

EECS 222A: System-on-Chip Description and Modeling Lecture 9

Rainer Dömer

doemer@uci.edu

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

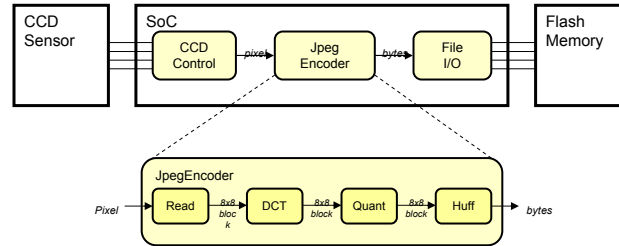
Lecture 9: Overview

- Project Discussion: Digital Camera
 - Assignment 2: JPEG Application Structure
 - Assignment 3: Sequential SpecC Model
 - Assignment 4: Sequential SpecC Model in SCE
 - Assignment 5: Parallel SpecC Model for Synthesis
 - Assignment 6: Design Space Exploration
 - Discussion, Q&A
- Modeling with SystemC SLDL
 - SystemC 2.0 Tutorial
 - Presentation by Thorsten Groetker, Synopsys
 - Assignment 1: Sender/Receiver Example in SpecC
 - Assignment 7: Sender/Receiver Example in SystemC

Project Discussion: Assignment 2

- Digital Camera Example

- Component Model



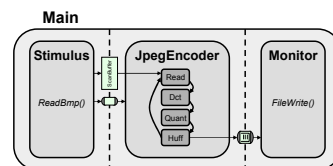
- Homework Assignment 2

- Become familiar with JPEG Encoder Application
 - Study reference code:
`/home/doemer/EECS222A_F09/jpegencoder.tar.gz`
 - Draw block diagram of files, functions, and key communication variables
 - Simplify code for a 116×96 pixel CCD (eliminate `malloc` calls!)

Project Discussion: Assignment 3

- Task

- Convert JPEG Encoder Application to a proper SpecC Specification Model



- Deliverables

- Email to `doemer@uci.edu` with subject "EECS222A Assignment 3"
 - Brief status description (in body of your email)
 - Source code `jpegencoder.tar.gz` (attachment)

- Due

- Next week: October 30, 2009, 12pm (noon)

Project Discussion: Assignment 3

- Hint:
 - Use the `sir_tree` tool to validate your hierarchy
 - The final model should look like this:

```
doemer@epsilon.eecs.uci.edu:3 > sir_tree -blt digicam.sir
B i o   behavior Main
B i o   |----- JpegEncoder jpeg
B i s   |           |----- Dct dct
B i l   |           |           |----- Bound bound
B i l   |           |           |----- ChenDct chendct
B i l   |           |           \----- Preshift preshift
B i s   |           |----- Huff huff
B i l   |           |           |----- Huffencode huffencode
B i l   |           |           \----- Zigzag zigzag
B i l   |           |----- Quantize quantize
B i l   |           \----- ReadBlock readblock
B i l   |----- Monitor monitor
B i l   |----- Stimulus stimulus
C i l   |----- c_queue data
C i l   \----- c_handshake start
```

EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

5

Project Discussion: Assignment 4

3. Analyze your digital camera model in SCE
 - Setup
 - ...continued from step 2 (previous page)
 - View the structural hierarchy chart
 - Select the **Main** behavior in the behavior browser
 - Right-click ->**Chart**
 - Double-click the chart to add further levels of hierarchy
 - Turn on connectivity **View->Connectivity**
 - **Window->Print...** to file "**digicam.ps**"
 - In your shell window, convert the PostScript file to PDF:
`ps2pdf digicam.ps`
 - Check the PDF file: `acroread digicam.pdf`
 - Deliverables
 - Hierarchy chart
 - "**digicam.pdf**"
 - Due
 - by Friday, Nov 6, 2009, at noon
 - by email to doemer@uci.edu with subject "EECS222C HW4"

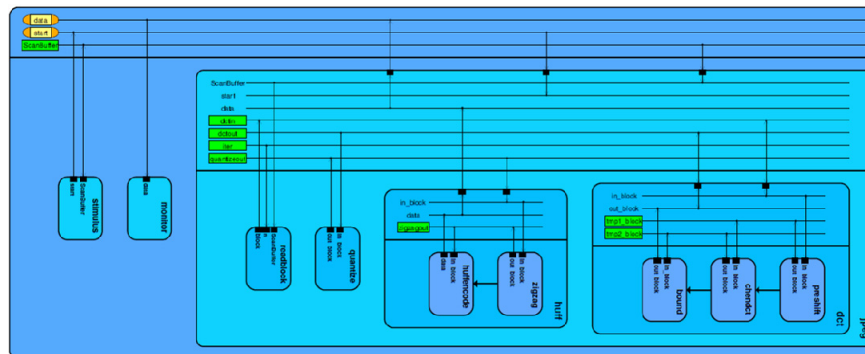
EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

6

Project Discussion: Assignment 4

- Sequential Digicam Model
 - Screen shot of Hierarchy Chart in SCE
 - (rotated by 90 degrees)

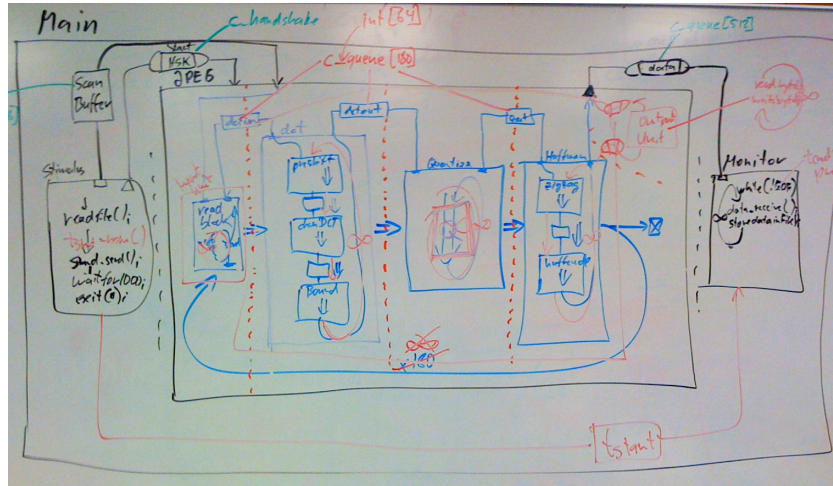


– jpegencoder2.tar.gz

Project Discussion: Assignment 5

- Create a parallel/pipelined and synthesizable Digicam model
 - Start from previous model
 - /home/doemer/EECS222A_F09/jpegencoder2.tar.gz
 - Insert timing checks into the test bench
 - Add explicit I/O units
 - Parallelize/pipeline the **JpegEncoder** block
 - Modify **Dct**, **Quantize** and **Huffman** to infinitely work on continuous streams of data over typed queue channels
 - Validate functionality
 - Print hierarchy chart with connectivity
- Deliverables
 - Status description
 - Source code "digicam.tar.gz"
 - Hierarchy chart "digicam.pdf"
- Due
 - by Friday, Nov 13, 2009, at noon
 - by email to doemer@uci.edu with subject "EECS222C HW5"

Project Discussion: Assignment 5



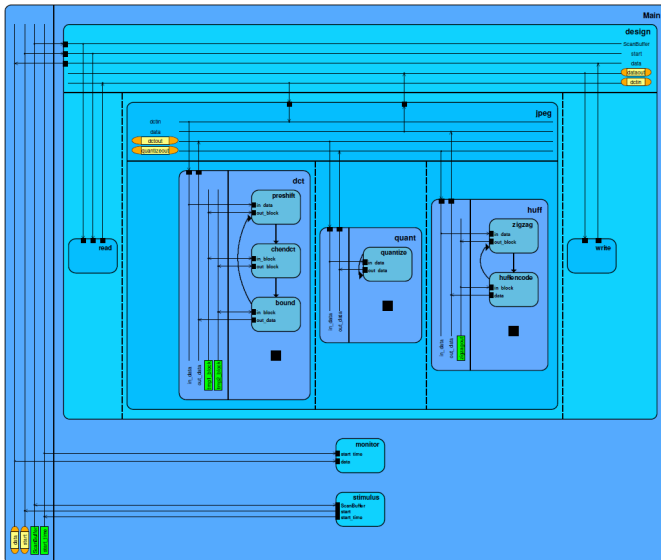
EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

9

Project Discussion: Assignment 5

- Pipelined Digicam Model
 - Explicit I/O Units
 - Explicit Pipelining
 - Communication via Queues
-
- jpegencoder3.tar.gz



EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

10

Project Discussion: Assignment 6

- Design Space Exploration using SCE
 1. Profile, analyze, estimate the digicam model (`jpegencoder3.tar.gz`)
 - For a single ARM_7TDMI CPU at 100MHz
 - For metric of "Computation"
 - For blocks `read`, `dct`, `quant`, `huff`, `write`
 - Create a bar chart of the Computation Profile
 - "`ARM100.pdf`"
 2. Create a software-only reference model
 - Allocate and Map
 - 1 ARM7TDMI CPU at 100MHz as "ARM100" for JpegEncoder
 - 1 Custom HW_Standard block at 100Mhz as "InputUnit" for ReadBlock
 - 1 Custom HW_Standard block at 100Mhz as "OutputUnit" for WriteBlock
 - Estimate the allocation (Validation->Evaluate)
 - Perform Architecture Refinement
 - Perform Scheduling Refinement
 - Use Round-Robin scheduling policy on ARM100
 - When executed, the resulting model should encode our test picture in 39.252ms

EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

11

Project Discussion: Assignment 6

- Design Space Exploration
 3. Architecture Exploration
 - Explore various other system architectures
 - Use up to 5 ARM_7TDMI processors
 - » Clock frequency may be 100, 150, or 200MHz
 - » Cost is assumed at \$100, \$150, \$200, respectively
 - » Use only 50MHz AMBA AHB bus
 - Use up to 5 HW_Standard accelerator blocks
 - » Clock frequency may be 100, 200, 300, or 400MHz
 - » Cost is assumed at \$200, \$400, \$600, \$800, respectively
 - Vary the mapping of blocks in the DUT to CPUs and HW units
 - Vary the scheduling policy as needed
 - Example:
 - Use 1 HW100 for DCT
 - Use 1 ARM100 for everything else

EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

12

Project Discussion: Assignment 6

- Design Space Exploration
 4. Scheduling Exploration
 - Explore various scheduling strategies for each selected CPU
 - Choose from
 - Static scheduling
 - » with varying execution order
 - Round-Robin scheduling
 - Priority-based scheduling
 - » with varying priorities
 - Example:
 - ARM100 scheduled with round-robin
 - ARM200 scheduled with priority-based scheduling
 - Static scheduling on HW blocks

EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

13

Project Discussion: Assignment 6

- Design Space Exploration
 5. Deliverables
 - Bar chart of the software-only computation profile
 - “ARM100.pdf”
 - Text file with table of 3 “good” architectures
 - List the allocation and mapping for each block
 - Simulate to estimate the resulting encoding delay
 - Calculate the assumed cost of the architecture
 - Due
 - by Friday, Nov 20, 2009, 2pm
 - by email to doemer@uci.edu with subject “EECS222A HW6”
 - bring a copy for discussion in class!

EECS222A: SoC Description and Modeling, Lecture 9

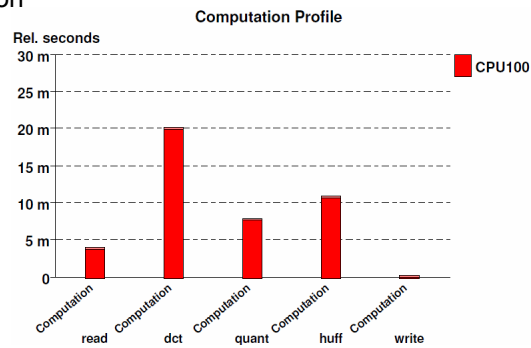
(c) 2009 R. Doemer

14

Project Discussion: Assignment 6

- Bar chart of the software-only computation profile
 - single ARM_7TDMI CPU at 100MHz
 - metric of “Computation”
 - “ARM100.pdf”

- Read: 3.9ms
- Dct: 20.1ms
- Quant: 7.8ms
- Huff: 10.8ms
- Write: 0.1ms
- Design: 42.7ms



- With assignment of dedicated I/O units, encoding time in simulation is 39.252ms

EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

15

Project Discussion: Assignment 6

- Design Space Exploration: Discussion in Class
 - Cost / performance trade-off graph
 - Visualize design space for “good” architectures
 - Issues:
 - ARM7TDMI frequency range patched
 - HW-only solution!?
 - One single HW PE
 - Multiple HW PEs
 - Scheduling policies
 - Round-robin vs. Priority-based Scheduling
 - » Both non-preemptive!
 - Static scheduling
 - » Not applicable for multiple behaviors with infinite loops

EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

16

SystemC 2.0 Tutorial

- Presentation by Thorsten Groetker, Synopsys, 2001
 - Motivation
 - Models of Computation
 - Model of Time
 - Communication, Interfaces and Channels
 - Platform Modeling
 - Transaction-level Model, Examples
 - Benefits
 - Summary

EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

17

Assignment 1: Sender/Receiver in SpecC

- Task: Introduction to SpecC Compiler and Simulator
 - Become familiar with `scc`
 - See `man scc` for manual page
 - Use `scc` to compile and simulate the examples in
 - `/opt/sce-20080601/examples/simple/`
 - Build and simulate the sender/receiver example
 - See Slide 25! (behavior `B` should be `Main`)
 - Sender `s` should send values 0.0, 0.5, ... 5.0 to the receiver `R` which prints them to the screen
- Deliverables
 - Source file: `SendReceive.sc`
 - Simulation log: `SendReceive.log`
- Due
 - By next week: October 9, 2009, 12pm (noon)
 - Email to `doemer@uci.edu` with subject "EECS222A Assignment 1"

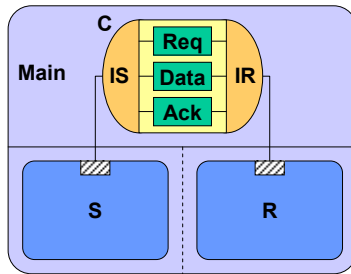
EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

18

Assignment 1: Sender/Receiver in SpecC

- Add behavior **Main**
- Add loop to **S**
- Add loop to **R**
- Compile and Simulate
- Done!



```
behavior S(IS Port)
{
    float X;
    void main(void)
    { ...
      Port.Send(X);
      ...
    }
};

behavior R(IR Port)
{
    float Y;
    void main(void)
    { ...
      Y=Port.Receive();
      ...
    }
};
```

```
interface IS
{
    void Send(float);
};
interface IR
{
    float Receive(void);
};

channel C
    implements IS, IR
{
    event Req;
    float Data;
    event Ack;

    void Send(float X)
    { Data = X;
      notify Req;
      wait Ack;
    }

    float Receive(void)
    { float Y;
      wait Req;
      Y = Data;
      notify Ack;
      return Y;
    }
};
```

EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

19

Assignment 7: Sender/Receiver in SystemC

- Goal: Introduction to SystemC
 - Review the producer/consumer example by Stuart Swan
 - See `Lecture8_SystemC_Intro.pdf`
 - Compile and simulate the example using SystemC
 - `mkdir SystemC ; cd SystemC`
 - `cp /opt/pkg/systemc-2.1.v1/examples/systemc/simple_fifo.cpp .`
 - `g++ simple_fifo.cpp -I/opt/pkg/systemc-2.1.v1/include -L/opt/pkg/systemc-2.1.v1/lib-linux -lsystemc -o simple_fifo`
 - `./simple_fifo`
 - Model and simulate the sender/receiver example from Assignment 1 in SystemC
 - Reference: `/home/doemer/EECS222A_F09/SendReceive.sc`
- Deliverables
 - Source file: `SendReceive.cpp`
 - Simulation log: `SendReceive.log`
- Due
 - In two weeks: December 4, 2010, 12pm (noon)
 - Email to `doemer@uci.edu` with subject "EECS222A Assignment 7"

EECS222A: SoC Description and Modeling, Lecture 9

(c) 2009 R. Doemer

20