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

ESE532, Spring 2017 — Analysis Milestone — Monday, March 20

Analysis Milestone

Due: Friday, March 24, 5:00PM

Group: Selecting your partner and profiling are group tasks for this milestone. You may discuss and refine your model as a project team.

Individual: Exercise and model writeup are individual tasks.

1. Report the partner you have agreed to work with for the project duration.
2. Requirements to achieve real-time for the 1080p30 task identified in the project assignment.
   (a) Reprofile the design as necessary to collect runtime for the entire 1080p30 MPEG encoding task as specificed in the project goal for the ARM at the `-O3` optimization level.
      - this may require additional profiling to identify requirements beyond `fullsearch`

   (b) Identify the computational components that must be accelerated to meet the real-time task and the acceleration required
   (c) Identify the set of components that could continue to run together on a single ARM core when you target the 1080p30 real-time task
   (d) Identify bandwidth constraints on the design to meet the real-time target (e.g. DRAM, can you afford to read frame data from DRAM multiple times?, how many?, can you afford to move data between functions through DRAM?)
   (e) Develop a refined and expanded execution time model for the next two dominate compute components in MPEG encode task beyond the `fullsearch` that you developed in HW7.
      - this may start with a reconcilation of the model you and your partner developed for HW7

   Note: for this milestone we specifically ask about the 1080p30 real-time task. For your overall project work, you should make sure you have a parameterized model that will also be useful for your optimization of the benchmark videos on the Zynq, and you should make sure you can answer similar questions for yourself relevant to the Zynq XC7Z020 acceleration task. Similarly, in part (e) we ask that you expand your detail model to 3 components. To fully achieve the 1080p30 real-time task, you will likely need to accelerate additional components, and you should plan on refining your model further as you address them.

3. VLIW exercise
   `fdct` contains two nearly identical loop nests (first one shown and will be our focus):

```
for (i=0; i<8; i++)
  for (j=0; j<8; j++)
  {
    s = 0.0;
    for (k=0; k<8; k++)
      s += c[j][k] * block[8*i+k];
    tmp[8*i+j] = s;
  }
```

   (a) Look at the ARM compiled assembly for this loop: how many instructions must be executed per inner (`k`) loop body?

   (b) Assuming the multiply operation takes 3 cycles and all other instructions one, how many cycles does it require to execute each iteration of the `k` loop?

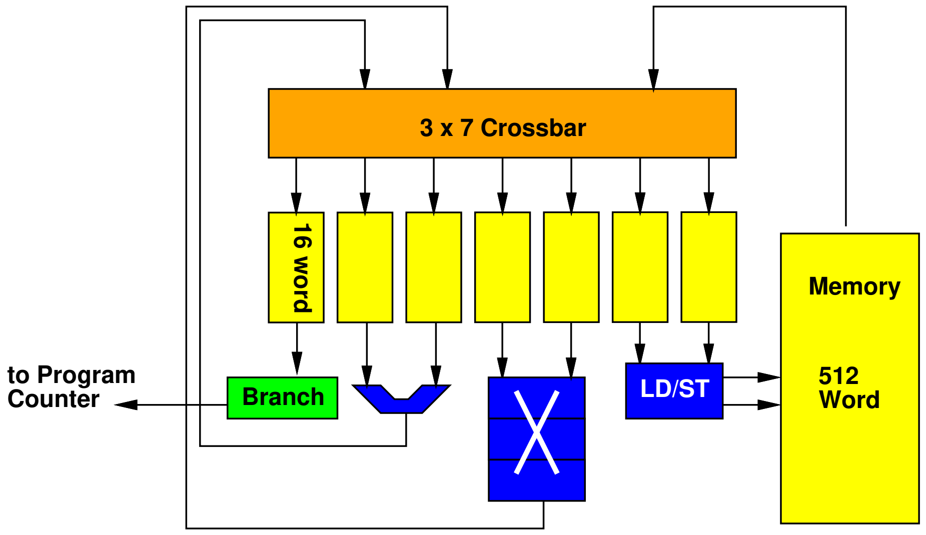
   (c) For the VLIW instance shown (Fig. 1) with one multiplier (pipelined with 3 cycles of delay), one ALU, one LD/ST unit, and one branch unit (`c`, `block`, and `tmp` live in 512 word memory).

   - What is the resource bound lower bound cycle count for computing the `k` loop body?
   - Schedule the DCT `k` loop body on the VLIW instance shown, accounting for multiplier delay and interconnect limitations; you may software pipeline the loop. (sample schedule matrix shown at end; color code entries by loop instance.)
   - Explain any assumptions you make about context from the outer loops (e.g. values computed outside of the `k` loop and stored in registers such as storing `i` in slot 3 of the ALU 2 input bank)

   (d) Assuming the area model shown on the following page, design a customized VLIW that is no more than twice the size of the design above, schedule for the revised design, and report the speedup achievable. Full solution includes:

   - original area estimate
   - customized design provisioning and area estimate
   - scheduled loop mapping on customized design

   (e) Based on resource bounds alone (you do not need to schedule), what operator distribution would you need to completely unroll the `k` loop to achieve an II of 1 for the `j` loop?



Assume:

- Branch unit effects instruction performed on following cycle
- ALU, LD/ST result is written into the selected memory at the end of the cycle (single cycle through RF read, operation, crossbar, to write of RF)
- Left (1) input to LD/ST is value (for a write); right (2) input is address
- Muliplier operation issued in cycle i uses the crossbar output on and writes into register file at end of cycle $i+2$. Order of inputs to multiplication does not matter.

Figure 1: Base VLIW Design

Area Model:

- Single Port Memory: 1 × words
- Multiport Memory: (1+ports) × words
- Crossbar: 4×inputs×outputs
- ALU: 10
- Multiplier: 100
- LD/ST: 50
- Branch Unit: limit of 1, leave out of area model

	Operators				Crossbar Outputs						
Cycle	ALU	Multiply	LD/ST	Branch	Br	ALU 1	ALU 2	Mpy 1	Mpy 2	LD 1	LD 2
0											
1											
2											

For Operators, indicate the operation and the input slots. For Crossbar Outputs, indicate the Operator the output comes from and the cycle in which it was produced[1] as well as the slot in the memory to which the result will be stored.

[1]This must always be the same cycle for ALU and LD/ST and -2 from the current cycle for multiplier, so this information is redundant, but it should make is easier to check the intent of the instructions.