University of Pennsylvania
Department of Electrical and System Engineering
System-on-a-Chip Architecture

ESE532, Fall 2018 — Final — Friday, December 14

- Exam ends at 11:00AM; begin as instructed (target 9:00AM).
  Do **not** open exam until instructed.
- Problems weighted as shown.
- Calculators allowed.
- Closed book = No text or notes allowed.
- Show work for partial credit consideration.
- Unless otherwise noted, answers to two significant figures are sufficient.
- Sign Code of Academic Integrity statement (see last page for code).

____

I certify that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this exam.

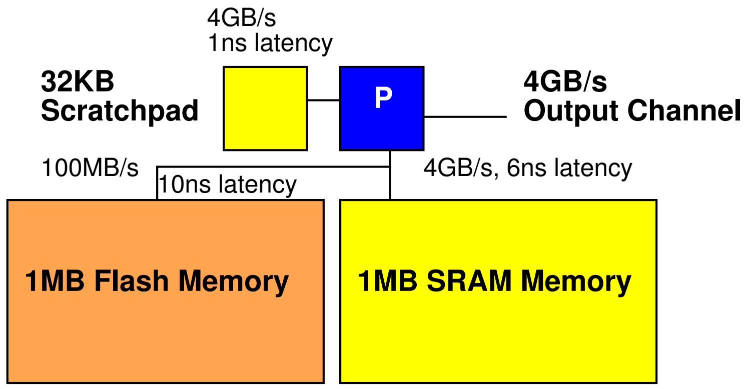
Name:

1a	1bc	2a	2b	2c	3	4	5	6a	6b	6c
5	5	2	2	8	9	10	9	3	3	4

7a	7b	7cd	7e	8a	8b	8c	8d	Total
6	2	6	8	6	2	6	4	100

```
// You will be determining a value for FREQBYTES
#define WINDOW 1024
#define MAXBITLEN 11
#define LOG_MAXBITLEN 4
#define MAX_FREQS 255
#define MASKLOOKUP ((1<<MAXBITLEN)-1)
#define MASKLEN ((1<<LOG_MAXBITLEN)-1)
#define AMPLEN 14
#define FREQLEN 14
#define MASKAMP ((1<<AMPLEN)-1)
#define MASKFREQ ((1<<FREQLEN)-1)
uint8_t in[FREQBYTES];
uint32_t fa[FREQS];
uint32_t lookup[1<<MAXBITLEN];
uint16_t s[MAX_FREQS][WINDOW];
while(1) { // Outer while loop
    uint32_t ts[WINDOW];
    for (j=0;j<WINDOW;j++) ts[j]=0; // Loop A
    uint8_t freqs=read_flash_byte(); // max rate 100MB/s
    for(int i=0;i<FREQBYTES;i++) // Loop B
        in[i]=read_flash_byte();
    uint11_t top11=((int *)in)[0]>>21;
    uint11_t next11=(((int *)in)[0]>>10)&MASKLOOKUP;
    int next11bitpos=11;
    for(i=0;i<freqs;i++) { // freqs<MAX_FREQS // Loop C
      uint32_t res=lookup[top11];
      uint32_t tfa=res>>LOG_MAXBITLEN; fa[i]=tfa;
      uint4_t len=MASKLEN & res;
      uint32_t t1=(top11<<len); uint4_t t2=(MAXBITLEN-len); uint32_t t3=(next11>>t2);
      top11= t1|t3;
      next11bitpos+=len;
      uint32_t bytepos=next11bitpos>>3; uint3_t bitoffset=next11bitpos%8;
      uint32_t wordval=(*((int *)(&in[bytepos]))); // treat as 1 cycle
      uint4_t t4=(21-bitoffset); uint32_t t5=(wordval>>t4);
      next11=MASKLOOKUP & t5;
      }
   for (i=0;i<freqs;i++) { // Loop D
       uint16_t freq=(fa[i]>>AMPLEN) & MASKFREQ;
       uint16_t amp=fa[i] & MASKAMP;
       for (j=0;j<WINDOW;j++) // Loop E
           ts[j]+=s[freq][j]*amp;
       }
   for (j=0;j<WINDOW;j++) // Loop F
       output(ts[j]); // max rate 4GB/s
}
```

We start with a baseline, single processor system as shown.



- For simplicity throughout, we will treat non-memory indexing adds (subtracts count as adds), shifts, mod-by-power-of-two, ORs, ANDs, and multplies as the only compute operations. We'll assume the other operations take negligible time or can be run in parallel (ILP) with the adds, multiplies, and memory operations. (Some consequences: You may ignore loop and conditional overheads in processor runtime estimates; you may ignore computations in array indecies.)
- Assume all additions are associative.
- Baseline processor can execute one compute operation (above) per cycle and runs at 1 GHz.
- Constant expressions (like 1 << 8) are evaluated by the compiler and take no time to compute at runtime.
- Maximum data rate for reading from flash is 100MB/s. Latency of read is 10 ns.
- The output port used by `output()` can transfer data at 4GB/s (one 32b word per cycle at 1 GHz).
- Baseline processor has a 32KB local scratchpad memory.
- in[], fa[], ts[], and lookup[] fit in the local scratchpad memory close to the processor and can be read or written in a single cycle.
- For the baseline processor, s[] lives in the large (1MB) memory and requires 6 cycles to access.
- lookup[] and s[] are prepopulated with content before entering the while loop (not shown).
- Assume adds and multiplies take 1 ns when implemented in hardware accelerator, so fully pipelined accelerators also run at 1 GHz.

1. For sequential evaluation and assuming FREQBYTES is 256.

   (a) Worst-case cycles to compute one iteration of the outer while loop?
   (show cycles per loop for partial credit consideration.)

   (b) Which outer loop is the bottleneck?
   Circle One:

A	B	C	D	F

   (c) What is the Amdhal's Law maximum speedup for accelerating the identified loop?

2. Loop C

   (a) How many memory operations does one instance of the loop perform?

   (b) How many compute operations (of the set identified) does the loop perform?

   (c) Assuming unlimited compute operators and memory ports, what is the minimum achievable Initiation Interval (II) for this loop?
   Draw dataflow graph and identify any data-dependent loops for full credit.

3. Data Parallel: Classify Loops C, D, and E:

Loop	Data Parallel?	Associative Reduce?	Must be Sequential?
C			
D			
E			

4. What is the latency bound for executing Loops C and D (from the beginning of C to the end of D)?

   - assume memories of unbounded width (no bandwidth limits)
   - respect latencies for memory access

5. Data Streaming: How big (minimum size) does the buffer need to be between the identified loops in order to allow the loops to profitably execute concurrently.
   (Hint: Based on data dependencies, under what scenarios and granularity can the identified loops act as a producer-consumer pair in a pipeline.)

   Explain size choices for partial credit consideration.

Loop Pair	Size (bytes)
B→C	
C→D	
D→F	

6. Consider trying to achieve a real-time rate of one window output per cycle (equivalently, the II of the outer while loop is WINDOW or 1024 cycles).
   Assume you exploit data streaming between loops so they can run concurrently.

   (a) Given that Flash memory has a maximum throughput of 100 MB/s, what is the maximum possible value for FREQBYTES?

   (b) Based on your II identified in Problem 2c, what is the maximum value for `freqs` in order fo meet this real-time throughput goal?

   (c) What II do you need to achieve for Loop D to meet this real-time throughput goal?

7. Define the composition of a custom VLIW datapath for loop C that can achieve the identified II in Problem 2c.
For full credit, minimize area of your implementation.
Assume:

- Design includes at least one write port to a scratchpad memory containing fa[] and one read port to a scratchpad memory containing in[]
- Assume a crossbar interconnect between operator (and memory port) outputs and operator (and memory address, data) inputs.

(a) How many operators of each type? Give both Resource Bound (RB) and number for which you can schedule.

			Number	
Operator	Inputs	Outputs	**RB**	**Schedule**
shifters	2	1		
ALU (includes |, &, +, - , %-by-powers-of-2)	2	1		
scratchpad memory banks	2	1		
ports to memory containing in[]	1	1		
ports to memory containing fa[]	1	0		
above error, should be	2	0		
branch units	1	0		

(b) How are the scratchpad memory banks used?

(c) Crossbar Inputs and Outputs for your design (final column, the one you can schedule)?

Inputs	
Outputs	

(d) Estimate the area for your design using the following costs.

- shifters: 1024
- ALU (includes |, &, +, - , %-by-powers-of-2): 32
- Scratchpad memory banks of depth d: $60(d+6)$
- ports to memory containing in[]: 200
- ports to memory containing fa[]: 200
- branch unit: 100
- crossbar: $128 \times \mathit{Inputs} \times \mathit{Outputs} + 2400 \times \mathit{Outputs}$
(Each crossbar output includes a 4 word memory acting as a small register file for input to the associated operator or memory.)

(e) Provide a schedule:

Operator→	fa[] write	in[] read	Label with your selected operators										
Cycle													
0													
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													

Label cells with the variable assigned by the operation (or array entry written).

(Note extra schedules at end. May want to use as scratch while exploring schedules and put final here.)

8. Considering a custom hardware accelerator implementation where you are designing both the compute operators and the associated memory architecture, how would you use loop unrolling and array partitioning on Loop D to achieve the identified II in Problem 6c, while minimizing area?
Use the following area model and assume s[], ts[], and fa[] are part of this loop module:

- n-bit counters: n
- 32b adder: 32
- 16×16 multiplier: 256
- Single-port, 32b-wide memory holding d words: $38(d + 6)$
- Double-port, 32b-wide memory holding d words: $60(d + 6)$

(a) Unrolling for each loop (D, E)?

Loop	Unroll Factor
D	
E	

(b) For the unrolling, how many multipliers and adders?

Multipliers	
Adders	

(c) Array partitioning for each array (s[], ts[], fa[])?
(each memory block has either 1 or 2 ports)

Array	Array Partition	Ports (select one)	Words/partition
s[]		1 2	
ts[]		1 2	
fa[]		1 2	

(d) Identify the component(s) that consumes most (>80%) of the area?
(you don't necessarily need to compute the area to fine precision, but you need to estimate where area is going well enough to answer the question above.)

This page left almost blank for pagination. You may use for answers and computations.

Extra schedule (in case you need it for trying schedules out, or if you need to put your answer here; be clear which schedule we should grade.)

Operator→	fa[] write	in[] read	Label with your selected operators										
Cycle													
0													
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													

Label cells with the variable assigned by the operation (or array entry written).

Extra schedule (in case you need it for trying schedules out, or if you need to put your answer here; be clear which schedule we should grade.)

Operator→	fa[] write	in[] read	Label with your selected operators										
Cycle													
0													
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													

Label cells with the variable assigned by the operation (or array entry written).

Code of Academic Integrity

Since the University is an academic community, its fundamental purpose is the pursuit of knowledge. Essential to the success of this educational mission is a commitment to the principles of academic integrity. Every member of the University community is responsible for upholding the highest standards of honesty at all times. Students, as members of the community, are also responsible for adhering to the principles and spirit of the following Code of Academic Integrity.*

Academic Dishonesty Definitions

Activities that have the effect or intention of interfering with education, pursuit of knowledge, or fair evaluation of a students performance are prohibited. Examples of such activities include but are not limited to the following definitions:

A. Cheating Using or attempting to use unauthorized assistance, material, or study aids in examinations or other academic work or preventing, or attempting to prevent, another from using authorized assistance, material, or study aids. Example: using a cheat sheet in a quiz or exam, altering a graded exam and resubmitting it for a better grade, etc.

B. Plagiarism Using the ideas, data, or language of another without specific or proper acknowledgment. Example: copying another persons paper, article, or computer work and submitting it for an assignment, cloning someone elses ideas without attribution, failing to use quotation marks where appropriate, etc.

C. Fabrication Submitting contrived or altered information in any academic exercise. Example: making up data for an experiment, fudging data, citing nonexistent articles, contriving sources, etc.

D. Multiple Submissions Multiple submissions: submitting, without prior permission, any work submitted to fulfill another academic requirement.

E. Misrepresentation of academic records Misrepresentation of academic records: misrepresenting or tampering with or attempting to tamper with any portion of a students transcripts or academic record, either before or after coming to the University of Pennsylvania. Example: forging a change of grade slip, tampering with computer records, falsifying academic information on ones resume, etc.

F. Facilitating Academic Dishonesty Knowingly helping or attempting to help another violate any provision of the Code. Example: working together on a take-home exam, etc.

G. Unfair Advantage Attempting to gain unauthorized advantage over fellow students in an academic exercise. Example: gaining or providing unauthorized access to examination materials, obstructing or interfering with another students efforts in an academic exercise, lying about a need for an extension for an exam or paper, continuing to write even when time is up during an exam, destroying or keeping library materials for ones own use., etc.

* If a student is unsure whether his action(s) constitute a violation of the Code of Academic Integrity, then it is that students responsibility to consult with the instructor to clarify any ambiguities.