MathLib – a MATLAB equivalent Library for SV/UVM

Anirudh Pradyumnan S
(King's College London)

Deepa Palaniappan
(AsFigo Technologies, Zurich)

Nambi J U
(Lyle, IN)

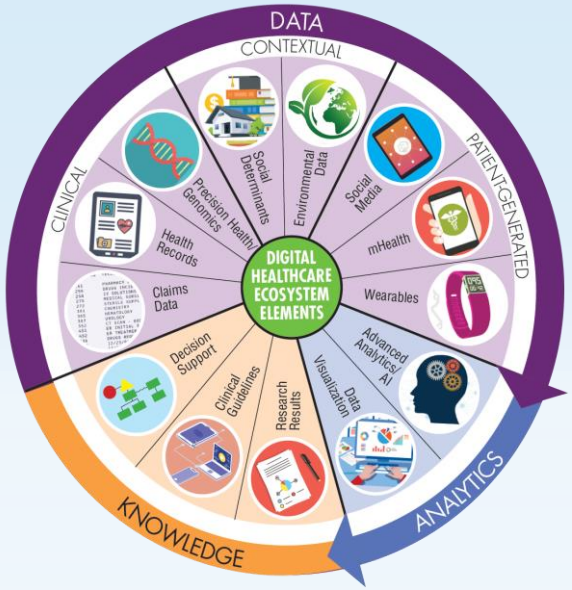
Srinivasan Venkataramanan
(VerifWorks, IN)

Shankar Hemmady
(Blue Horizons, USA)

AsFigo
Chip Design as it should be

2

Motivation



- System design involves higher level mathematical models, algorithms
  - MATLAB models as reference
  - Intention is to leverage such models
  - Cost of various tools, licenses, ease of integration
- UVM
  - Most widely used in verification
- Ideal is to have reference model in native SV/UVM
  - Saves cost of extra tools
  - No extra “plumbing” needed
  - Debug is native
- Standard IEEE 1800 lacks one-to-one mapping of MATLAB functions
  - Some close-matches do exist, need wrappers
  - MathLib – a solution, used across few designs

3 Modern day System Designs (AMS, Biomed)

- Creative Confluence of Analog, Digital, Statistical, predictive AI etc.
- Code generation – uses Generative AI techniques
  - MatGPT (Matlab)
  - SystemVerilog, UVM
- Heavy leverage of reference models from System Design to RTL verification
- Budget constraints on tools, licenses
- Debug productivity

MatGPT powered by Open AI™ ChatGPT API

4 MathLib in SystemVerilog - implementation

- SystemVerilog package
  - Perfect namespace for common functions
  - Usable in various contexts:
    - module
    - class
    - Interface
- SystemVerilog class
  - Full featured Object-oriented context
  - Suitable for class-based environments
- Model MATLAB equivalent functions in native SystemVerilog
- Verify against reference results
- Open-source library on top of GO2UVM library

```
rounding function
case positive:
  if fraction >= 0.5 ---> round return the "integer part" + 1 (for example 4.5 --->
  if fraction < 0.5 ---> round return the "integer part" (for example 4.2 --->4)

Case negative:
  if fraction >= 0.5 ---> round return the "integer part" -1 (for example -4.5 --->
  if fraction < 0.5 ---> round return the "integer part" (for example -4.2
```

```
30  function int g2u_ams_round(input real rval);
31    int ret_val;
32    ret_val = int'(rval);
33    return ret_val;
34  endfunction : g2u_ams_round
35
36  function int g2u_ams_truncate(input real rval);
37    int ret_val;
38    ret_val = $rtoi(rval);
39    return ret_val;
40  endfunction : g2u_ams_truncate
41
42  /* MATLAB equivalent sign function
43   Y = sign(x) returns
44   1 if x is greater than 0.
45   0 if x is  == 0
46    -1 if x < 0
47     */
48
```

5 MathLib in SystemVerilog – implementation (2/2)

- Built-in SV system functions handle scalars
- MATLAB models:
  - Scalar
  - Vector
  - Matrix
  - MDA
- SV – no “function overloading”
- Parameterized class + static functions

```
class MathLib_c #(
    type T = ml_vec_t
);
  static function void mean(T in_val);
    real rval;
    int  num_vals;

    rval = 0.0;
    rval = in_val.sum;
    num_vals = in_val.size;
    rval = rval / num_vals;
```

MathLib – typical functions

Modulo Division and Rounding	
mod	Remainder after division (modulo operation)
rem	Remainder after division
idivide	Integer division with rounding option
ceil	Round toward positive infinity
fix	Round toward zero
floor	Round toward negative infinity
round	Round to nearest decimal or integer

Arihtmetic operators	
sum	Sum of array elements
cumsum	Cumulative sum
movsum	Moving sum
diff	Differences and approximate derivative
prod	Product of array elements
cumprod	Cumulative product
rdivide (./)	Right array division

- Analytics – mean, std, median, mode
- Plots – we prefer to do outside SV/UVM
- Statistical functions – beyond the current scope

7 Heart Rate monitoring– biomedical application

- Checks either heart rate or pulse rate
- 3 main things/issues to look out for
  - Bradycardia (Heart rate below 60 per minute)
  - Tachycardia (Heart rate above 100 per minute)
  - Arrythmia (Irregular heart rhythm)
- Wearable sensors (Wristbands, smartwatches, chest strap sensors):
  - Good
  - Not as accurate as medical devices

8 MATLAB implementation - Heart-rate monitoring system

$$me = \frac{1}{N}\sum_{1}^{N} S_{novel}(t) - S_{ref}(t)$$

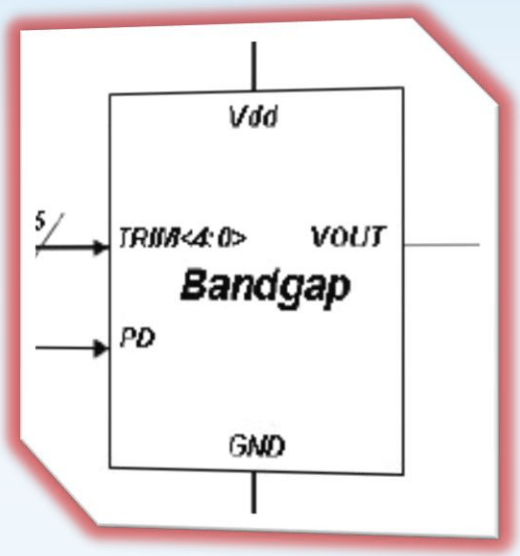
$$mae = \frac{1}{N}\sum_{1}^{N} \|S_{novel}(t) - S_{ref}(t)\|$$

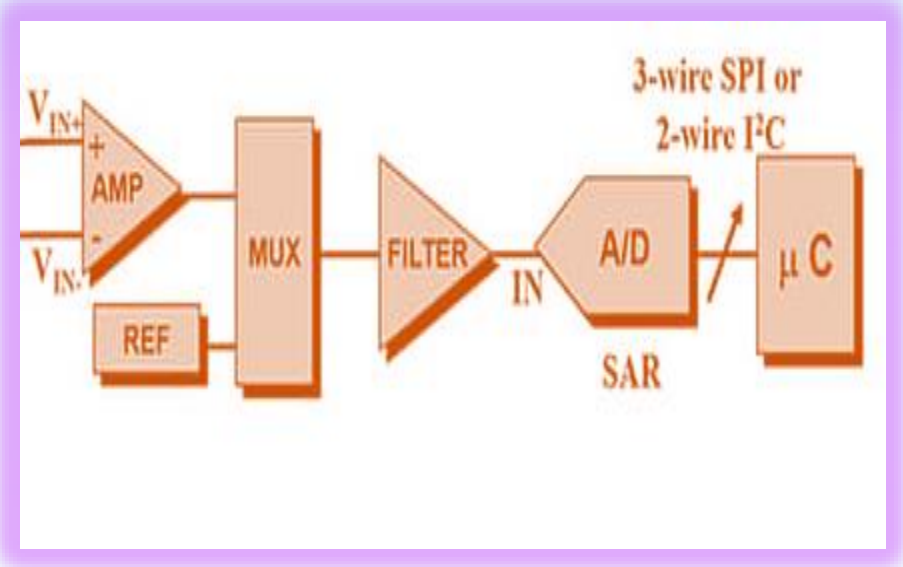
```
%% Mean Error
me_novel = mean(novel_readings - reference_readings);

%% Mean Absolute Error
mae_novel = mean(abs(novel_readings - reference_readings));

%% Percentage of outliers
outliers = isoutlier(novel_readings, "mean");
num_of_outliers = length(find(outliers == 1));
perc_of_outliers = num_of_outliers/length(novel_readings);
```

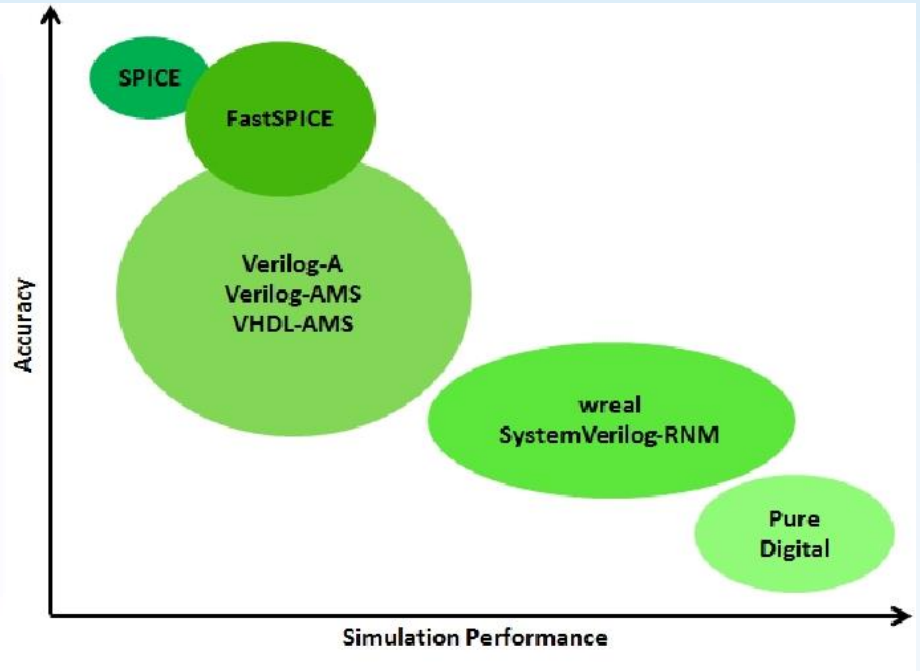
- Statistical indicators of the novel/new system when compared to an existing reference sensor/device
  - Mean error
  - Mean absolute error
  - Percentage of outliers (anomalous readings) (If present)
- Sample MATLAB code to implement above indicators

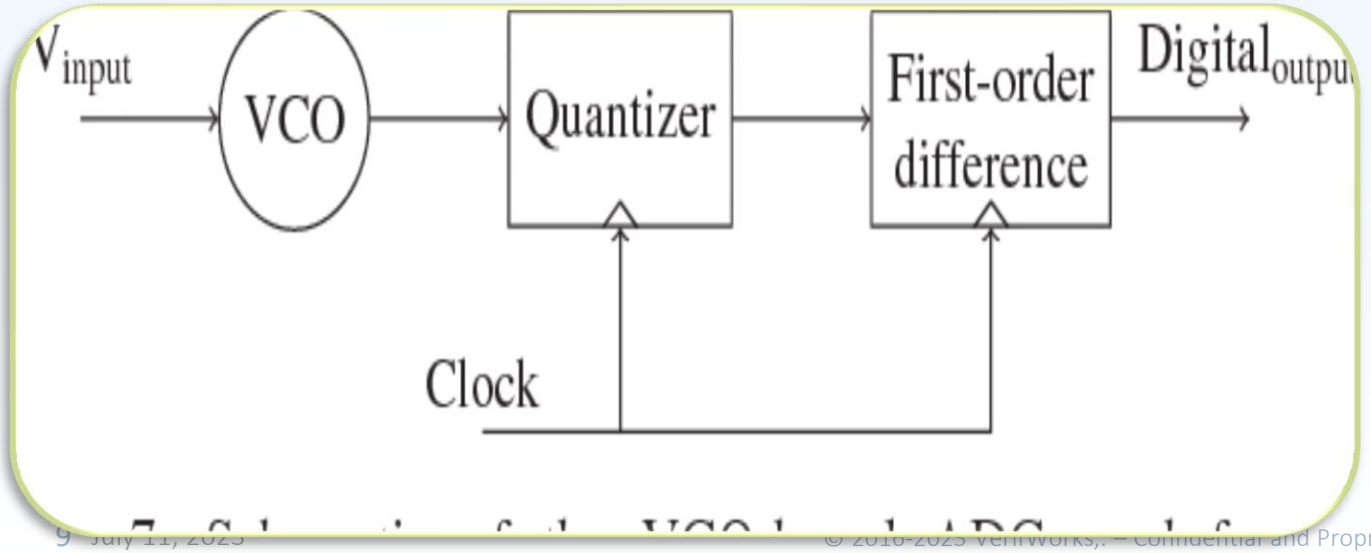
9

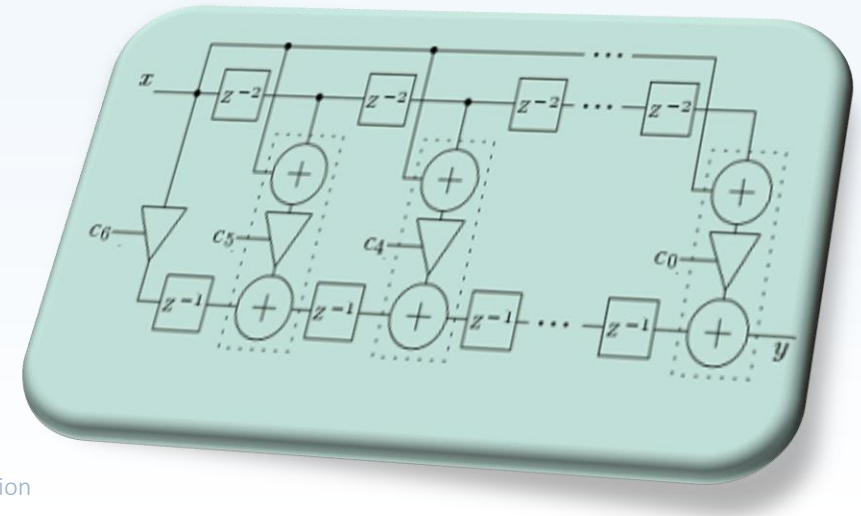
MathLib – application in AMS designs











MathLib code snippets

VCO Matlab code

```
%=========
%=========

EN=0.5+0.5*(sign(EN-0.5));
vout=(PAR_LDO_VMIN+PAR_LDO_STEP*RDIV
tsto=vout;
```

MathLib VEC example

```
26      end
27      ida_1d = '{1,-2.1,3,4.2};
28      rda_1d = '{1,-2.1,3,4.2};
29
30      m_rda_1d = MathLibVec #(int_darr_1d_t)::mean(ida_1d);
31      `ml_printf (("ida_1d: \n%p \nmean: \n%p",
32        ida_1d, m_rda_1d))
33      m_rda_1d = MathLibVec #(real_darr_1d_t)::mean(rda_1d);
34      `ml_printf (("REAL: \n%p \nmean: \n%p",
          rda_1d, m_rda_1d))
36
37      ida_2d = new [4]; // [3];
38      foreach (ida_2d[ii]) begin
39        ida_2d[ii]= new [3];
40      end
41      ida_2d = '{'{1,1,1}, '{2,2,2}, '{3,3,3}, '{4,4,4}};
42      // 0 1 1; 2 3 2; 1 3 2; 4 2 2]
43      ida_2d = '{'{0,1,1}, '{2,3,2}, '{1,3,2}, '{4,2,2}};
44      m_res_mean_of_2d = MathLibMat #(int_darr_2d_t)::mean(ida_2d);
45      `ml_printf (("MAT COL Mean: \n%p \nmean: \n%p",
46        ida_2d, m_res_mean_of_2d))
                                                  ,0,1}, '{1,2,3}};
                                                                27,1
```

MathLib types

```
15
16      typedef real ml_vec_t[];
17      typedef real ml_matrix_t[][];
18      typedef real ml_mda_t[][][];
19      typedef enum bit [1:0] {
20        ML_MEAN_ROW, ML_MEAN_COL, ML_MEAN_ALL} ml_mean_dim_t;
21
22      typedef enum bit [3:0] {
23        ML_TIE_NONE, ML_TIE_EVEN, ML_TIE_ODD, ML_TIE_PLUSINF,
24        ML_TIE_MINUSINF, ML_TIE_FROMZERO, ML_TIE_TOZERO
25      } ml_round_tie_t;
26
```

MathLib development – tools, framework

- Compatible with free tools such as Modelsim Intel FPGA Edition
- Runs on Icarus Verilog
  - Open-source
  - Limited class support in SV
  - No UVM
- Verilator
  - Open-source
  - Better SV, SVA
  - Unit Tests for each MathLib functions

Alternate implementations

- C/DPI based integration
  - Non-native
  - Harder to debug
  - Not easy to port across OS (Win/Linux)
- Simulink
  - Expensive $$

13

Summary

- Open-source library on top of UVM
- Models many commonly used MATLAB functions
  - More getting added
  - Open to contributions
- Used in various real-life SoC verification projects
  - Biomedical chips
  - AMS IPs – LPF, VCO, LDO etc.
- Saves 1000s of $$ in license costs
- Native code, easier to debug