System-on-Chip Design Supervision Answer (Set 3)

Wei Song 15/05/2016

This answer provides a summary of key points in each question. It is the understanding of the supervisor, which does not necessarily always correct and does not represent the view of lecturers. It is expected to expand the key points into detailed description when similar questions are asked in exams.

SP 6. ESL and TLM

Q1: Briefly explain how and why an ESL model that uses a TLM model of its buses can run the embedded software with no modification to its device drivers.

The key is that the abstract bus function model and its interface to software is the same no matter what the hardware model is implemented, TLM or RTL. The difference in bus model only affects differences in collecting power/performance statistics. A TLM bus model and an RTL-level bus model will each model the same number of bus transactions as the device driver runs but the style of modelling is very different in detail.

Q2: Explain how the device driver for an on-chip network might be modified if the network devices itself is not to be modelled and instead transactions are to be used to directly pass packets between network nodes.

The API to the device driver and the code for higher-level internal functions (such as CRC check and credit-based flow control) should not be changed. But rather than copying the packet body from its local bus to the NoC interface, the complete payload is sent directly to the receiving peer in a single transaction, normally a function call. Note that this may greatly affect the accuracy in terms of transactions and time.

Q3: What is the difference between a blocking and non-blocking transaction in terms of implementation, efficiency and callability?

Blocking might be easier to use but introduce the overhead of using threads. When a thread is blocked, it holds up the calling client. For non-blocking, the results comes back immediately. So it does not block the calling client. However, the client needs to repeatedly call until success, which introduces overhead as well.

When using blocking transactions, hardware flow is implicitly controlled by thread's call and return while non-blocking transactions rely on poll/retry.

In SystmeC, blocking must use SC_THREAD while non-blocking usually uses SC_METHOD.

Q4: Sketch a SystemC code for a shim function that converts a transactional port from blocking to non-blocking.

```
Function blocking(arg) {
   while(!nonblocking(arg)) continue;
}
```

Q5: What is the advantage of putting a reference-passed delay parameter in a TLM call?

The most significant advantage is faster simulation if the performance in concern is only aggregated delay. Putting a reference-passed delay in a TLM call allows the TLM to run ahead without stall the thread. This causes loosely-timed simulation but the frequency of SystemC scheduling is reduced.

Q6: Consider what simulation performance an ISS might give and can it ever be faster than real time?

It depends on the speed of the machine running the simulation and the speed of the target machine. If the target machine is slow, such as a low-end processor while the simulation machine is a high-end machine, ISS simulation might run faster than the actual machine. Also JIT can be used to compile some frequent in-function loops into native code to the simulation machine, which provides further speed up.

Q7: Briefly describe each of: cycle-accurate, approximately timed, loosely timed and untimed.

Cycle-accurate: The values of registers/wires are updated every hardware cycle. This model provide the same value found in real hardware simulation. RTL model is normally cycle-accurate.

Approximately timed: This model uses a number of fine-grain timing annotations for each component and delays the scheduler by each amount.

Loosely timed: This model allows threads to run ahead of the global time. This reduce the frequency of global synchronization but leads to out-of-order transactions mismatch with real hardware model.

Untimed: The model is functional arcuate but has no regard of timing.

Q8: What is the purpose and effect of the timing quantum in the loosely-timed model? Why might a transactional system exhibit different behavior as the quantum is adjusted?

The quantum is a limit of how far ahead a thread can run regarding the global time. Transactions can get out of order in a loosely-timed model where the quantum is longer than the inter-transaction time and the results of the transaction can be returned without rescheduling.

Q9: How can contention for a resource be modeled with and without actual queuing of the transaction?

This is an interesting question. Basically this is trying to model the contention delay using some virtual queue or some mathematic models.

In a cycle accurate model, normally an actual queue is needed to mimic the hardware.

In a TLM model where the accurate aggregated delay is simulated, a virtual queue or thread scheduling can be used to collect transaction delay.

Alternatively, simulation can run ahead without stall. The contention delay is then calculated according to the level of congestion. For an example, queuing theory can be used to model the delay of buffers/arbiters. In this case, only the averaged (overall) delay makes sense.

Extra. Past exam

(a) Explain what factors limit the complexity and performance of an SoC at the heart of a portable electronic device. [4 marks]

With the ever-shrinking VLSI technology, the most significant limit today is heat dissipation. So the main factors to consider include: peak power, average power, energy consumption (battery life), energy

efficiency. Also to reduce power/energy, multiprocessing, parallel computing, heterogeneous multi-core, on-chip DRAM are all available techniques.

(b) Compare and contrast the use of hardware and software to implement a compute-intensive algorithm on an SoC, such as data encryption. Include customised processors and co-processors in your analysis. [5 marks]

Hardware typically save energy because it lacks the instruction fetch/decode overhead. However, its functionality is fixed so it is only appropriate when the specification (function) is stable. To utilize hardware to accelerate software computing, several factors need to be considered:

The cost of data movement. Moving data between processor and coprocessor can be expensive both in time and delay.

Cost of frequency. Using special instruction or customized processor introduces customized hardware into the processor design. This might prolong the critical path; therefore, it comprises the processor frequency. Depending on the Ahmdal's law, this customization may not always beneficial.

(c) (i) Define the term fully-pipelined with respect to a hardware component. [2 marks]

A fully-pipelined design has a fixed processing delay and can accept a new set of operands every clock cycle.

(ii) Describe and compare three designs for a fixed or floating-point multiplier that vary in performance: one at least should be fully pipelined. [6 marks]

Three designs: single-cycle, multi-cycle non-pipelined, and nulti-cycle fully pipelined.

Single-cycle design has the minimal delay but usually the largest area or the longest latency. It can prolong the critical path.

Multi-cycle non-pipelined design does not prolong the critical path but need multiple cycle to finish every operation. Handshake based interface is normally needed. One benefit of this design is the reduced area, some common components in different cycles might be reused to reduce area.

Multi-cycle pipelined design provides the best throughput performance at the cost of extra storage and scheduling logic.

(iii) Define the term structural hazard and explain why these can affect system performance. Which of your designs from part(c)(ii) might present such a hazard and why? [3 marks]

A structural hazard is lack of physical resources which blocks operations from taking places. Examples include the lack of read ports of a register file and the lack of parallel accessing points to a bus. For the previous example, the multi-cycle non-pipelined design introduces structural hazard because when one operation is in process, no new operation can occur. However, all designs may introduce structural hazard if more than one multipliers are needed for certain cycles.