Verilog-AMS

Verilog-AMS is a derivative of the Verilog hardware description language that includes analog and mixed-signal extensions (AMS) in order to define the behavior of analog and mixed-signal systems. It extends the event-based simulator loops of Verilog/SystemVerilog/VHDL, by a continuous-time simulator, which solves the differential equations in analog-domain. Both domains are coupled: analog events can trigger digital actions and vice versa.[1]

Contents

Overview

The Verilog-AMS standard was created with the intent of enabling designers of analog and mixed signal systems and integrated circuits to create and use modules that encapsulate high-level behavioral descriptions as well as structural descriptions of systems and components.[2][3][4]

Verilog-AMS is an industry standard modeling language for mixed signal circuits. It provides both continuous-time and event-driven modeling semantics, and so is suitable for analog, digital, and mixed analog/digital circuits. It is particularly well suited for verification of very complex analog, mixed-signal and RF integrated circuits.[5]

Verilog and Verilog/AMS are not procedural programming languages, but event-based hardware description languages (HDLs). As such, they provide sophisticated and powerful language features for definition and synchronization of parallel actions and events. On the other hand, many actions defined in HDL program statements can run in parallel (somewhat similar to threads and tasklets in procedural languages, but much more fine-grained). However, Verilog/AMS can be coupled with procedural languages like the ANSI C language using the Verilog Procedural Interface of the simulator, which eases testsuite implementation, and allows interaction with legacy code or testbench equipment.

The original intention of the Verilog-AMS committee was a single language for both analog and digital design, however due to delays in the merger process it remains at Accellera while Verilog evolved into SystemVerilog and went to the IEEE.

Code example

Verilog/AMS is a superset of the Verilog digital HDL, so all statements in digital domain work as in Verilog (see there for examples). All analog parts work as in Verilog-A.

The following code example in Verilog-AMS shows a DAC which is an example for analog processing which is triggered by a digital signal:

```
`include "constants.vams"
`include "disciplines.vams"
// Simple DAC model
module dac_simple(aout, clk, din, vref);

    // Parameters
    parameter integer bits = 4 from [1:24];
    parameter integer td = 1n from[0:inf);  // Processing delay of the DAC

    // Define input/output
    input clk, vref;
    input [bits-1:0] din;
    output aout;

    //Define port types
    logic clk;
    logic [bits-1:0] din;
    electrical  aout, vref;

    // Internal variables
    real aout_new, ref;
    integer i;

    // Change signal in the analog part
    analog begin
        @(posedge clk) begin // Change output only for rising clock edge

            aout_new = 0;
            ref = V(vref);

            for(i=0; i<bits; i=i+1) begin
                ref = ref/2;
                aout_new = aout_new + ref * din[i];
            end
        end
        V(aout) <+ transition(aout_new, td, 5n); // Get a smoother transition when output level
changes
    end
endmodule
```

The ADC model is reading analog signals in the digital blocks:

```
`include "constants.vams"
`include "disciplines.vams"
// Simple ADC model
module adc_simple(clk, dout, vref, vin);

    // Parameters
    parameter integer bits = 4 from[1:24]; // Number of bits
    parameter integer td = 1 from[0:inf);  // Processing delay of the ADC

    // Define input/output
    input clk, vin, vref;
    output [bits-1:0] dout;

    //Define port types
    electrical vref, vin;
    logic clk;
    reg [bits-1:0] dout;

    // Internal variables
    real ref, sample;
    integer i;


    initial begin
        dout = 0;
    end

    // Perform sampling in the digital blocks for rising clock edge
    always @(posedge clk) begin

        sample = V(vin);
        ref = V(vref);

        for(i=0; i<bits; i=i+1) begin

            ref = ref/2;

            if(sample > ref) begin
                dout[i] <= #(td) 1;
                sample = sample - ref;
            end
            else
                dout[i] <= #(td) 0;
        end
    end
endmodule
```

See also

- Verilog
- Verilog-A
- VHDL-AMS
- SystemC-AMS (http://www.systemc-ams.org/)

References

1. Scheduling semantics are specified in the Verilog/AMS Language Reference Manual, section 8.
2. Accellera Verilog Analog Mixed-Signal Group, "Overview," http://www.verilog.org/verilog-ams/htmlpages/overview.html
3. Verilog-AMS Language Reference Manual (https://www.accellera.org/images/downloads/standards/v-ams/VAMS-LRM-2-4.pdf)
4. The Designer's Guide to Verilog-AMS (http://www.designers-guide.org/Books/dg-vams/index.html)
5. Verification of Complex Analog Integrated Circuits (http://www.designers-guide.com/docs/cicc06-dgc.pdf) Archived (https://web.archive.org/web/20061018055626/http://www.designers-guide.com/docs/cicc06-dgc.pdf) October 18, 2006, at the Wayback Machine

External links

- I. Miller and T. Cassagnes, "Verilog-AMS Eases Mixed Mode Signal Simulation," *Technical Proceedings of the 2000 International Conference on Modeling and Simulation of Microsystems*, pp. 305–308, Available: https://web.archive.org/web/20070927051749/http://www.nsti.org/publ/MSM2000/T31.01.pdf

General

- Accellera Verilog Analog Mixed-Signal Group (http://www.accellera.org/activities/committees/verilog-ams/)
- verilogams.com (http://www.verilogams.com) — User's manual for Verilog-AMS and Verilog-A
- The Designer's Guide Community, Verilog-A/MS (http://www.designers-guide.org/VerilogAMS/) — Examples of models written in Verilog-AMS

- EDA.ORG AMS Wiki (http://www.eda.org/twiki/bin/view.cgi/VerilogAMS) - Issues, future development, SystemVerilog integration

Open Source Implementations

- OpenVAMS, an Open-Source VerilogAMS-1.3 Parser with internal VPI-like representation (http://ovams.sourceforge.net/)
- V2000 project - Verilog-AMS parser & elaborator (http://www.v-ms.com/)