EEE4120F Final Exam

May 2019

MEMO

Question 1.1 [10]

1. Would you classify your device as a high performance embedded computing system? Explain why. [3]
2. Write 2 requirements for each of the *sensing* and *processing* subsystems of the system. Use proper requirements language (use complete sentences, and words like 'should' and 'will' to express yourself confidently and clearly.) [4]
3. You have two main architectural options for the implementation of the translator system: distributed (i.e., networked) or non-distributed (i.e., stand-alone). Which one of these two architectural approaches would you use and why? [3]

Sample soln.:

1. Yes.
   -Tight form factor requirements (it's a wearable device)
   -Real-time performance
   -High performance computing technologies will be incorporated to achieve real-time performance
2. Requirements:

Module	**Requirements**
Sensing	R1: Should input user speech and convert it to a digital file R2: Should
Processing	R1: Should convert the digitised speech to text R2: Should convert the text to equivalent speech in another language R3: Should perform speech to speech translation in real-time

3. Distributed.
   -20 languages is a lot, each language has its own dictionary and unique grammar that should be understood and analysed through complex comparisons to perform speech-to-speech conversion.
   -It is virtually impractical to house this processing on a stand-alone wearable platform and also provide real-time performance.
   -Therefore, in order to achieve form-factor and real-time processing constraints, house the processing intensive functionality (e.g speech recognition, machine translation and speech synthesis) inside a high performance computing server
   -This leaves minimal processing on the device and thus can be built to achieve the small form-factor required for it to be wearable

Question 1.4 [10]

1. A programmer has developed a C implementation of the algorithm which was found to be extremely slow when tested on a 3.06 GHz P4 platform. In order to improve response time, the programmer transferred the code to a platform that uses the latest multi-core ARM Cortex-A72 chip. However, a degradation in performance was observed instead. Explain the possible reasons for this behavior. [3]
2. Analyse the worst-case execution time of the Hough Transform. You may use the Big-Oh notation for your analysis. [5]
3. Consider an embedded application of the algorithm for lane detection that uses a 4K video camera with a chip that captures the input video stream with 4096 H × 2160 V pixels resolution at 60 fps and 16 bits. What is the minimum size in bytes of the frame buffer required to store a frame? [2]

Sample soln:

1. -Probably, the C implementation was not parallelised.
   -So, the hardware is fast but the algorithm has not been efficiently mapped to the parallel hardware.
   -Sub-linear speed-up (performance degradation when more processing nodes are added) can happen when the algorithm data is too large for the parallel architecture's cache.
2. -$O(N^4)$
   In computing the conventional Hough Transform of an n × n image into an m × m × m Hough space we need to process $O(n^2)$ pixels and generate $O(m^2)$ votes for each. Typically m ≈ n, so the conventional Hough transform is $O(n^4)$.
3. 849346560 bytes (= 4096 x 2160 x 3 x 16 x 2)

Question 1.3 [12]

1. Given that the pavement damage model problem above requires the use of hundreds of cores for one day damage model calculation, long double floating point arithmetic and fused multiply/add operations, advanced vector processing, and shared memory, what kind of compute machine would you use and why? [2]

Sol: One of the following:

- Use a CPU cluster with vector processing capabilities, preferably Intel Xeon Processors with AVX (Advanced Vector eXtentions)
- Create a cluster as CPUs have limited number of cores.

The sequential damage model algorithm takes 30 seconds to complete during execution. In order to speedup the computation, the developer designed a compute cluster with 150 Intel Xeon Gold processors with 16 cores each, each core running at 3.6GHz clock frequency.

2. Given that each instance of the pavement/road damage calculation requires 16 cores (i.e. Common industry practice of using processor physical cores not processor threads) and the road damage calculating function for one load is named ***road_damage_calc(Input &in, int mpi_rank)***, discuss how you would use a combination of OpenMP and MPI code in order to process 150 load configurations. [4]

Sol: One of the following:

- Use OpenMP for internal parallelism inside the road_damage_calc(...) function
- Use openMP to schedule each instance of the function to run in different mpi ranks
  Or
- Each MPI rank should have 16 cores, so the best way is to allocate 16-core processor for each load type, thus 150 nodes each with 16-cores, then rank by processor sockets or nodes, and finally OpenMP in each rank/node to parallelize
  Or
- Use MPI to scatter load types damage model calculations, and use OpenMP to parallelize load-points calculations inside the road_damage_calc(...) function

3. Given that each of the load damage model (i.e. ***road_damage_calc(Input &in, int mpi_rank)*** function) takes 8 seconds when running in parallel in the 16-cores machine recommended in 2. above, how much efficiency is achieved? [4]

Sol:
S = Ts/Tp = 30s/8s = 3.7 [2 marks]
Efficiency = S/p = 3.7/16 = 0.2 [2 marks]

4. The damage model of a pavement/road in a given period in years is the sum of every monthly damage computed and accumulated over a given number of years. The damage model of each month is the input to the damage model of the next month until the last month of the last year (i.e. Damage value of January is the initial value of damage value for February, ...). Given that the damage model of one month is equivalent to the daily sum of damages for 150 load configurations (i.e. damages are computed once every month), What parallel programming communication patterns can you use to implement the parallel pattern for calculating this pavement damage model in a compute cluster of Intel Xeon Gold Processors in b. above? [2]

Sol: One of the following

- **Scatter and Reduction (similar to Map-Reduce pattern) [2]**
  or
- **Scatter road_damage_calc(...) function in multiple processors and Wait for results from 150 nodes, and reduce/sum to one value. [2]**
  or
- **Map-reduce parallel pattern [2]**

Question 1.2 [12]

1. Engineers often confuse terms when describing the performance of a communications network. Explain the differences between Latency, Bandwidth and Baud Rate. [6]
2. Effective bandwidth, which takes total latency into account, is a more realistic metric than just the raw bandwidth of a communications channel. Calculate the effective bandwidth of a particular channel with the following parameters: [6]
    - Distance = 1km
    - Raw bandwidth = 12Mbits/s
    - Message site = 10 000 Bytes
    - Sending overhead = 100us
    - Receiving overhead = 150us

Solution:

1)

- Latency: Time it takes to send a minimal length message from one task to another. Expressed in us.
- Bandwidth: The amount of data that can be sent per unit of time. Expressed in Mbps / Gbps.
- Baud rate: The amount of symbols can be sent per unit of time. Expressed in Bdps.

2)

$$Transmission\ time = \frac{80000bits}{12Mbits/s} = 6666.67us$$

$$Time\ of\ flight = \frac{1000m}{3e8m/s} = 3.33us$$

$$Total\ latency = sending\ overhead + transmission\ time + time\ of\ flight + receiver\ overhead$$

$$Total\ latency = 100us + 6666.67us + 3.33us + 150us = 6920us$$

$$Effective\ bandwidth = \frac{message\ size}{total\ latency}$$

$$Effective\ bandwidth = \frac{80000bits}{6920us} = 11.56Mbits/s$$

Question 2.2.1 [5]

In the OpenMP code segment shown below, how many iterations are executed if four threads execute this program?

```
#pragma omp parallel private(i)
for (int i = 0; i < 100; i++) {
     a[i] = i;
}
```

A. 100
B. 50
C. 25
D. 0
E. None of the above

Soln: C

25, the processing is divided across the four threads

Question 2.2.2 [5]

Give that energy the dissipation for the load/save, add/sub/cmp/branch/mov, multiply/divide instructions are 40nJ and 20nJ and 60nJ respectively. The total energy dissipation for the following assembly program is:

```
  MOV B, #0x0000; set B to constant value
LOOP: LD A, [B]; read memory
  MUL A, A, C; A = A * C
  ST A, [B];save changed value of A back to [B]
  ADD B, #1;B = B + 1
  CMD B, #0xDDFF; compare B to 0xDDFF
  JNE LOOP
```

A. 5242800
B. 3932120
C. 3932100
D. 13107020
E. 11366220

Soln: E

	inst. energy	**inst. count out loop**	**inst. count in loop**	**loop max**	**total inst. count**	**total inst energy**
L/S	40	0	2	56831	113662	4546480
A/S/CMP/B/MOV	20	1	3	56831	170493	3409880
MULT/DIV	60	0	1	56831	56831	3409860
					340986	**11366220**

Question 2.2.3 [5]

Which one of the following statements is correct:

a) The Cell processor is considered to be a homogeneous multicore processor because it has 1 power processing element and 8 synergistic processing elements.
b) A softcore processor is used for compiling software in a personal computer, whereas a hardcore processor is synthesized in FPGA hardware.
c) It is recommended to use blocking and nonblocking within the same always block to improve FPGA performance.
d) Fine-grained parallel tasks require a great deal of intercommunication relative to the amount of computing they do.
e) Wishbone, Altera's proprietary bus system, supports shared bus topologies and bus widths of different sizes.

Soln:
d) Fine-grained parallel tasks require a great deal of intercommunication relative to the amount of computing they do.

Question 2.2.4 [5]

Which of the following is true about Cache coherence
a) It is when when one processor writes a location in shared variable, all other processors are hindered from access it at the same time
b) It is when one processor writes a location in shared variable, all other processors are updated
c) It is when one processor writes a location in a shared memory and all other processors are updated
d) It is implemented at the software level when using reconfigurable embedded processor.
e) It is only used in small memory capacity requirement

Sol: c. is the most accurate answers. [5]

Question 2.2.5 [5]

A compute cluster is often build using three or more compute nodes with high-speed processors such as the Intel Xeon Gold processors with 16 cores at 3.6GHz clock frequencies. Which one of the following is the most accurate reason why a HPC developer for a small company would opt for an NVidia Titan Xp GPU even though it has a clock speed of 1.5GHz?

a) It is too complicated to parallel program compute clusters
b) Communication latencies in compute clusters are too high and interconnect suppliers have not made a breakthrough in a while
c) Compute clusters are very costly, can only be afforded by government funding and huge private companies
d) This GPU has thousands of CUDA cores, when the parallel code is efficiently optimized to run on them, they can outperform high-speed processors like the Xeon Gold
e) GPUs are very fast regardless of their clock speed

Sol: d. Is the most accurate answer. [5]

Question 2.2.6 [5]

4) The algorithm complexity of algorithm X is given by 5log2(N), for N=number of samples = 2048. Given that 1000 instances of this algorithms are executed in k=1000 processors in parallel, with a uniform communication latency of 10us between the master processor and all these 1000 processors. How many operations per second (e.g. In GFLOPS) can be achieved by these 1000 processors machine if a single instance of algorithm X takes 0.1ms to execute and consists of adder and multiplier floating-point operations only.

a. 500 GFLOPS
b. 100 MFLOPS
c. 1,000,000,000 GFLOPS
d. 1 GFLOPS
e. 0.5 GFLOPS

sol:
FLOPS = 2*5*log2(2048)*1000 flop /110us = 2*5*log2(2048)*1000 (flop)/0.00011(s) = 2*500000000 FLOPS = 1 GFLOPS, for 110us = comm latency + execution time, and flop = X complexity * 1000 instances

Question 2.2.1 (2)

To benefit from a multiprocessor, an application must be concurrent.

Soln:

False.

Question 2.2.2 (2)

Like SMPs, message-passing computers rely on locks for synchronization.

Soln:

False.

Question 2.2.3 (2)

Three main constraints for embedded computers are power, size, and weight.
Soln: True

Question 2.2.4 (2)

There is no such thing as a 64-bit embedded processor.
Soln: False

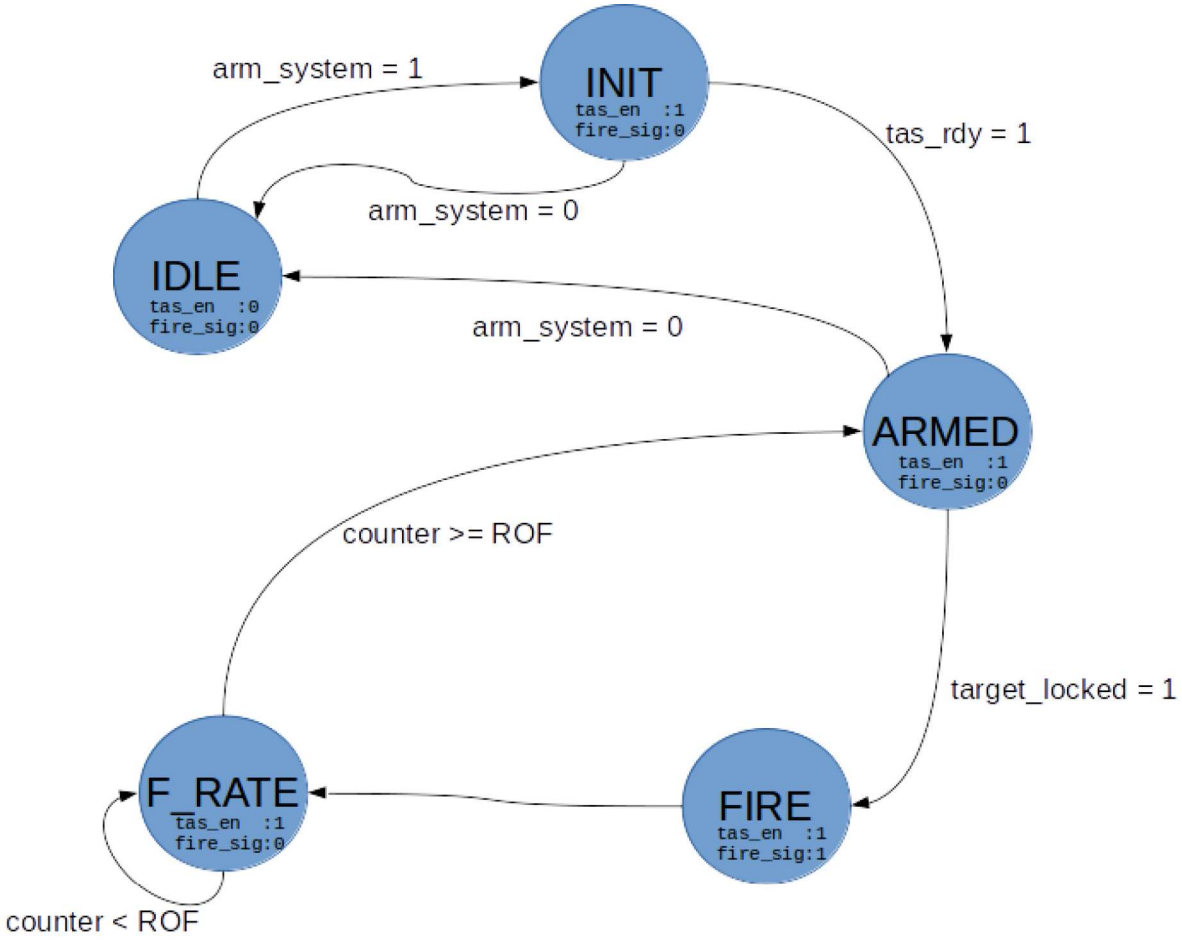
Question 3.1 [12]

The explanation in the previous paragraph indicates that a finite state machine would be the ideal way to implement Rick's Controller. A state diagram is a good place to start when designing FSMs.

Draw a state diagram for Rick's Controller.

Sample Soln

Student should have around 4 - 6 states for the controller.



Question 3.2 [14]

Using your state diagram you drew in Q3.1 as reference, you will now help Rick with his Verilog.

Implement a one-hot encoded finite state machine for Rick's Controller.

Here is a starting point for your implementation:

```
module controller (clk_100MHz, rst, arm_sys, fire_rate, tas_rdy,
                   target_lock, tas_en, fire_sig);
  // inputs
  input clk_100MHz, rst, arm_sys, tas_rdy, target_lock;
  input [9:0] fire_rate;

  // outputs
  output reg tas_en, fire_sig;

  // internal register(s), parameter(s) and FSM code
  // *---answer for Q3.2---*

endmodule
```

Soln

```
// internal register(s), parameter(s) and FSM code
      // define state register and state types
      // 5 states = 5 bit state register for one-hot encoding
      parameter [4:0] IDLE   = 5'b00001;
      parameter [4:0] INIT   = 5'b00010;
      parameter [4:0] ARMED  = 5'b00100;
      parameter [4:0] FIRE   = 5'b01000;
      parameter [4:0] HALT   = 5'b10000;

      reg [4:0] state;

      // define internal reg for rate of fire
      reg [31:0] rof_count;




      always @(posedge clk_100MHz)
```

```
begin
if (rst) begin
      state <= IDLE;
      tas_en   <= 1'b0;
      fire_sig <= 1'b0;
      rof_count<= 0;
      end
else begin
      case(state)
            IDLE:   begin
                  if (arm_sys) begin
                        state  <= INIT;
                        end
                  end
            INIT:   begin
                  tas_en   <= 1'b1;
                  if (tas_rdy) begin
                        state <= ARMED;
                        end
                  end
            ARMED:  begin
                  if (!arm_sys) begin
                        state      <= IDLE;
                        rof_count <= 0;
                        end
                  if (target_lock) begin
                        state <= FIRE;
                        end
                  end
            FIRE:   begin
                  fire_sig <= 1'b1;
                  state <= HALT;
                  end
            HALT:   begin
                  fire_sig <= 1'b0;
                  if (rof_count >= (fire_rate*1000)-2) begin
                        state      <= ARMED;
                        rof_count <= 0;
                        end
                  else
                        rof_count <= rof_count + 1;
                  end
      endcase
  end
end
```

Question 3.3 [6]

Morty explains that his Override module should allow the fire_sig to reach the cannon only if the laser cannon is not overheating and that the cannon enable switch is on. He says that for the quickest response time, the module should be purely combinational.

Write the verilog code for Morty's Override module.

Soln

```
module override (fire_sig, cannon_en, overheat, fire_cannon);
  // inputs
  input fire_sig, cannon_en, overheat;

  // outputs
  output reg fire_cannon;

  always @(*)
  begin
    if (fire_sig == 1'b1 && cannon_en == 1'b1 && overheat == 1'b0)
      fire_cannon = 1'b1;
    else
      fire_cannon = 0'b1;
  end
endmodule;
```

another way:

```
module override (fire_sig, cannon_en, overheat, fire_cannon);
  // inputs
  input fire_sig, cannon_en, overheat;
  // outputs
  output fire_cannon;

  assign fire_cannon = fire_sig & cannon_en & (~overheat)
endmodule;
```

Question 3.4 [6]

Whilst you were coding, Morty was reading up about FPGAs and how a logic function is mapped to a lookup table (LUT) consisting of memory and a multiplexer.

Draw a LUT that implements the combinational logic used in Morty's Override module.
Hint: start with a truth table.

Soln

X = fire_sig
Y = cannon_en
Z = overheat
F = fire_cannon

$$F = XY\overline{Z}$$

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0