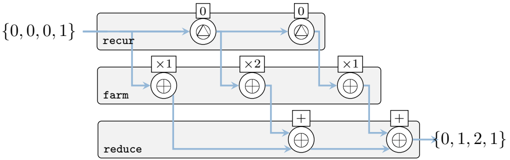
$$N_{\text{Farm}} = N_{\text{Input}} \times N_{\text{Coefficients}}$$

A Closer Look at the MAV Operation

$\texttt{reduce}(\ \texttt{comb}(+)\ ) \circ \texttt{farm}(\ c \mapsto \texttt{comb}(\times c)\ ) \circ \texttt{recur}(\ \texttt{delay}(0)\ )$

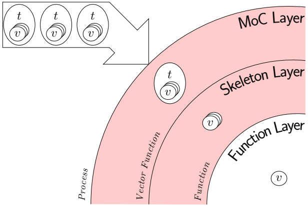


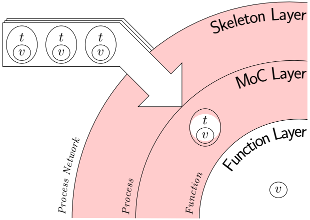
$$N_{\text{Farm}} = N_{\text{Input}} \times N_{\text{Coefficients}}$$

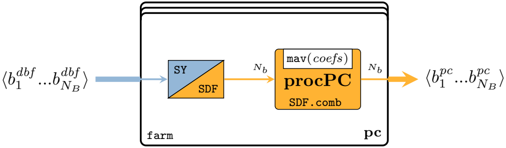


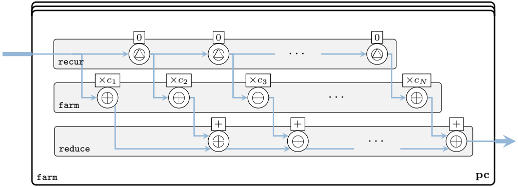
$$N_{\text{Farm}} = N_{\text{Coefficients}}$$

Refining PC into a FIR Process Network





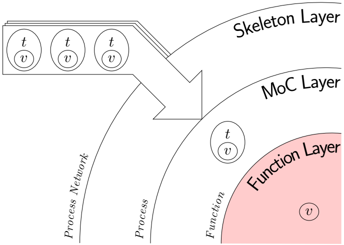




Floating Point to Fixed Point Number Representation

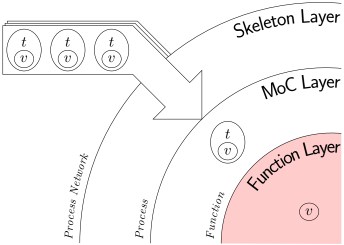
Specification

"After the DBF stage the numbers are represented with sufficient accuracy using 20 bits."




```
pc :: Vector (SY.Signal (Complex Float)) -> Vector (SY.Signal (Complex Float))
```

Floating Point to Fixed Point Number Representation

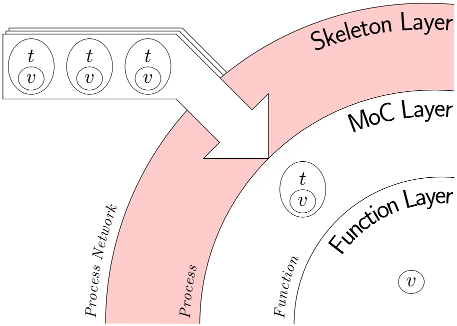


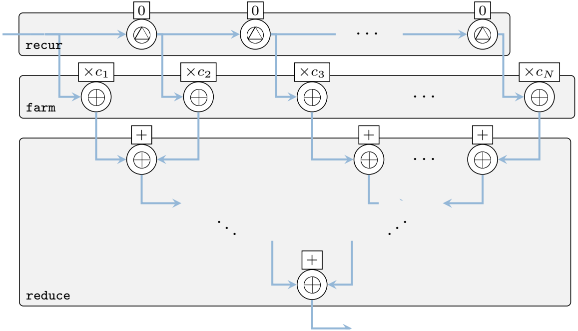
Specification

"After the DBF stage the numbers are represented with sufficient accuracy using 20 bits."

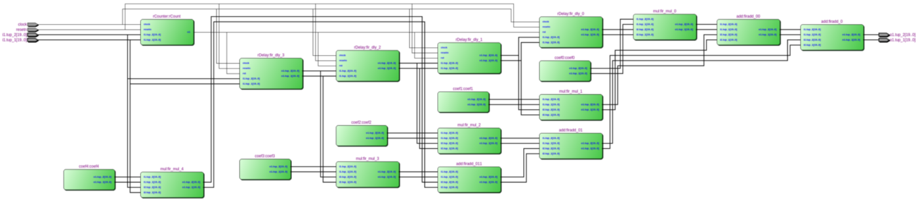
```
pc :: Vector (SY.Signal (Complex Float)) -> Vector (SY.Signal (Complex Float))

pc :: Vector (SY.Signal (Complex Fixed20)) -> Vector (SY.Signal (Complex Fixed20))
```

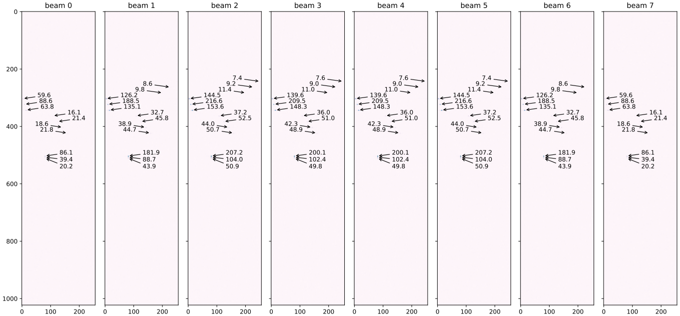
Load Balancing the FIR Reduction



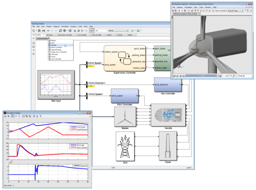


RTL Synthesis



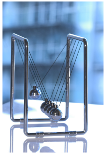


Why Have We Done This?



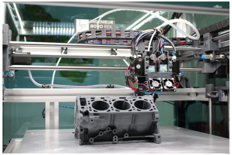
...to understand systems ✓

$$v(t) = v(0) + \frac{1}{m}\int_0^t F(\tau)d\tau$$



...to design systems ✓

...to create systems ✓



Conclusions & Future work

A Framework for Layered Languages

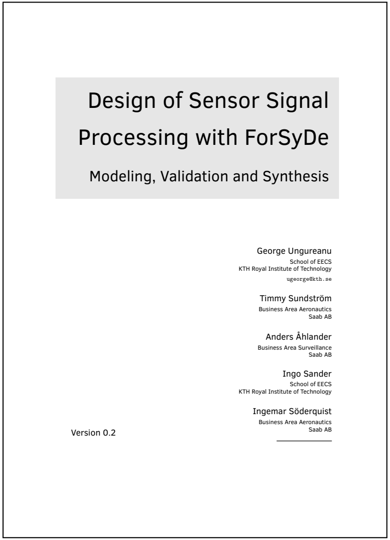
- orthogonalize aspects of CPS modeling
- isolates design problems

Demonstration on the AESA System

- using ForSyDe ecosystem (`https://forsyde.github.io`)
- open-source (`https://github.com/forsyde/aesa-radar`)
- technical report

Future Work

- explore other modeling paradigms (e.g. contracts)
- HW/SW transformation-based synthesis flow


Design of Sensor Signal Processing with ForSyDe

Modeling, Validation and Synthesis

George Ungureanu
School of EECS
KTH Royal Institute of Technology
ugeorge@kth.se

Timmy Sundström
Business Area Aeronautics
Saab AB

Anders Åhlander
Business Area Surveillance
Saab AB

Ingo Sander
School of EECS
KTH Royal Institute of Technology

Ingemar Söderquist
Business Area Aeronautics
Saab AB

Version 0.2

Thank you!