2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION

UNITED STATES

SAN JOSE, CA, USA
MARCH 4-7, 2024

SV_LUT: A SystemVerilog Look Up Table package for developing complex AMS Real Number Modeling

FNU Farshad — Ulkasemi Inc

Shafaitul Islam Surush — Ulkasemi Inc

Simul Barua — Ulkasemi Inc

accellera SYSTEMS INITIATIVE

Agenda

- Need for Look Up Table (LUT) in SV-RNM modeling
- Challenges associated with LUT based AMS modeling and Our solution
- LUT Structure
- Overview of SV_LUT_PKG
  - Methods for defining and Population of LUT
  - Method for fetching data from LUT
- Application of LUT package in AMS modeling
  - RNM modeling demonstration of simple PTAT core
- Advanced use cases of SV_LUT_PKG
- Conclusion and Future Works

Need for Look Up Table (LUT) in SV-RNM modeling

- SV-RNM modeling of analog system is getting popular for Mixed Signal Verification
- Modeling process of Complex behavior of Analog Systems like Solar Cell, Ptat core, etc. involves,
  - Utilization of Look Up Tables (LUT)
  - Data points are exported from waveform dump on external simulators,
  - Most used file format for dumping and exporting data: Comma Separated Value (CSV)
- VerilogAMS supports .tbl format for LUT using $table_model()
- SystemVerilog do not have any built-in support for LUT

Challenges associated with LUT based AMS modeling

- Mechanism for addressing unknown data points between two known data points
- Real number-based index value matching mechanism for fetching data
- Performance of LUT should have minimum impact on AMS model performance
- Integration process of LUT in SV-RNM flow should be simple and require minimum code

Our Solution

- A SystemVerilog package named "sv_lut_pkg" which,
  - Defines a data type for representing the LUT
  - Parse the CSV file and map the data points into the LUT.
  - Search the populated LUT and fetch value based on index and data column definition and index column value
  - Apply reasonable tolerance while fetching values based on real number indexes
  - Support interpolation (Linear and cubic spline) and extrapolation
- Simple to integrate in traditional SV-RNM flow with minimum lines of codes

Overview of SV_LUT_PKG

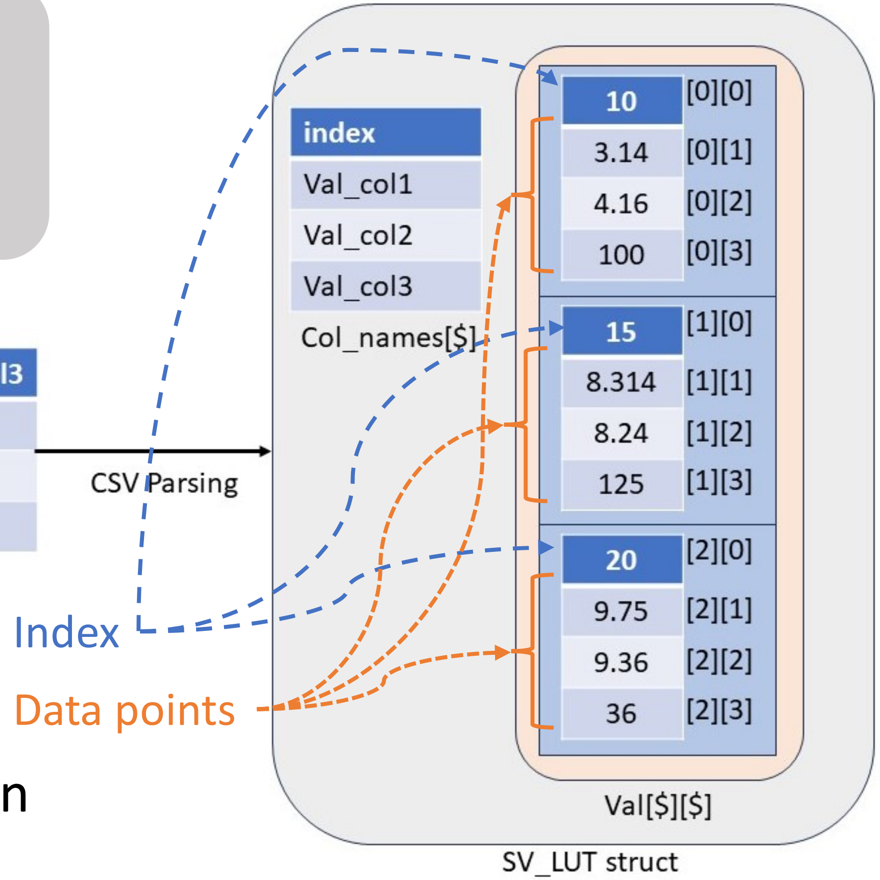
LUT Structure

- LUT is defined as a SV struct datatype that contains,
  - Two SV dynamic array/queue for,
    - Storing CSV column headers i.e. col_name[$]
    - Storing CSV Column data i.e. val[$][$]

```
typedef struct {
    string col_name[$];
    real val[$][$];
 } lut_struct_t;
```

	index	Val_col1	Val_col2	Val_col3
1	10	3.14	4.16	100
2	15	8.314	8.24	125
3	20	9.75	9.36	36

CSV File



- For every row x and column y,
  - val[x][0] represents index column value of CSV file
  - val[x][y], where y>0 holds the data points of the CSV column

Overview of SV_LUT_PKG

- Defined as SV struct data type containing two dynamic arrays/queue members
- Pre-processor macros are used for defining, populating and fetching value from LUT.
  - `POP_LUT() macro defines and populates the LUT
  - `SV_LUT() macro fetches value from LUT
- These pre-processor macros implement all the logic and functions required to define, populate and fetch value from LUT

Methods for Defining and Population of LUT

testbench.sv | sv_lut_pkg.sv

```
package sv_lut_pkg;
  typedef struct {
      string col_name[$];
      real val[$][$];
  } lut_struct_t;

  // create lut and populate it from csv file
  // parameters: lut_name = name of lut variable
  //                  csv_file = full path to csv file
  `define POP_LUT(lut_name, csv_file) \
    lut_struct_t lut_name; \
    int ``lut_name``_x_indx_col_num = 0; \
    real ``lut_name``_x_indx_val_tol = 0.001; \
    initial begin \
      assert(read_csv(`"csv_file`", lut_name)); \
    end
```

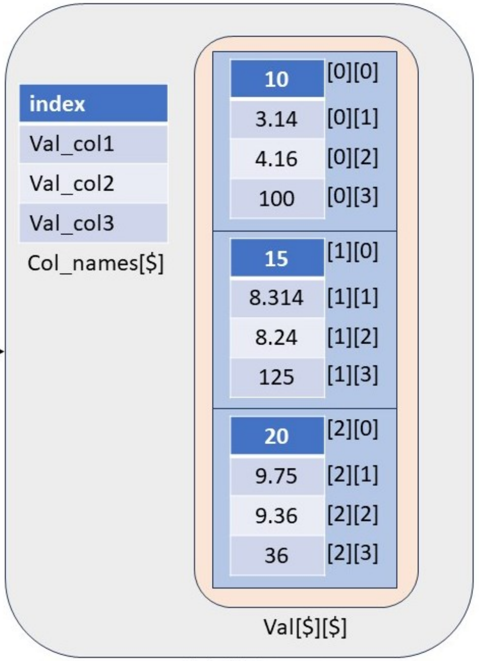
Name of the LUT

Path to CSV File

	index	Val_col1	Val_col2	Val_col3
1	10	3.14	4.16	100
2	15	8.314	8.24	125
3	20	9.75	9.36	36

CSV File

CSV Parsing



SV_LUT struct

Method for fetching data from LUT

testbench.sv | sv_lut_pkg.sv

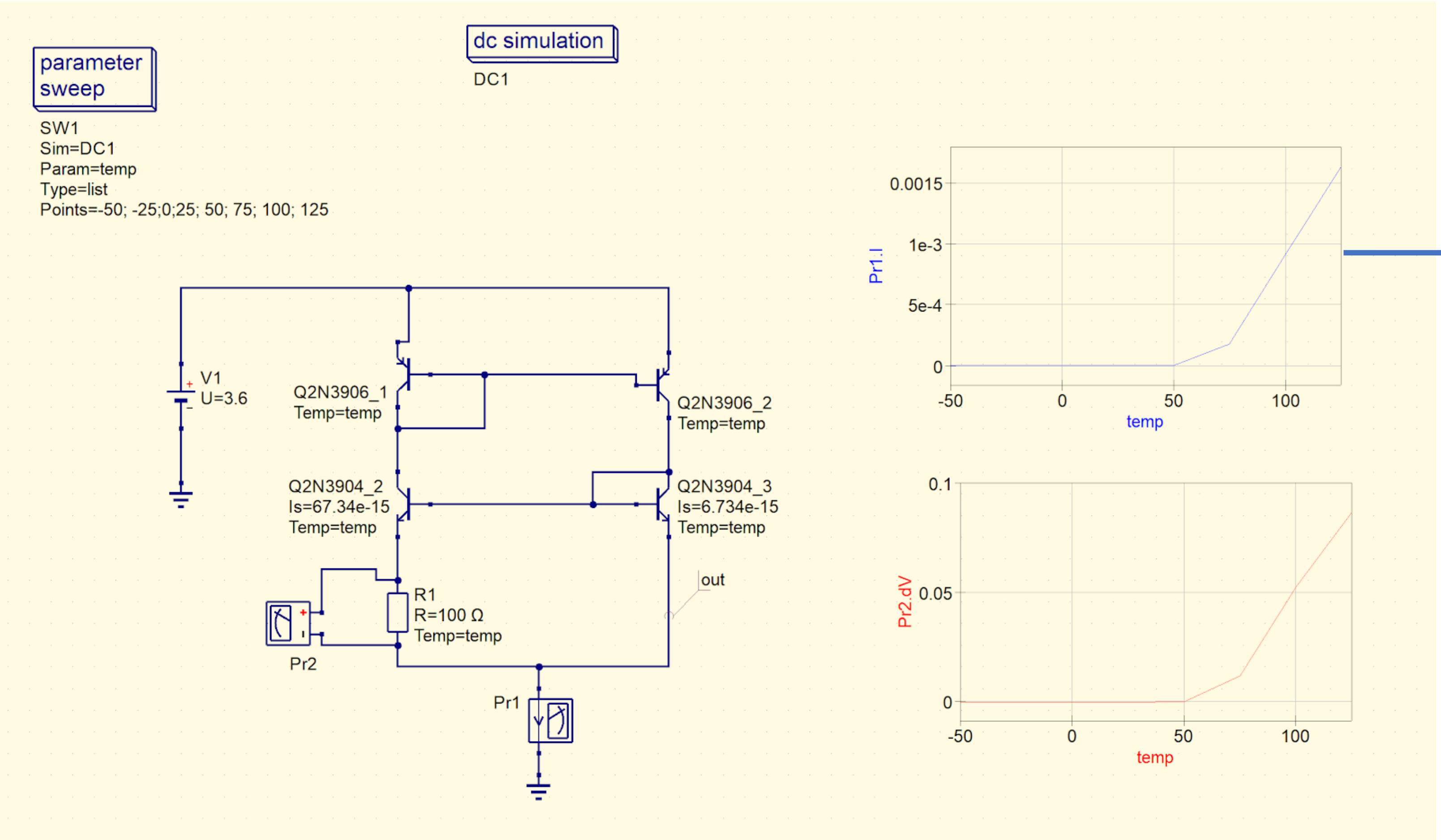
```
  // main lut function
  // parameters:
  //     x_indx: (string) lookup index of look up table
  //     lut_name: ()
  //     y_indx: (string) return index selector for LUT
  //     interpolation_type: (string) L: linear interpolatrion, C: cuble spline interpolation
  //         x_indx_col_num: (int) column number for x_index
  `define SV_LUT(x_indx_val,  lut_name, y_indx, interpolation_type) \
lut_fetch_val(x_indx_val, lut_name, y_indx, interpolation_type, ``lut_name``_x_indx_col_num,
``lut_name``_x_indx_val_tol);

  function real lut_fetch_val(real x_indx, lut_struct_t lut, int y_indx_col, string intrpol_type,
int x_indx_col_num=0, real x_indx_val_reltol=0.01);
```

Integration of SV_LUT_PKG in SV-RNM flow

Application of LUT package in AMS modeling

- Demonstrated with SV-RNM modeling flow for Simple PTAT core design
  - Opensource qucsstudio is used for designing the schematic
  - Spice simulation waveform is dumped as CSV file
- SV-RNM model of ptat core is developed with sv_lut_pkg to demonstrate the modeling flow and capabilities

Schematic View of the PTAT core and CSV



	A	B
1	temperature (°C)	r Pr1.I (A)
2	-50	4.85E-13
3	-25	1.10E-11
4	0	2.10E-10
5	25	4.42E-09
6	50	5.85E-07
7	75	0.000176578
8	100	0.000911202
9	125	0.00163012
10		

CSV dump

RNM modeling flow with sv_lut_pkg

- LUT define and population
  - lut_name: ptat_lut
  - csv_file_path: ptat_sim.csv

- Value fetching from LUT
  - x_indx_val: temp
  - lut_name: ptat_lut
  - y_indx: 1 i.e. col 1 (rPr1.I)
  - Interpolation_type: “1L” i.e. linear

design.sv | ptat_sim.csv | EE_pkg.sv | multi_lut_design.sv | sample_csv.csv

```
// RNM model of ptat core
`timescale 1s/1ps
`include "sv_lut_pkg.sv"

module ptat(
            input real vdd,
            input real gnd,
            output real ptat_out
);
  import sv_lut_pkg:: *;

//    real vdd;
//    real gnd;
//    real ptat_out;

  real temp;   // temperature in degree c

  // create lut and populate data from csv file
  `POP_LUT(ptat_lut, ptat_sim.csv)

  always @(temp) begin
    // `SV_LUT("input_index_name"/Int(input_index_number), "csv_file_name",
"output_index_name"/Int(output_index_number), "interpolation type = linear")
    ptat_out = `SV_LUT(temp, ptat_lut, 1, "1L");
  end

endmodule
```

SV testbench

- Design is simulated from -50°C to 125°C
- Incremental Step: 5°C
- Simulated using Cadence Xcelium Suite

testbench.sv | sv_lut_pkg.sv

```
module tb;

  import sv_lut_pkg::*;

  real vdd, gnd, out;

  ptat DUT(
    .vdd(vdd),
    .gnd(gnd),
    .ptat_out(out)
  );

  initial begin
    for(int temp = -50; temp<150; temp=temp+5) begin
      $display("Setting temp: %0d", temp);
        DUT.temp = temp;
        # 100us;
      $display("V(out):%0e", out);
    end
  end

  initial begin
    vdd = 3;
    gnd = 0;
  end

  initial begin
    $dumpvars(0, DUT);
    $dumpfile("ptat_sim.vcd");
  end
endmodule
```

Simulation Results

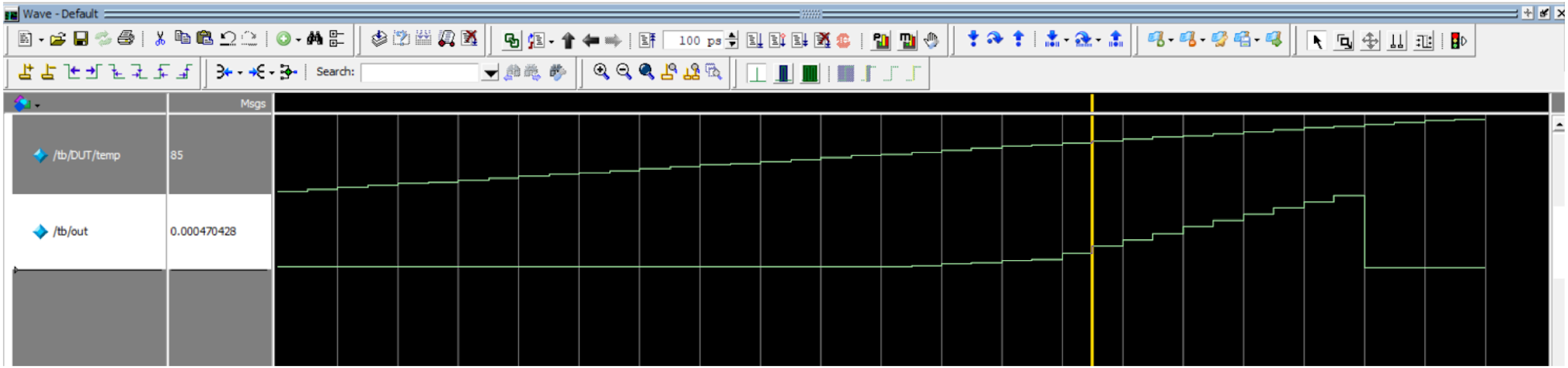
- Data points in CSV File
  - -50°C,
  - -25°C,
  - 0°C,
  - 25°C,
  - 50°C,
  - 75°C,
  - 100°C,
  - 125°C
- Rest of the data points are linearly interpolated

```
# V(out):4.850000e-13
# Setting temp: -45
# V(out):2.588000e-12
# Setting temp: -40
# V(out):4.691000e-12
# Setting temp: -35
# V(out):6.794000e-12
# Setting temp: -30
# V(out):8.897000e-12
# Setting temp: -25
# V(out):1.100000e-11
# Setting temp: -20
# V(out):5.080000e-11
# Setting temp: -15
# V(out):9.060000e-11
# Setting temp: -10
# V(out):1.304000e-10
# Setting temp: -5
# V(out):1.702000e-10
# Setting temp: 0
# V(out):2.100000e-10
# Setting temp: 5
# V(out):1.052000e-09
```

```
# Setting temp: 10
# V(out):1.894000e-09
# Setting temp: 15
# V(out):2.736000e-09
# Setting temp: 20
# V(out):3.578000e-09
# Setting temp: 25
# V(out):4.420000e-09
# Setting temp: 30
# V(out):1.205360e-07
# Setting temp: 35
# V(out):2.366520e-07
# Setting temp: 40
# V(out):3.527680e-07
# Setting temp: 45
# V(out):4.688840e-07
# Setting temp: 50
# V(out):5.850000e-07
# Setting temp: 55
# V(out):3.578360e-05
# Setting temp: 60
# V(out):7.098220e-05
```

```
# Setting temp: 65
# V(out):1.061808e-04
# Setting temp: 70
# V(out):1.413794e-04
# Setting temp: 75
# V(out):1.765780e-04
# Setting temp: 80
# V(out):3.235028e-04
# Setting temp: 85
# V(out):4.704276e-04
# Setting temp: 90
# V(out):6.173524e-04
# Setting temp: 95
# V(out):7.642772e-04
# Setting temp: 100
# V(out):9.112020e-04
# Setting temp: 105
# V(out):1.054986e-03
# Setting temp: 110
# V(out):1.198769e-03
# Setting temp: 115
# V(out):1.342553e-03
# Setting temp: 120
# V(out):1.486336e-03
```

Simulation Waveform Results



Advanced Use cases of SV_LUT_PKG

SYSTEMS INITIATIVE

Advanced Use cases of SV_LUT_PKG

- Use of Multiple LUTs in a single model
  - Lut_name parameter should be different in POP_LUT

design.sv | ptat_sim.csv | EE_pkg.sv | multi_lut_design.sv | sample_csv.csv

```
`timescale 1s/1ps
`include "sv_lut_pkg.sv"

module multi_lut(
                    input real vdd,
                    input real gnd,
                    output real out_V_1,
                    output real out_V_2,
                    output real out_V_3
                );
  import sv_lut_pkg:: *;
  real temp;

    // create lut and populate data from csv file
  `POP_LUT(ptat_lut1, ptat_sim.csv)
  `POP_LUT(ptat_lut2, ptat_sim2.csv)
  `POP_LUT(ptat_lut3, ptat_sim3.csv)

  always @(temp) begin
    // `SV_LUT("input_index_name"/Int(input_index_number), "csv_file_name", "output_index_name"/Int(output_index_number), "interpolation type = linear")
    out_V_1 = `SV_LUT(temp, ptat_lut1, 1, "1L");
  end
  always @(temp) begin
    // `SV_LUT("input_index_name"/Int(input_index_number), "csv_file_name", "output_index_name"/Int(output_index_number), "interpolation type = linear")
    out_V_2 = `SV_LUT(temp, ptat_lut2, 1, "1L");
  end
  always @(temp) begin
    // `SV_LUT("input_index_name"/Int(input_index_number), "csv_file_name", "output_index_name"/Int(output_index_number), "interpolation type = linear")
    out_V_3 = `SV_LUT(temp, ptat_lut3, 1, "1L");
  end
endmodule
```

Different lut_name parameter

Advanced Use cases of SV_LUT_PKG Cont..

- Different y_indx value to fetch data from a single CSV against a single index column
  - In the y_indx value in the SV_LUT() macro should be unique

design.sv | ptat_sim.csv | EE_pkg.sv | multi_lut_design.sv | sample_csv.csv | single_lut_muti_op

```
`timescale 1s/1ps
`include "sv_lut_pkg.sv"

module single_lut_multi_op(
                          input real vdd,
                          input real gnd,
                          output real out_V_1,
                          output real out_V_2,
                          output real out_V_3
                         );
  import sv_lut_pkg:: *;
  real temp;

    // create lut and populate data from csv file
   `POP_LUT(ptat_lut, ptat_sim.csv)

  always @(temp) begin
    // `SV_LUT("input_index_name"/Int(input_index_number), "csv_file_name", "output_index_name"/Int(output_index_number), "interpolation type = linear")
    out_V_1 = `SV_LUT(temp, ptat_lut, 1, "1L");
    out_V_2 = `SV_LUT(temp, ptat_lut, 2, "1L");
    out_V_3 = `SV_LUT(temp, ptat_lut, 3, "1L");
  end

endmodule
```

Different y_indx value

Conclusion and Future Works

SYSTEMS INITIATIVE

Conclusion and Future Works

- Conclusion
  - Modeling is effortless
  - Most of the LUT design complexity is transparent to verification engineers
  - Easy to integrate with traditional RNM modeling flow
- Limitations
  - Maps single index to single data column
  - Mapping of multiple inputs for fetching corresponding output not supported
  - No data validation in CSV parser
  - Slow data processing for huge CSV files

Any Questions

Thank You

accellera
SYSTEMS INITIATIVE