# EECS 222C: System-on-Chip Software Synthesis Lecture 2

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

## Lecture 2: Overview

- The Concept of a Model
  - Modeling and abstraction
  - System-on-Chip model features
  - Separation of concerns
- The SpecC Model
  - Basic concepts
  - Intellectual Property (IP) components
- Application Case Study
  - MP3 decoder
- Assignment 1

## The Concept of a Model

- What is a Model?
  - Definition: A Model is an Abstraction of Reality.
- Examples of Models
  - Toy car
    - Abstract model of a real car
      - Smaller scale
      - Simpler, many details left out (no motor, no lights, …)
      - Less expensive, less dangerous
      - Less, but sufficiently functional
  - Architectural blueprint of a house
    - 2-dimensional model of a real building (3-dimensional)
      - Smaller, but to scale (floor plan, room sizes, window placement..)
      - Simpler, many details left out (no bricks, just paper)
      - Some features over-emphasized (e.g. layout of pipes, cables)
      - Less expensive, easy to estimate and modify



## The Concept of a Model

- What is a Model?
  - Definition: A Model is an Abstraction of Reality.
- What is Abstraction?
  - Part of model building
  - Simplification or omission of details
    - Some aspects of reality are simplified or omitted
  - Approach to reduce and factor out details
    - so that one can focus on a few features at a time
- What is Modeling?
  - Model building
  - To make or construct a model
- What is Specifying?
  - Modeling the initial model in a design flow

# Embedded System Models

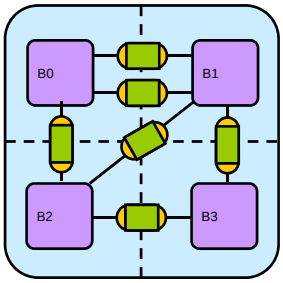
- Modeling an Embedded System
  - Decide what feature/property/characteristic
    - is needed (and to what degree)
    - is not needed (can be abstracted away)
- Typical Features in System-on-Chip Models
  - Functionality: important, most often needed (to a varying degree of accuracy)
  - Executability: important, often needed
  - Structure: increasingly needed in later design phases
  - Communication: needed to a varying degree of accuracy
  - Timing: needed to a varying degree of accuracy
  - Power consumption: sometimes needed, sometimes not
  - Temperature: usually not needed



# Co-Design Models: Hardware and Software

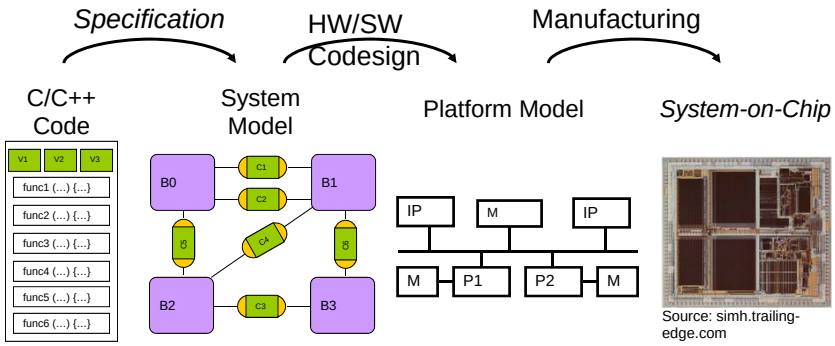
- System Level Modeling
  - Abstract description of a complete system
  - Software + Hardware
- Key Concepts in System Modeling
  - Explicit Structure
    - Block diagram structure
    - Connectivity through ports
  - Explicit Hierarchy
    - System composed of components
  - Explicit Concurrency
    - Potential for parallel execution
    - Potential for pipelined execution
  - Explicit Communication and Computation
    - Channels and Interfaces
    - Behaviors / Modules

System Model



# System-on-Chip Co-Design Flow

- Specification and Modeling
- Codesign: concurrent HW design and SW development
- Ongoing Research: Automatic Model Transformations




Source: simh.trailing-edge.com

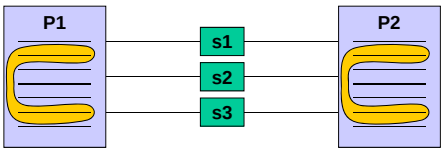


# Separation of Concerns

- Fundamental Principle in Modeling of Systems
- Clear *separation of concerns*
  - address separate issues independently
- System-Level Description Language (SLDL)
  - orthogonal concepts
  - orthogonal constructs
- System-level Modeling
  - Computation
    - encapsulated in modules / behaviors
  - Communication
    - encapsulated in channels

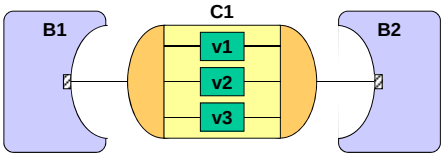
# System-Level Model

- Traditional model




  - Processes and signals
  - Mixture of computation and communication
  - Automatic replacement impossible

- SpecC model

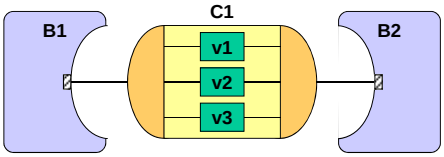



  - Behaviors and channels
  - Separation of computation and communication
  - Plug-and-play!

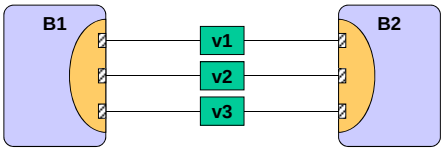


# System-Level Model

- SpecC Model
  - Behaviors
    - Computation
  - Channels
    - Communication
  - ➢ System Modeling!



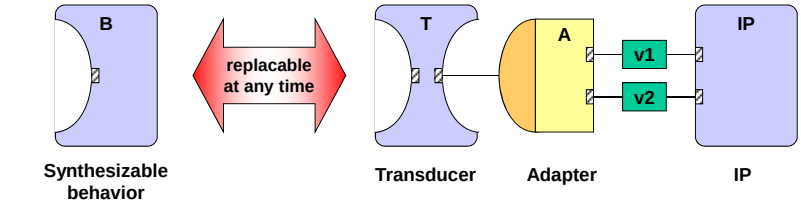

- Implementation through *Protocol Inlining*
  - Channel disappears
  - Communication is inlined into behaviors
  - Signals are exposed



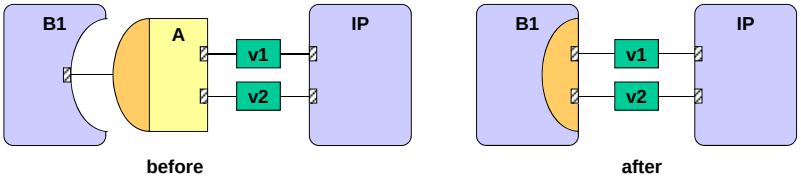

  - ➢ Model is converted to traditional model for implementation!

# Intellectual Property (IP)

- Computation IP: Adapter model



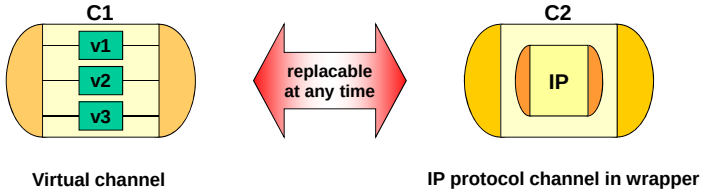

- Protocol inlining with adapter



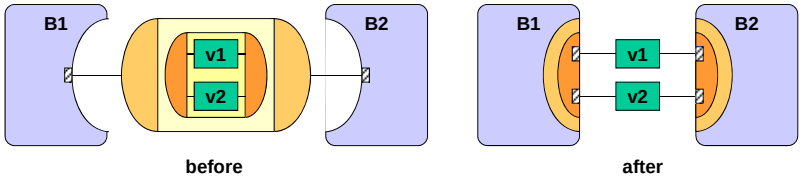



# Intellectual Property (IP)

- Communication IP: Channel with wrapper

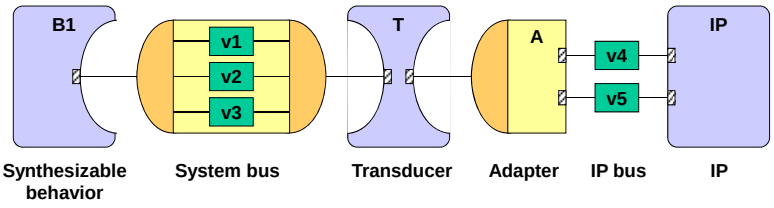



- Protocol inlining with hierarchical channel

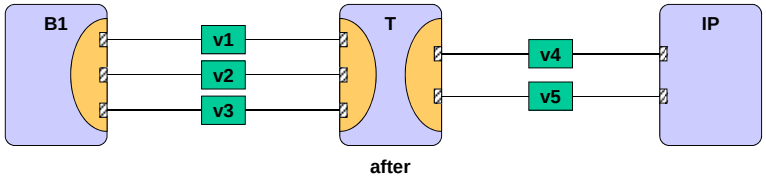


# Intellectual Property (IP)

- Incompatible busses: Transducer insertion



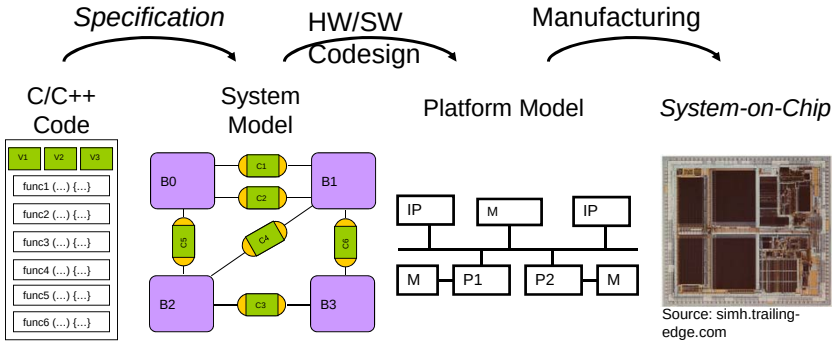

- Protocol inlining with transducer






# Application Case Study

- Taking an example application through the SoC Codesign flow
  - With the focus on software aspects
    - EECS222C: *SoC Software Synthesis*




Source: simh.trailing-edge.com

# Application Case Study

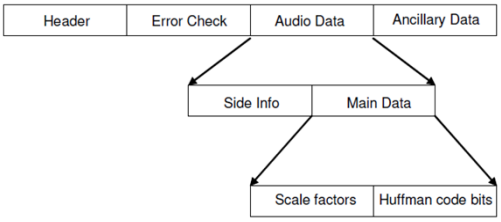
- Project Application: MP3 Audio Decoder
  - Digital compression of audio data reduces
    - Communication bandwidth and
    - Storage requirements
  - MPEG 1 Layer 3 (aka. MP3) compression algorithm
    - most commonly used
    - uses a variety of clever tricks to compress digital music
      - by 90% or more!
    - performs lossy compression
    - applies perceptual science of psycho acoustic models
      - exact input signal does not need to be retained
      - human ear can only distinguish a certain amount of detail
      - sufficient if output signal sounds identical to the human ears

[Source: CECS-TR-05-04.pdf]



# Application Case Study

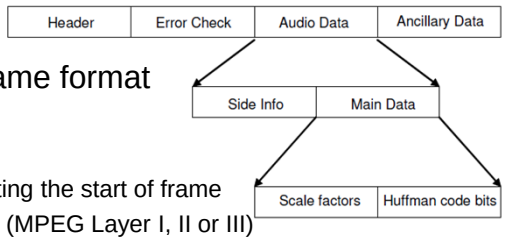
- Project Application: MP3 Audio Decoder
  - MP3 audio bit stream
    - organized in frames of bits
    - each frame contains 1152 encoded PCM samples
    - frame length depends on the bit rate (quality)
    - bit rate may vary in variable rate encoded streams
    - frame header contains information for the frame detection
  - MPEG 1 Layer 3 frame format




[Source: CECS-TR-05-04.pdf]

# Application Case Study

- Project Application: MP3 Audio Decoder
  - MPEG 1 Layer 3 frame format
    - Header
      - 4 bytes
      - Sync word indicating the start of frame
      - Layer information (MPEG Layer I, II or III)
      - Bitrate information
      - Sampling frequency
      - Mode information (mono or stereo)
    - Error Check
      - 16 bit parity check for optional error detection

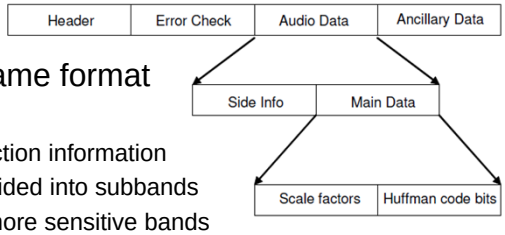



[Source: CECS-TR-05-04.pdf]



# Application Case Study

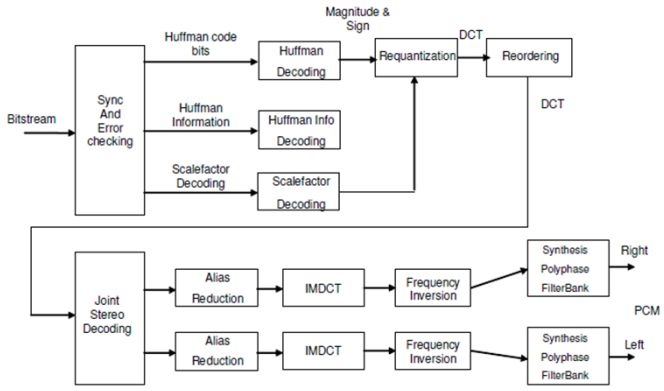
- Project Application: MP3 Audio Decoder
  - MPEG 1 Layer 3 frame format
    - Side Information
      - Scale factor selection information
        - » spectrum divided into subbands
        - » samples in more sensitive bands are scaled more than others
      - Global gain (to be applied to all samples)
      - Number of bits used to encode scalefactors
      - Huffman table selection (1 out of 32 Huffman tables)
    - Main data
      - Scale factors
      - Quantized values encoded using Huffman codes




[Source: CECS-TR-05-04.pdf]

# Application Case Study

- Project Application: MP3 Audio Decoder
  - MP3 decoder block diagram

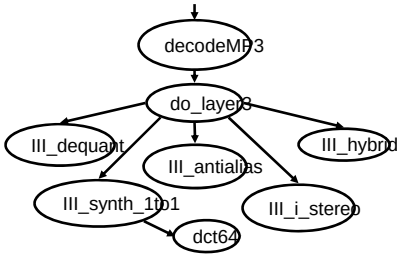



[Source: CECS-TR-05-04.pdf]



# Application Case Study

- Project Application: MP3 Audio Decoder
  - MP3 decoder C reference code
    - Underbit Technologies Inc.
    - MAD: MPEG Audio Decoder
    - http://www.underbit.com/products/mad




Partial function hierarchy in MP3 reference code

[Source: P. Chandraiah]

# Assignment 1

- Administration
  - Linux Servers
    - **`alpha.eecs.uci.edu`** (NSF client)
    - **`gamma.eecs.uci.edu`** (NSF client)
    - **`mu.eecs.uci.edu`** (NSF host)
    - Intel Pentium based PCs
    - RedHat Linux (Fedora Core 12)
    - Access via secure shell protocol (**`ssh`**)
  - Accounts
    - User ID same as your UCI net ID
    - Password as discussed in class
  - SpecC Software (© by CECS, UCI)
    - SpecC Compiler and Simulator
    - System-on-Chip Environment (SCE)



# Assignment 1

- Login on Server via SSH
  - Account infos will be emailed
- Install MP3 Decoder example
  - **`mkdir eecs222c`**
  - **`cd eecs222c`**
  - **`gtar xvzf /home/doemer/EECS222C/mad_C.tar.gz`**
  - **`cd mad_C`**
  - **`make clean`**
  - **`make`**
  - **`make test`**
- Become familiar with the application and its structure
  - Browse and read the source files
  - Draw a block diagram of the major functions