Hardware Software Codesign with FPGAs

Instructor:

Prof. Jim Plusquellic

Text:

- A Practical Introduction to Hardware/Software Codesign, 2nd Edition", Patrick Schaumont, Springer, ISBN 978-1-4614-3736-9, 978-1-4614-3737-6 (eBook)

References:

- Hardware/Software Co-Design - Principles and Practice"
  J. Staunstrup and W. Wolf, Kluwer Academic Publishers, 1997.
  ISBN: 0792380134
- "Co-Design for System Acceleration A Quantitative Approach"
  Nadia Nedjah, Luiza de Macedo Mourelle, 2007.
  ISBN: 978-1-4020-5545-4
- "Embedded System Design, A Unified Hardware/Software Introduction"
  Frank Vahid and Tony Givargis, 2002.
  ISBN: 978-0-471-38678-0

Face-to-face Website: http://www.ece.unm.edu/jimp/codesign

Introduction

Relevant Conferences/Symposia on Codesign:

- Conference on Formal Methods and Models for Codesign (MEMOCODE)

- CODES+ISSS: The premier conference for System-Level Design, Embedded Systems Week

  The CODES+ISSS Conference is the merger of two major international symposia on hardware/software codesign and system synthesis.

- DAC: Design Automation Conference

- ASP-DAC: Asia South Pacific Design Automation Conference

- CASES: International Conference on Compilers, Architecture, and Synthesis for Embedded Systems

- ICCAD: International Conference on Computer Aided Design

The Nature of Hardware and Software

What is H/S Codesign (Prof. Schaumont's definition):

Hardware/Software Codesign is the partitioning and design of an application in terms of fixed and flexible components

Other definitions

- HW/SW Codesign is a design methodology supporting the **concurrent** development of hardware and software (cospecification, codevelopment and coverification) in order to achieve *shared functionality and performance goals* for a combined system

- HW/SW Codesign means *meeting system level objectives* by exploiting the **synergism** of hardware and software through their concurrent design

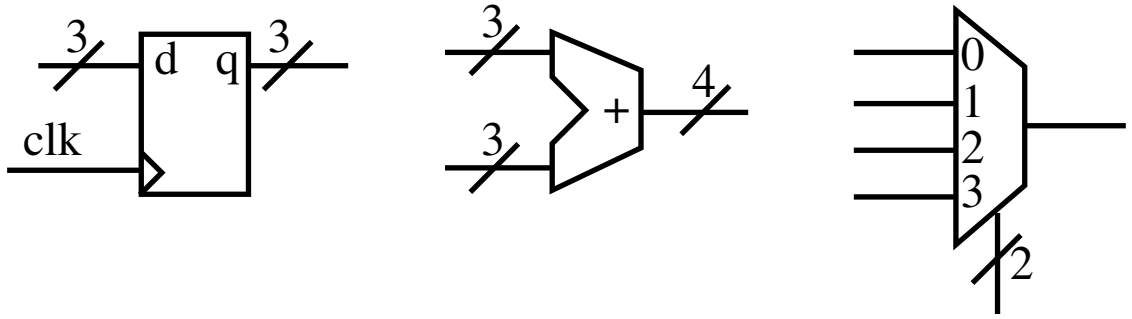
  Giovanni De Micheli and Rajesh Gupta, "Hardware/Software Co-design", IEEE Proceedings, vol. 85, no.3, March 1997, pp. 349-365

- Codesign is the concurrent development of hardware and software

Hardware

We assume **hardware** refers to *single-clock synchronous* digital circuits

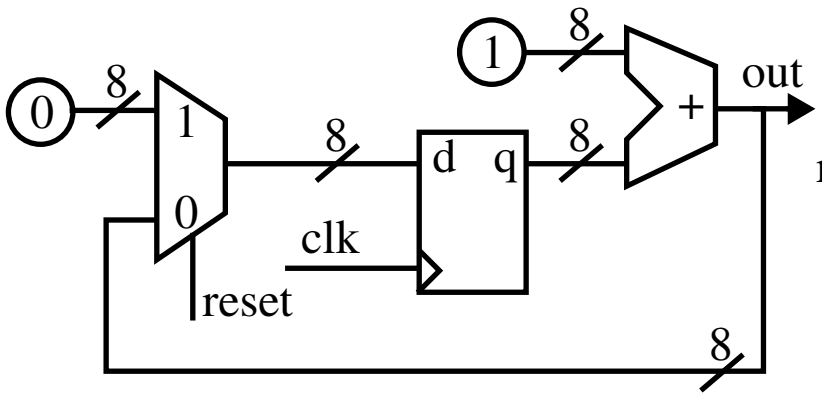
Hardware is realized by word-level combinational and sequential components, such as registers, MUXs, adders and multipliers



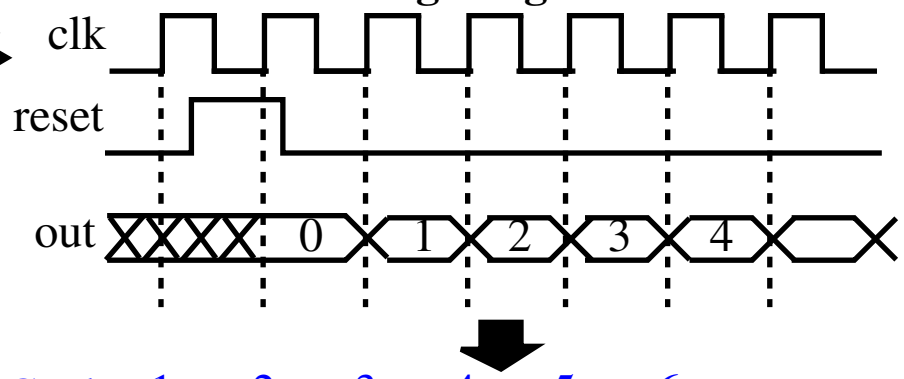
Cycle-based word-parallel hardware modeling is called **register-transfer-level** (RTL) modeling

RTL refers to a description in which the design is **abstracted** as a set of operations performed on data as it is transferred between registers

Hardware



Timing Diagram



This model allows us to **ignore** all events in between steps, and use a set of numbers to represent behavior, i.e. waveforms not needed →

Cycle	1	2	3	4	5	6
Reset	0	1	0	0	0	0
out	x	0	1	2	3	4

Bear in mind that this is a very *simplistic* treatment of actual hardware

We ignore advanced circuit styles including *asynchronous hardware*, *dynamic logic*, *multi-phase clocked hardware*, etc.

The cycle-based model is **limited** because it does not model *glitches*, *race conditions* or events that occur within clk cycles

However, it provides a convenient abstraction for a designer who is mapping a behavior, e.g., an algorithm, into a set of discrete steps

Software

We assume **software** refers to a *single-thread sequential* program written in C or assembly program

Programs will be shown as listings, e.g., **Listing 1.1** C example

```
1 int max;
2
3 int findmax(int a[10]) {
4    unsigned i;
5    max = a[0];
6    for (i = 1; i < 10; i++)
7       if (a[i] > max) max = a[i];
8 }
```

Software

Listing 1.2 ARM assembly example

```
                ldr     r2,   .L10          "max = a[0];"
                ldr     r3,   [r0, #0]      "max = a[0];"
                str     r3,   [r2, #0]      "max = a[0];"
                mov     ip,   #1             "for loop"
.L7:
                ldr     r1,   [r0, ip, asl #2] "scale i"
                ldr     r3,   [r2, #0]       "read a[i]"
                add     ip,   ip, #1         "i++"
                cmp     r1,   r3             "a[i] > max"
                strgt   r1,   [r2, #0]       "cond. store"
                cmp     ip,   #9             "i < 10"
                movhi   pc,   lr            "cond. return"
                b       .L7                   "uncond. br"
.L11:
                .align  2
.L10:
                .word   max
```

Hardware and Software

Hardware and software are *modeled* using RTL and C programs

A designer creates a model, i.e., an abstract representation, of the system from a specification

For example, a VHDL behavioral description is a *model* of the hardware circuit, which consists of gates and wires

Similarly, a C program is a *model* of a set of micro-processor instructions, i.e., a binary

NOTE: C programs are rarely referred to as models, and instead many generally consider them as actual implementations

Models and programs are used as input to simulation and implementation tools

Hardware/software codesign uses both models (RTL) and programs (C) as descriptions of a system implementation

Hardware/Software Ambiguities

There are many examples of systems in which the distinction between hardware and software is not always crystal clear

For example:

- An FPGA is a hardware circuit that can be reconfigured

  The **program** for an FPGA is a **bitstream**, which is used to configure its logic function

  VHDL and verilog are used to generate software models, which in turn are translated into bitstreams that configure the hardware

- A **soft-core** is a processor configured into an FPGA's reconfigurable logic

  The soft-core can then be used to execute C programs

- A **digital signal processor** (DSP) has instructions which are optimized for signal-processing applications

  Efficient programs require detailed knowledge of the DSP HW architecture, which creates a strong relationship between software and hardware

Hardware/Software Ambiguities

- An **application-specific instruction-set processor** (ASIP) is a processor with a customizable instruction set

  Users can design customized instructions for the processor, which are later referenced in programs

- The **CELL processor**, used in the Playstation-3, contains one control processor and 8 slave-processors, interconnected through a high-speed on-chip network

  The software for a CELL is a set of 9 concurrent communicating programs and configuration instructions for the on-chip network

A common characteristic of all these examples is that creating the SW requires **intimate familiarity** the HW