Heterogeneous Physical Modeling
Jim Armstrong
Professor, ECE, Va. Tech
02-05-03
Modeling Goals

- Provide a *library component based* approach to the modeling of physical systems
- Employ principles of hierarchy
- Physical systems: biological, chemical, electronic
- Model the interaction between these various types of systems
- Provide an automatic synthesis path for the electronic components.
Component Library Based Modeling Requirements

• Requirements:
  – be able to define components
  – have a well defined notion of connectivity
  – specify timing and execution semantics for each environment (domains)
  – allow co-simulation of components from different domains

• Needed: heterogeneous modeling
Homogeneous Vs. Heterogeneous

- Dictionary:
  - Homogeneous: composed of parts all of the same kind (simple, straightforward)
  - Heterogeneous: composed of parts of different kinds. (handles complexity better)
Some Good Homogeneous Modeling Solutions
% Computes the root of the
% equation X**3 + X**2 -5*X -3

x = 0
i = 10

while (i > 0 )
x = 0.2*(x^3 + x^2 -3.0)
i = i-1
end

A contraction map!
entity MULT_B0 is
    port ( IN0 : in STD_LOGIC_VECTOR(7 downto 0);
          OUTPUT : out STD_LOGIC_VECTOR(7 downto 0) );
end MULT_B0;

architecture behavior of MULT_B0 is
constant RoundProdWidth_M_M1_1_1: INTEGER := 8;
beg
main: process(IN0)
variable Input2_M_M1_1_1: SIGNED(7  DOWNTO  0);
variable RoundProd_M_M1_1_1:
SIGNED(RoundProdWidth_M_M1_1_1 -1 DOWNTO 0);
begin
  Input2_M_M1_1_1 := const2fxp(0, 12, 1, 8);
  RoundProd_M_M1_1_1 := fxp_round(SIGNED(IN0) * Input2_M_M1_1_1,
                                   0, RoundProdWidth_M_M1_1_1);
  OUTPUT <= STD_LOGIC_VECTOR(fxp_saturate(RoundProd_M_M1_1_1, 1, 8));
end process main;
end behavior;
Synthesized DSP Multiplier
Some Complex Modeling Situations
System On A Chip

GP Processor

DSP Processor

Memory

Bus Interfaces

Glue Logic

Sensors

MEMS

Analog
111 psec speed of light transit time

.75 in.

9.375 ns RC Delay

\( f_C = 20 \text{ GHz} \)

\( \tau_c = 500 \text{ psec} \)

SOC Chip

When does the clock rise?
Globally Asynchronous, Locally Synchronous Time
Eye Opening Control System

Light (I) → Feedback → FIR

Noise
Case For Heterogeneity

• Model of Computation (MOC) : How is time handled?
• What modeling tools ?
  – handle different MOCs
  – combine MOCs
  – efficiency
  – plug into synthesis
The Nature of Time
What Time Is It?

\[ V = 10^8 \text{ m/sec} \]
References

• God’s Equation: “Einstein, Relativity and the Expanding Universe,” Amir Aczel
• “A Brief History of Time”, Steven Hawkings
• “The Arrow of Time” Coveney and Highfield
Timing Models

• Continuous Time
  – solutions of differential equations
• Discrete Event
  – abstract or hardware oriented
  – central or distributed
• Un-timed
  – execution ordering based on:
    • "topological sort" of component graph (statically scheduled)
    • separate process threads
The Eye Model
Response of Eye To Light

• As the intensity of light increases pupil size decreases
• As the intensity of light decreases pupil size increases
• The dynamics of the response are different depending on whether the light is increasing or decreasing
Bi-Modal Response Model

\[ H(s) = \Omega \left[ \frac{(1.2 \ s^2 \ e^{-0.3s})}{(1+.4s)(1+.46s)} \right] + \left[ \frac{(.24 \ s \ e^{-0.6s})}{(1+.1s)} \right] \]

If I is the input then

For \( \frac{dI}{dt} \geq 0 \), \( \Omega = 1 \)

For \( \frac{dI}{dt} < 0 \), \( \Omega = 0 \)
Other Model Properties

• Noise in optic nerve signal
• IIR Filter implanted to reject the noise (added electronics)
Model Response
First Tool:
Ptolemy
Ptolemy

• Developed at UC Berkeley (EECS Dept)
• Models are interconnection of Actors
• Choice of Director dictates:
  – Timing
  – Interconnection semantics
Actor Library

• Extensive Built-In Library: sources, sinks, math, DSP, data transformation, timing and sequencing control

• Many actors are *polymorphic*
  – data type
  – domain
Actor Development

• Actors are coded in JAVA
• C and C++ algorithms can be inserted into JAVA actor templates developed for modelers
• Template development requires knowledge of JAVA and the Ptolemy JAVA classes
• But experienced users can do it.
Example 2:
SOC Implementation of the Sobel Algorithm
Modeling Study: Sobel Edge Detection Alg.

Input Image $x(m,n)$

3 x 3 Sobel Filter

Output Image $y(m,n)$

Borders are one pixel wide and set to zero
Sobel Computations

\[
\begin{array}{ccc}
  a_0 & a_1 & a_2 \\
  a_7 & a_{ij} & a_3 \\
  a_6 & a_5 & a_4 \\
\end{array}
\]

\[
\begin{align*}
  \text{yy} &= (a_0 - a_6) + (a_2 - a_4) + C(a_1 - a_5) \quad \text{Vertical} \\
  \text{yx} &= (a_2 - a_0) + (a_4 - a_6) + C(a_3 - a_7) \quad \text{Horizontal} \\
  \text{y(i,j)} &= |\text{yx}| + |\text{yy}| \\
\end{align*}
\]

Sobel Calculations For Pixel a\textsubscript{ij}
Ptolemy Demo of SOC Sobel
Un-timed Modeling
The Ones Counter

SequencePlotter

output
sel

SDF Director

not valid

ArrayElement1

Accumulator

ones count

Multiplexor

SequenceP

DATA

CONTROL

Sequence Base

Array index

Start

inp==0

Done

inp==7

FSM Control

Control

true

start==true

sel=2

done==true

sel=0

out_1s_cnt

out_0s_cnt

cnt

idle

start==false

sel=2

done==false

sel=2

0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
Benefits of Un-timed Simulation

• Un-timed models are topologically sorted and then statically scheduled

• No event processing, thus very efficient simulation, important for long simulation

• Formerly used only for data flow applications, but can be adapted to sequential circuits

• Sequence number takes the place of bit time
Mixing Un-timed With Timed

- Timed domain in control
- Few timed events (require queue handling) initiate extensive un-timed activity (statically scheduled)
- Overall simulation very efficient
- Example: generating the Fibonacci series and the Golden Ratio
  - schedule controlled by input size
Fibonacci Numbers

Golden Ratio

HDE (timed)

SDF (untimed)
Limitations of High Level Modeling Tools

• Efficiency: speeds too slow for extensive model evaluation
• Model portability: how do you carry the model away to other environments?
• No good path to synthesis
Tool 2 : SystemC
What is SystemC?

- Industrially sponsored effort to develop a system level modeling language (systemc.org)
- C++ classes that support hardware modeling, e.g. concurrency and modularization
- A simulation environment from the web site and Synopsys and Coware tools, syntax checker from Willamette HDL
- Modeling process similar to HDL modeling
SystemC Example:

A Simple Counter
// counter.h
#include "systemc.h"
SC_MODULE(counter) {
sc_in<bool> clock;
sc_in<bool> load;
sc_in<bool> clear;
sc_in<sc_int<8> > din;
sc_out<sc_int<8> > dout;
int countval;
void onetwothree ();
SC_CTOR(counter) {
SC_METHOD(onetwothree);
sensitive_pos (clock);
countval = 0;
}
};

Counter Declaration
// counter.cc
#include "counter.h"

void counter::onetwothree () {
    if (clear == '1') {
        countval = 0;
    } else if (load == '1') {
        countval = din.read(); // use read when a type
        // conversion is happening
        // from an input port
    } else {
        countval++;
    }
    dout = countval;
}
GSM Model
- Complex Speech Encoder, Decoder
- Data Reformatting
- Long Latency
Online Demo

- Jack Kerouac: “In the morning frost the cats stepped slowly”
- Three au files:
  - Input speech file
  - Output speech file with no channel error
  - Output speech file with bit error rate of .001
A Bridge Needed

- Ptolemy
  - ?
  - SystemC
    - Hardware Synthesis
    - Embedded Code Synthesis
\[ C_m \frac{\delta T_m}{\delta t} = K \frac{\partial^2 T_m}{\partial x^2} - H(T_m - T_p) \]

\[ C_p \frac{\delta T_p}{\delta t} = H(T_m - T_p) \]
Finite Element Difference Equations

\[ M_{new} = M_{old} \left[ 1 - \frac{H\Delta t}{C_m} - \frac{2k\Delta t}{C_m\Delta x^2} \right] + P_{old} \left[ \frac{H\Delta t}{C_m} \right] + \left[ M_{old}^{right} + M_{old}^{left} \right] \left[ \frac{k\Delta t}{C_m\Delta x^2} \right] \]

\[ P_{new} = P_{old} + \frac{H\Delta t}{C_p} \left[ M_{old} - P_{old} \right] \]
Ptolemy Model
Results For Middle Cell
/fe.h
#include <systemc.h>
SC_MODULE(fe) {
  // Ports
  sc_in<double> ml_old;
  sc_in<double> mr_old;
  sc_inout<double> m_out;
  sc_inout<double> p_out;
  sc_in<bool> clk;
  // sc_signal<double> m_old;
  // sc_signal<double> p_old;
  // Parameters
  double dx;
  double dt;
  double h;
  double k;
  double cp;
  double cm;
  // Parameter Expressions
  double a,b,c,d;
// Methods
// Calculate a, b, c, d
// Process runs once at initialization
void abcd() { a = (h/cp)*dt;
    b = (h/cm)*dt;
    c = (dt*k)/(cm*(dx*dx));
    d = 1.0-b-2*c; }

// rhs pout and mout will be the old values
void calc_p() { p_out = (1.0-a)*p_out + a*m_out; }
void calc_m() { m_out = d*m_out + c*(ml_old+mr_old) + b*p_out; }

// Constructor
SC_CTOR(fe) {
    SC_METHOD(calc_p); sensitive << clk.pos();
    SC_METHOD(calc_m); sensitive << clk.pos();
    SC_METHOD(abcd);
    dx = .33;
    dt = .01;
    h = k = cp = cm = 1.0;}
};
Summary

• Theory, tools and techniques exist for heterogeneous physical modeling

• Models allow verification of:
  – Theoretical assumptions underlying the model
  – The model's interaction with other system components in other domains

• Models representing electronic components can be targeted to embedded processor code, ASIC, or FPGA implementation