EEE4084F: Digital Systems

19 April 2012

Quiz 3

Lectures 10-15; Seminar 7: Ch 9 pp. 191-197

Time: 45 minutes 40 marks

SAMPLE SOLUTION

Marking procedure: Each **bold underlined** word indicates something expected in the answer. Each of these can be given one mark. Note that there are generally more bold words in the answer than marks allocated to the question but the number of marks awarded to an answer cannot exceed the maximum number of marks allocated to the question in the question paper.

Question 1 [18 marks] *IMPORTANT:* try to answer this question as it relates to ECSA requirements.

This question relates to the conceptual assignment.

(a) Briefly recap in a sentence or two what the WIISMA conceptual system is. [4 marks]

A: The WISSMA is a **multi-core processor/architecture** design. It executes **wide instructions** which combine a collection of individual instructions for each processor core. Each core in the WISSMA **runs synchronously** in lock-step (one instruction per clock). The WISSMA combines a **set of arithmetic cores** (Acores) and a Control Core (Ccore) together with **memory control unit** and **instruction cache**.

(b) The proposed architecture had a set of three general purpose shared registers, called SA, SB, SC. These were used to hold arithmetic operations. In your solution to the assignment did you choose to add any shared variables? Is so briefly elaborate why.[4 marks]

A: Having three Acores and one Ccore makes the wide instruction 128 bits (16 bytes) in length. This means that 16 bytes have to be loaded from the instruction cache before the instructions can start executing. This may cause sequential, non parallel parts of the code to run slowly as it could mean most of the instructions loaded are NOP instructions that leave cores idle. For this type of system having less than 3 shared registers are too few. Having a few more would probably make sense considering it is designed as a MISD architecture designed for specific types of application. So I would suggest adding at least another two Acores to make a total of 6 cores. This means more shared registers are needed. There are generally too few registers to share between the cores, e.g. for passing on intermediate results. I would recommend having at least two 'input' registers and two 'output' shared registers per

core, that would be (in the case of a 6-core system) 6 x 2 = 12 shared general purpose registers. There can only be one shared memory register (due to the MISD structure). Furthermore a register to share flags is necessary, e.g. if ACore1 does a comparison it would need to pass its result to the CCore for it to perhaps perform a branch. Each core would need a H (hold), Z (zero / equal), C (carry), GT (greater than), LT (less than) and NEG (negative value) flag – in all 6 x 5 = 30 bits (for the 5 Acores) which fits comfortably into a single 32-bit word.

(c) In the WIISMA processor design there were three Acores proposed and one Ccore (you had the option to add or remove Acores). Discuss the implications of adding or removing Acores. Focus on the benefits and drawbacks in terms of both programs that run on the system or the cost of manufacturing the system. [10]

A: Adding cores has the drawback of using up more space / silicon (or logic elements if prototyped on a FPGA). This can also further reduce speed of sequential code and loading the ICache for blocks of sequential code. But on the positive side it could make parallelized parts run faster.

Reducing the number of Acores can make the sequential parts of code load and run faster, but it is a tradeoff as the parallelized parts will be less parallel and possibly end up in reduced overall performance.

Question 2 [12 marks]

This question relates load balancing and performance analysis of parallel programs.

(a) Explain why idle time is an effective measure for determining how effectively balanced processor loads and, and similarly why the process of load balancing can be expressed as the minimization of idle time across processors. (You can include a diagram should you think this would aid your description) [10 marks].

A: Idle time is easy to measure as most O/S keep track of this. Idle time is a measure of processors being *unused*. The inverse of idle time is 'busy time' – we want to overall maximize busy time across all processors (by the definition of load balancing) which is equivalent to minimizing the overall idle time.

(b) The *gettimeofday* function is often said to be the preferred way to time sections of code within a program running on Linux. Briefly motivate why this is so. [2 marks]

A: It is highly portable (it is a requirement to conform to the StdC library). It also gives microsecond accuracy (millisecond accuracy in Cygwin) which is a good level of resolution for most performance evaluation needs.

Question 3 [10 marks]

This question relates to Chapter 9 of the textbook.

(a) The terms MOSFET and CMOS have become widely used in computer engineering literature related to ASIC design. What is CMOS? What precisely is the relation between MOSFET and CMOS? Why is it that CMOS tends to be the preferred choice for designing ASIC circuits? [5 marks]

A: CMOS stands for Complementary MOSFET. CMOS is defined as a circuit that combines complimentary MOSFETS, i.e. combines p-type and n-type MOSFETS in the same circuit (is is not a special type of MOSFET). CMOS is the preferred choice at it takes little space (especially when using dynamic logic) and is reliable, and can be used to build very low-power circuits.

(b) There are two general types of CMOS circuit: static logic and dynamic logic. By why do ASIC designers bother with static logic – surely static logic is pointless for ASIC designs since all the circuit elements are surely dynamic in nature, doing things and changing state. Briefly describe the advantage of static logic over dynamic logic and why static logic can be so useful for ASIC designers. [5 marks]

A: This is discussed on page 196 of the text book (you did not have to read pg. 197). Static logic is used a great deal as it is lower power. It is also simpler to build circuits with this. Static logic is very useful for implementing storage (i.e. storage that can be maintained with virtually 0 power drain except when the data is changed), and also produces less heat (e.g. reducing need for heat sinks or fans). Dynamic logic has to actively keep drawing power to keep its state refreshed, even if it is just storing data that is not being changed.
