Abbreviated Quiz1 Solution:

1. Explain the utility of each of the following DSP architectural features. Demonstrate with the aid of a simple specific example (described in pseudo-code).

   - **Circular Buffers**
     - Automatic hardware mechanism does automatic pointer wrap around on a fixed size buffer when the end of the buffer is reached. The code to implement a circular buffer is replaced by the hardware mechanism.
       - e.g.
       - Assume we have allocated a FIFO of a finite size then the following code:
         ```
         write_buffer(data) {
             FIFO[write_ptr++] = data;
         }
         read_buffer(data) {
             return FIFO[read_ptr++];
         }
         ```
       In both cases, when the ptrs reach the end of the buffer, they are automatically reset to the beginning of FIFO.

   - **Zero-wait loop**
     - Automatic hardware mechanism checks if the loop exit condition has been met. The code to check loop bounds is replaced by this parallel hardware mechanism.
       - e.g.
       - without Zero-wait loop
         ```
         MOV R2, 16
         loop: LD R0, (R1)
             ADD R0, R0, R0
             ADDI R1, R1, 4
             SUBI R2, R2, 1
             BNEZ R2, loop
         ```
       - with Zero-wait loop
         ```
         MOV R2, 16
         LOOPGTZ R2, done // Assumption: next line is the loop
         loop: LD R0, (R1)
             ADD R0, R0, R0
             ADDI R1, R1, 4
         done: // code after the loop
         ```

   - **Separate instruction and data memories**
     - Allows for greater memory bandwidth by allowing an instruction and data access in the same cycle. (Allowed solutions here without examples).
       - e.g.
       - Assume a pipelined processor
         ```
         LD R0, (R1) // Reading (R1) into R0 at time t
         LD R2, (R3) // Reading the instruction at time t
         ```

   - **Auto-increment address generation unit**
     - Address generation unit automatic increments a value stored in a register. This removes the need to use an instruction to increment a register value by a constant amount.
       - e.g.
       - without Auto-increment address generation unit
         ```
         LD R0, (R1)
         ADD R0, R0, 4
         ```
       - with Auto-increment address generation unit
NOTE: The question called for a simple specific example described in pseudo-code. Several valid examples exist and there should be an example.

2. Name 5 significant differences between general purpose and embedded processors. Give for each of these one example on how this impacts the architectural design process.

- Embedded systems run a handful of applications known at design time.
  GPC must run a fully general set of applications.
    o An impact:
      Specialized instructions can be utilized for the embedded processor (i.e. MAC).
      Configuration options (memory, FUs, etc.) to tune the embedded processors.

- Embedded systems are generally not end-user programmable.
  GPC is end-user programmable
    o An impact:
      ROM could be used for the instruction memory in the embedded system.
      One does not have to worry about binary compatibility in embedded systems.

- Embedded systems have fixed and predictable run-time performance.
  GPC always want faster.
    o An impact:
      Discourage the use of unpredictable functional units such as caches.
      The clock speed only has to be fast enough to meet constraints.

- Embedded systems must have low power requirements.
  GPC are not concerned with low power.
    o An impact:
      Dynamic scheduling consumes a lot of power, so it would probably not exist in a low power embedded processor.
      Design low power modes and power management units for the embedded processor architecture.

- Embedded systems must have a low cost.
  GPC have a high cost.
    o An impact:
      Design instruction sets for high code density to save money on instruction memory.
      No expensive mechanisms such as branch prediction or superscalar architectures are used.

NOTE: These are a few examples. Other solutions were also acceptable. Based on the specific embedded system, the answers may be more or less appropriate.

3. Consider the following code segment, which is a typical representative of an application domain. Determine which architectural features will most impact the performance, and make an estimate of obtained savings (use some reasonable execution time estimates).

```c
if(a) {
    if(b) {
        if(c) {x=3;} else {x =2;}
    }
}
```
else {
    if(c) {x=2;} else {x=1;}
}

else {
    if(b) {
        if(c) {x=2; } else {x=1;}
    } else {
        if(c) {x=1;} else {x=0;}
    }
}

- Use conditional execution.
  o Estimates must take into account the savings of removing the conditional branches from the old ISA and replacing them with conditional execution in the new ISA.

- Further speedup can be achieved by executing in parallel the four conditional executions above (i.e. x=0 or x=1 or x=2 or x=3). Only one will actually assign x a value.
  o Estimates must take into account the parallelism and again the savings of removing the conditional branches from the old ISA and replacing them with conditional execution in the new ISA.

- Branch prediction can be used to perform the computations speculatively. The value of this mechanism depends on the predictability of the branching. Various prediction schemes were presented.
  o The miss rate, hit rate, and miss penalty must be taken into account when doing the estimate of obtained savings.

- Some people showed a compiler transformation on the code which identified that the calculation is really x = a + b + c assuming that a, b, and c are binary.
  o The impact would remove the branches and replace them with two data instructions.
    The estimate should take into account the savings of removing all the branches.

- Not specific to this example, but caches will decrease access time to memory.
  o The estimate is based on miss rate, hit rate, and miss penalty of the cache and environment that you assume.

NOTE: I have not shown reasonable execution time estimates, but in your answers there should be an estimate for the architectural feature you have chosen. Answers on estimates were accepted if a valid argument for the estimation exists. In the estimate criteria shown above, more sophisticated estimates including such factors as memory architecture may also be considered. It was important to talk about total estimation, not branch delay or pipeline delay. These are good, but they need to be addressed in the context of the code presented.

4. Give an example of each of the following. Be detailed and explain why.

   a. An application where a VLIW is superior in performance to a superscalar.

   VLIW hardware is simpler than superscalar hardware since the logic for detecting dependencies does not have to be included. Superscalar processors have a limited issue width due to the dependency checking complexity. Since VLIWs do not have this complex detection hardware, more room is available for wider issue, more functional units, and higher speeds. If an application has wide parallelism, the VLIW is better. Compiler methods can search a larger
window for ILP.

b. An example where a VLIW is superior in performance to a vector processor.

VLIW is better when different operations occur in parallel instead of the same operation on multiple data elements.

c. An example where a superscalar is superior in performance to a VLIW.

Superscalars have a superior performance when dependencies are better determined at runtime or if a VLIW compiler is not powerful enough to identify the same dependencies statically. It is also superior to running legacy code that would have to be transformed by binary-binary translation or code-morphing (Transmeta) on a VLIW.

d. An example where a vector processor is superior in performance to a VLIW.

Vector is better when the same operation occurs on multiple data elements (SIMD) instead of having different operations occur in parallel. This occurs quite a bit in scientific computing.

NOTE: I have given general scenarios. Examples were expected for the quiz. The question was on performance only.

5. Describe very concisely the advantages of Tomasulo’s method for dynamic scheduling over the CDC 6600 Scoreboard. Are there any disadvantages relative to the scoreboard?

Advantages
- Register renaming via reservation stations eliminates WAR and WAW hazards/stalls. Registers do not need to always be retrieved from the register file. They can arrive on the CDB.
- CDB distributes completing results to all reservation stations that require the operand and the register file. This allows multiple instructions to begin execution simultaneously.
- Unlike the CDC 6600 Scoreboard, execution of multiple iterations of the same loop can occur. In the CDC 6600, the execution is limited to a basic block.

Disadvantages
- Complexity of the distributed hardware of the reservation stations and CDB.
- Only one instruction can be completed at a time since there is only one CDB.

NOTE: There are many more advantages that I saw in your answers that were acceptable. For the disadvantages, I expected that you mention the two listed above.

6. Enumerate 4 reasons why reconfigurable computing can lead to lower energy solutions for programmable embedded applications.

- Customization of hardware to the problem allows a problem to be completed faster. Specialized hardware and parallel data paths result in high performance implementations of certain algorithms. If the problem completes faster at an equivalent power usage, then there is an energy savings.
- Bit level parallelism can be exploited. For example a 6 bit add would use a 32-bit adder on a GPC. This is an inefficiency that consumes power for the extra 26 bits.
Hardware can be reconfigured so that communicating elements are close to each other. This results in shorter wire lengths to be driven. Shorter wire lengths have lower resistance so the energy usage is less.

Specialized ISAs allow for FUs to match application requirements. The replacement of a sequence of instructions by a new more powerful instruction which takes fewer clock cycles will decrease energy use since fewer clocks are required to perform the computation.

Allocating LUTs for function and storage allows for efficient retiming. Efficient retiming lowers the number of clock cycles; this in turn lowers energy.

Reconfigurable interconnect can be configured so that only the interconnect that is in use is connected. This means that unnecessary data paths can be turned off to conserve power.

Determine also 2 conditions that these application programs have to meet to exploit these opportunities.

- The portion of the design targeted for reconfigurable computing must fit on the device.
- Power used to reconfigure the device must not exceed the benefit of specialization.
- There must be a limited number of machines that are swapped in and out for the application.
- The machines implemented for the application should benefit from spatial computation.
- Generally, feed-forward kernels opposed to feed-back kernels are more exploitable in reconfigurable fabrics due to the long feedback wire lengths.

NOTE: Other answers were accepted. Some answers addressed the energy question, but provided solutions that were not specific to reconfigurable computing.

7. Consider the following configurable network topologies bus, crossbar, mesh, and binary tree. Assume that we are trying to connect \( n \) elements to each other. Rank order each of them with respect to the following properties (best first):

- overall bandwidth: based on the number of connections
  - crossbar
  - mesh
  - binary tree
  - bus

  The bandwidth is based on the number of connections. Since a crossbar has a direct connection between each node, it has the highest bandwidth. Since the bus must be shared by all communications between any two nodes, it has the lowest bandwidth. A binary tree is essentially a mesh with some links removed so it ranks below the mesh. The binary tree has more links than a bus.

- maximum latency
  - mesh
  - binary tree
  - crossbar
  - bus

  This is unloaded maximum latency. There were a number of answers for this question. Some people assumed complicated RC delay models. This was fine if you explained it and the explanation justified the answer. My answer is based on a simple count of the number of hops from nodes that a communication would have to take. The crossbar and bus could be interchanged reasonably using this model. I put the crossbar before the bus because it is more complicated and therefore potentially slower. Some had the order backwards, but I accepted it if it was clear what you meant.
- area (or cost)
  - bus
  - binary tree
  - mesh
  - crossbar

  This is related to the arguments made for the overall bandwidth (i.e. it is reversed). If the width of the bus and \( n \) is the number of nodes, then the area of the topologies have the following orders. The area of the bus is proportional to \( O(wn) \). The area of a mesh is \( O(w^2n) \). A binary tree is essentially a mesh with links removed so it has lower area than a mesh. The area of a crossbar is proportional to \( O((wn)^2) \). These come from Lecture 13: (Re)configurable Computing.

- energy dissipation per transfer
  - bus
  - crossbar
  - mesh
  - tree

  This is the same as for maximum latency since hops are equivalent to energy dissipation per transfer if you consider clock ticks dominating power.

Use simple models and assume that connection requirements are equally spread amongst all of the elements (i.e. there is no inherent locality in the connections).

If your model was sound and your answers followed, then I gave you credit.

NOTE: Although I have shown a solution, if people use and explain different models and justify their answers, then they received credit. If no explanation was explained then I graded according to my solution. I had no way of understanding one’s models if they were not stated since there were potentially different models. Points were taken for simply writing an answer down without justification since I had no idea which model one was using. Also one should include references to the models that were used since just writing a number down is not good enough.

NOTE: I decided to take half off if no models were stated. The reason is that I myself had to rat my brain to figure out the answers and I could formulate different ones by easily changing the parameters of my models. If you include simply hops, then you would have my answers, but if you have my answers and don’t state your model, then I still took off credit because I don’t know how you were thinking. These points are not negotiable since I had no other fair way to grade this problem. In the future, state your assumptions.