System Simulation Using VHDL-AMS: Modeling Multiple Physical Domains for HEV Applications

Xiao Hu
Scott Stanton
Leon Voss
ANSYS Inc.
Introduction

• Ansys System Simulation
• Component Modelling:
  – Battery: Electrochemistry
  – Heat Exchanger: Fluid, Thermal
  – PM Machine: Electromechanical
• Towards Virtual Prototyping ...
Automotive and railway systems, electric drives, home appliances and other systems consist of a variety of components. Each component may influence the behavior of another component.
ANSYS Comprehensive Solution

ANSYS Mixed-Signal Multi-Domain System Simulator

Model Extraction & Cosimulation

ANSYS Workbench

Electrical  Magnetic  Fluid  Mechanical  Thermal  Acoustic
What is VHDL-AMS?

VHDL-AMS is a strict superset of IEEE Std. 1076

Very Large Scale Integrated Circuit Hardware Description Language – Analog and Mixed-Signal

Description & simulation of event-driven systems

IEEE 1076 VHDL 1993

IEEE 1076.1 VHDL-AMS 1999

Description & simulation of analog and mixed signal circuits and systems

VHDL-AMS is a strict superset of IEEE Std. 1076
Why Use VHDL-AMS

- **Standard Format Allows Model Portability**
  - Different engineering groups within same company
  - With Sub-Contractors
  - Between different simulators

- **Multi-level Modeling**
  - Different levels of abstraction of model behavior

- **Multi-domain Modeling**
  - Electrical, Thermal, Magnetic, Mechanical, etc.

- **Mixed-signal Modeling**
  - Supports analog and digital modeling
VHDL-AMS Model Construct

- **Entity**
  - Interface description of a subsystem or physical device
  - Specifies input and output ports to the model

- **Architecture**
  - Behavior description
  - Can be dataflow, structural, procedural, etc
  - Modeling can deal with both analog (continuous) and digital (discrete) domains
VHDL-AMS Code for a Resistor

- **Entity**
  - Interface description of a subsystem or physical device
  - Specifies input and output ports to the model

```vhdl
LIBRARY Ieee;
use Ieee.electrical_systems.ALL;
ENTITY vhdl_resistor IS
  GENERIC (
    resistance : resistance := 1.0e3
  );
  PORT ( 
    QUANTITY temp : IN real := 300.0;
    TERMINAL p : electrical;
    TERMINAL n : electrical
  );
END ENTITY vhdl_resistor;
```

The resistor model has one model constant, one input quantity, and two terminals.
VHDL-AMS Code for a Resistor

• Architecture
  – Description of the model and no solving information is required

```
ARCHITECTURE arch_vhdl_resistor OF vhdl_resistor IS
  QUANTITY voltage ACROSS current THROUGH p TO n;
BEGIN
  voltage == current * resistance;
END ARCHITECTURE arch_vhdl_resistor;
```

No solving information is needed!!

```
ARCHITECTURE arch_vhdl_resistor OF vhdl_resistor IS
  QUANTITY voltage ACROSS current THROUGH p TO n;
BEGIN
  voltage == current * resistance * (1.0+1.0e-2*(temp-300.0)+1.1e-3*(temp-300.0)**2);
END ARCHITECTURE arch_vhdl_resistor;
```

A second architecture is possible!!
The capacitor entity has one model constant, one input, and two terminals

The model description essentially has two lines, one for initial condition and one for the governing equation!!
VHDL-AMS Model of a Permanent magnet Synchronous Generator

LIBRARY IEEE;
USE IEEE.ELECTRICAL_SYSTEMS.ALL;
USE IEEE.MECHANICAL_SYSTEMS.ALL;
USE IEEE.MATH_REAL.ALL;

ENTITY symp_mech_pin IS
  GENERIC(
    L1d  : INDUCTANCE := 0.042;
    L1q  : INDUCTANCE := 0.042;
    ke  : INDUCTANCE := 0.875;
    p    : REAL := 2.0;
    j    : MOMENT_INERTIA := 0.075;
    i1a0 : CURRENT := 0.0;
    i1b0 : CURRENT := 0.0;
    i1c0 : CURRENT := 0.0;
    phi0 : ANGLE := 0.0);
  PORT(
    TERMINAL a, b, c : ELECTRICAL;
    TERMINAL m : ROTATIONAL_V;
    QUANTITY r1 : IN RESISTANCE := 0.4;
    QUANTITY n : OUT REAL;
    QUANTITY phi : OUT ANGLE);
END ENTITY symp_mech_pin;
ARCHITECTURE behav OF symp_mech_pin IS

BEGIN

v1d == i1d * r1 + i1d'dot * l1d - omel * psi1q;
v1q == i1q * r1 + i1q'dot * l1q + omel * psi1d;

i1beta == bc2beta * (i1b - i1c);
i1d == i1a * cos(y) + i1beta * sin(y);
i1q == - i1a * sin(y) + i1beta * cos(y);

psi1d == i1d * l1d + ke;
psi1q == i1q * l1q;

mi == kmi * (psi1d * i1q - psi1q * i1d);

omg'dot == (1.0/j) * (mi + tld);
ang == omg'integ;

v1alpha == two_three * v1a - one_three * (v1b + v1c);

phi == y;
n == omg * om2n;
y == p * ang;
omel == p * omg;

END ARCHITECTURE behav;
PM synchronous machine without damper VHDL-AMS model
What About PDEs – Boundary Value Problems?

• Example:

\[
\begin{aligned}
\frac{d\Phi}{dx} &= x \\
\Phi(0) &= \phi_{bc}
\end{aligned}
\]

• Entity

ENTITY steady_state_boundary_value IS
generic (phibc: real := 0.0);
END ENTITY steady_state_boundary_value;

• Architecture

ARCHITECTURE arch_steady_state_boundary_value OF
steady_state_boundary_value IS

quantity phi1,phi2,phi3,phi4,phi5 : real;
constant h : real := 0.25;

BEGIN

phi1 == phibc;
(\(\text{phi3}-\text{phi1}\))/(2.0*h) == 1.0*h;
(\(\text{phi4}-\text{phi2}\))/(2.0*h) == 2.0*h;
(\(\text{phi5}-\text{phi3}\))/(2.0*h) == 3.0*h;
(1.0*phi3-4.0*phi4+3.0*phi5)/(2.0*h) == 4.0*h;

END ARCHITECTURE
arch_steady_state_boundary_value;
What About PDEs – Initial Value Problems?

• Example:

\[
\frac{\rho C_p \, \partial T}{k \, \partial t} = \frac{\partial^2 T}{\partial x^2}
\]

\[
T(x, 0) = 20000x \quad x \in [0, 0.005]
\]

\[
T(x, 0) = 20000(0.01 - x) \quad x \in [0, 0.01]
\]

\[
T(0.0, t) = 0.0
\]

\[
T(0.01, t) = 0.0
\]

• Entity

ENTITY transient_diffusion IS
  generic (rhop: real := 2000.0; k: real := 2.0; Cpp: real := 1000.0);
END ENTITY transient_diffusion;

• Architecture (main part)

IF (domain = quiescent_domain) USE
  T1 == 20.0;
  T2 == 60.0;
  T3 == 100.0;
  T4 == 60.0;
  T5 == 20.0;
ELSE
  rho*Cp*T1'dot == -(N1p - N0p)/h;
  rho*Cp*T2'dot == -(N2p - N1p)/h;
  rho*Cp*T3'dot == -(N3p - N2p)/h;
  rho*Cp*T4'dot == -(N4p - N3p)/h;
  rho*Cp*T5'dot == -(N5p - N4p)/h;
END USE
Newman’s 1D Electrochemistry Model in Simplorer

- Electrochemical Kinetics
- Solid-State Li Transport
- Electrolytic Li Transport
- Charge Conservation/Transport
- (Thermal) Energy Conservation

\[ j_{Li} = a_z i_o \left\{ \exp \left[ \frac{\alpha_z F}{RT} \eta \right] - \exp \left[ - \frac{\alpha_z F}{RT} \eta \right] \right\} \]

\[ \frac{\partial (\varepsilon_0 c_e)}{\partial t} = \nabla \cdot (D_e \nabla c_e) + \frac{1-t^+}{F} j_{Li} \]

\[ \eta = (\phi_s - \phi_L) - U \]

Results from Simplorer
Results from Newman
Governing Equations

The governing equations of porous electrode model of the lithium-ion battery (Electrochemical Systems, 3rd by John Newman)

\[
\varepsilon \frac{\partial c}{\partial t} = \nabla \cdot (\varepsilon D \nabla c) - \frac{i_2 \cdot \nabla t^0}{z_+ v_+ F} + \frac{aj_n(1-t^0)}{v_+}
\]

\[
i_2 = -\kappa \nabla \phi_2 + \frac{\kappa RT}{F} (1-t^0) \nabla \ln c
\]

\[
I - i_2 = -\sigma \nabla \phi_1
\]

\[
aFj_n = \nabla \cdot i_2
\]

\[
j_n = i_0 \left\{ \exp\left( \frac{\alpha_a F}{RT} \eta_s \right) - \exp\left( -\frac{\alpha_c F}{RT} \eta_s \right) \right\}
\]
LIBRARY leee;
use leee.thermal_systems.all;
use leee.fluidic_systems.all;
use leee.math_real.ALL;
use leee.electrical_systems.ALL;
ENTITY battery IS
  generic (  
    diffD : real := 7.5e-11;  
    epslohn: real := 0.357;  
    epslonp: real := 0.444;  
    epslonfn: real := 0.172;  
    epslonfp: real := 0.259;  
    sigman : real := 100.0;  
    sigmap : real := 3.8;  
    diffDsn: real := 3.9e-14;  
    diffDsp: real := 1.0e-13;  
    kn: real := 2.334e-11;  
    kp: real := 2.334e-11;  
    Bruggn: real :=1.5;  
    Bruggp: real :=1.5;  
    zp : real := 1.0;  
    mup: real := 1.0;  
    sp : real :=-1.0;  
    n : real := 1.0;  
  )
  PORT (  
    TERMINAL negative :  electrical;  
    TERMINAL positive :  electrical  
  );
END ENTITY battery;

Note that the model has two terminals and
the rest are simply parameters of the model.
VHDL-AMS Implementation: Architecture

```vhdl
-- Governing equation for concentration in particles in anode

4.0/3.0 (**(1.0*hpa, 3.0) - **(0.0*hpa, 3.0)) * csl_1' dot == -(diffDsa*(csl_2-csl_1)/hpa)*4.0*(1.0*hpa)*(1.0*hpa));
4.0/3.0 (**(2.0*hpa, 3.0) - **(1.0*hpa, 3.0)) * csl_2' dot == -(diffDsa*(csl_3-csl_2)/hpa)*4.0*(2.0*hpa)*(2.0*hpa) + diffDsa*(csl_2-csl_1)/hpa;
4.0/3.0 (**(3.0*hpa, 3.0) - **(2.0*hpa, 3.0)) * csl_3' dot == -(diffDsa*(csl_4-csl_3)/hpa)*4.0*(3.0*hpa)*(3.0*hpa) + diffDsa*(csl_3-csl_2)/hpa;
4.0/3.0 (**(4.0*hpa, 3.0) - **(3.0*hpa, 3.0)) * csl_4' dot == -(diffDsa*(csl_5-csl_4)/hpa)*4.0*(4.0*hpa)*(4.0*hpa) + diffDsa*(csl_4-csl_3)/hpa;
4.0/3.0 (**(5.0*hpa, 3.0) - **(4.0*hpa, 3.0)) * csl_5' dot == -(diffDsa*(csl_6-csl_5)/hpa)*4.0*(5.0*hpa)*(5.0*hpa) + diffDsa*(csl_5-csl_4)/hpa;
4.0/3.0 (**(6.0*hpa, 3.0) - **(5.0*hpa, 3.0)) * csl_6' dot == -(diffDsa*(csl_7-csl_6)/hpa)*4.0*(6.0*hpa)*(6.0*hpa) + diffDsa*(csl_6-csl_5)/hpa;
4.0/3.0 (**(7.0*hpa, 3.0) - **(6.0*hpa, 3.0)) * csl_7' dot == -(diffDsa*(csl_8-csl_7)/hpa)*4.0*(7.0*hpa)*(7.0*hpa) + diffDsa*(csl_7-csl_6)/hpa;
4.0/3.0 (**(8.0*hpa, 3.0) - **(7.0*hpa, 3.0)) * csl_8' dot == -(diffDsa*(csl_9-csl_8)/hpa)*4.0*(8.0*hpa)*(8.0*hpa) + diffDsa*(csl_8-csl_7)/hpa;
4.0/3.0 (**(9.0*hpa, 3.0) - **(8.0*hpa, 3.0)) * csl_9' dot == -(diffDsa*(csl_10-csl_9)/hpa)*4.0*(9.0*hpa)*(9.0*hpa) + diffDsa*(csl_9-csl_8)/hpa;
4.0/3.0 (**(10.0*hpa, 3.0) - **(9.0*hpa, 3.0)) * csl_10' dot == -(diffDsa*(csl_11-csl_10)/hpa)*4.0*(10.0*hpa)*(10.0*hpa) + diffDsa*(csl_10-csl_9)/hpa;
```

© 2010 ANSYS, Inc. All rights reserved.
Results: Newman 1d Model

Simplorer’s Results

White’s Results

Results: Newman 1d Model

Simplorer’s Results

White’s Results

What is Not Suitable for VHDL-AMS?

• PDEs in 3D
  – Discretization becomes complex without help of a commercial code.
  – It becomes tedious to write discretized equations.
  – Efficiency

• If a specialized solver is needed to solve the governing equations efficiently, then VHDL-AMS might not be as efficient.
  – 1D Euler equation in fluid dynamics
    • Flux difference splitting (Roe’s method)
Virtual Prototyping: Multi-domain Drive System Optimization

- Model Extraction from FEM / CFD
- Thermal Domain
- Electrical Domain
- Mechanical Domain
- VHDL-AMS Macro-Model
- IGBT Device Characterization
Thermal MOR Coupling Simpler Results
Conclusion

- VHDL-AMS is a means for users to include an existing mathematical model in system level simulations.

- VHDL-AMS programmers only need to write the governing equations after discretization. Solver and post-processing is provided by the simulating environment, for instance, Simplorer.

- VHDL-AMS can easily handle devices governed by algebraic, ODEs, and PDEs in 1D or even 2D.

- VHDL-AMS is a powerful technique contributing to the generation of system-level models in combination with physical modeling techniques.