System Level Design and Modelling
4 May 2007
Major Exam, 3:30 PM to 5:30 PM
Maximum Marks: 50

1. [10 Marks] Consider a simple merge sort algorithm for sorting 1024 elements stored in the memory. The result again needs to be stored back in the memory. Assume the processor does not support cache. Compare the following three implementations from both cost as well as energy considerations.
   (a) Pure software implementation on a simple RISC processor
   (b) Software + Special ALU for "Compare and Swap" on the RISC processor
   (c) An accelerator for sorting one complete sub-list as part of merge algorithm

2. [10 Marks] Describe the role of communication and synchronization delay in hardware-software partitioning with a simple example.

3. [10 Marks] Consider a behaviour that is dominated by the following loop:

```
for (i = 0; i < 10000; i++)
  a = a + b[i];
```

Suggest a low power memory organisation for the above behaviour and compare the expected power consumption with a standard data cache configuration.

4. [3+3+4=10 Marks] The following code is to be executed on a processor with a standard data cache architecture.

```
int A[1000] = {2,4,3,5,...}; // All values known at compile time
int B[1000] = {3,4,1,6,...}; // All values known at compile time

int f (int k)
{
  int t = 0;

  for (i = 0; i < k; i++)
    t = t + A[i]*B[i];
  return t;
}

int main ()
{
  int k = input (...); // Input obtained at run time
  int r = f (k);
  output (r); // Send r to some output device
}
```

A and B are arrays of integer constants, whose values are known at compile time. Suppose it is known that the value of k passed as a parameter to f is known to be biased towards the smaller values (say $k < 100$). Indicate mechanisms to transform the code to improve performance and power if it is given that:
   (a) you can use any amount of additional memory
   (b) you can use a small amount ($= r$) of additional memory
   (c) you can use a small amount ($= r$) of additional memory and the value of k is NOT biased towards smaller values; it is uniformly distributed between 1 and 1000.

5. [5+5=10 Marks] In SystemC SC_METHOD processes, there is a relevant construct called *next_trigger*. When invoked as *"next_trigger (t);"*, this is an instruction to the SystemC simulator that the next triggering of the process should be after time t, overriding the process sensitivity list. Similarly, one can also pass an event e to next_trigger, asking the simulator to trigger the process next only when the event e occurs. In general, the parameters to next_trigger can be as general as the parameters to *wait()*. After the call to next_trigger, execution continues to the following statement. Note that next_trigger ONLY AFFECTS THE VERY NEXT TRIGGERING of the process. The subsequent triggerings occur according to the sensitivity list.

(a) Can this mechanism be used to completely eliminate waits and SC_THREADs (being replaced by next_trigger and SC_METHOD respectively)? Explain why or why not.

(b) Give a method to generate a clock-type of signal using SC_METHOD and next_trigger. Compare this implementation to an equivalent implementation using SC_THREADs. Which is more efficient?