Bilinear Filtering

Recall that the blend equation was:
\[ C_{\text{new}} = C_a \times f + C_b \times (1-f) \]

Where \( C_a, C_b \) were two 8-bit colors, and \( C_{\text{new}} \) was a blend of these two colors using the blend factor \( f \) (a 9-bit value).

A similar operation is performed when a texture is mapped onto an object in 3D graphics, except that 2 blend factors and four colors are used:
\[ T_{\text{new}} = (1-v)(1-u)T_{00} + (1-v)uT_{01} + v(1-u)T_{10} + uvT_{11} \]

\( T_{00}, T_{01}, T_{10}, T_{11} \) are 8-bit color values as before, with two 9-bit factors \( v, u \) used to determine \( T_{\text{new}} \).

Bilinear Filtering (cont)

We will use 9-bits to represent \( u, v \) as with the blend equation in order to represent 1.0 accurately.

Sample calculations:
\[ u=1.0, v=1.0, \text{ then } T_{\text{new}} = T_{11} \]
\[ u=0.0, v=1.0, \text{ then } T_{\text{new}} = T_{10} \]
\[ u=1.0, v=0.0, \text{ then } T_{\text{new}} = T_{01} \]
\[ u=0.0, v=0.0, \text{ then } T_{\text{new}} = T_{00} \]
\[ u = 0.5, v=0.5 \text{ then } T_{\text{new}} = 0.25 \times T_{00} + 0.25 \times T_{01} + 0.25 \times T_{10} + 0.25 \times T_{11} \]

The Problem

- Implement a datapath + FSM that computes 8 \( T_{\text{new}} \) values from 32 \( T_{xx} \) values stored in a RAM for fixed values of \( u, v \).
- Will use a minimum resource approach – only 1 multiplier, 1 adder.
  - Note that 8 multiplies and 3 adds are required to implement the bilinear filter equation
- You will be provided with a datapath
  - You must schedule the operations on the datapath
  - Write an ASM chart that implements the schedule
  - Implement the FSM for the datapath and test your design

How to Compute \( T_{\text{new}} \)?

- The Sync RAM holds the values for \( T_{xx} \)
  - Each calculation of \( T_{\text{new}} \) requires 4 values from the Sync Ram
  - Each 4-tuple stored in order of \( T_{00}, T_{01}, T_{10}, T_{11} \)
  - Sync Ram has 32 locations, so 8 \( T_{\text{new}} \) calculations
- Each calculation of \( Txx \times u|1-u \times v|1-v \) requires:
  - 1st multiply: \( Txx \) (from Sync Ram) \( \times v|1-v \) (use 4/1 mux to select appropriate \( v \) or \( 1-v \)). Store result in \( \text{mult reg} \).
  - 2nd multiply: compute \( \text{mult reg} \times u|1-u \times v|1-v \). Use the mult feedback path and mult muxes to select proper operands
- The saturating adder + accumulator register is used to accumulate the result.
Step 2: 1-u * mult_reg (active paths shown in RED)

Step 3: Mult reg = (1-u)(1-v) * T00, store in Acc reg
Compute (1-v) * T01 (in Sync Ram)
If acc initially zero, then can just add value.

Step 4: u * mult_reg (active paths shown in RED)

Step 5: Mult reg = (u)(1-v) * T01, add to Acc reg
Compute v * T10 (in Sync Ram)

Etc, etc… The rest of the steps are left up to you.

Datapath - bifilt.gdf

- The ZIP archive contains a datapath (bifilt.gdf) that you can use.
  - Cannot change the interface signals (inputs/outputs) or their functionality
  - Cannot change number of multipliers (1) or satadds (1), or size of sync SRAM (32 locations)
  - Make any other changes that you want
- Your datapath + FSM has to compute 8 values of T_{new} in 100 clock cycles (this constraint is easy to meet)
  - If your number of clock cycles matches or is less than the number of clock cycles in the golden waveform, then you will get 10 points added to any test grade.
- You will have to add a FSM to bifilt.gdf to complete the functionality
  - The exact number of states and the sequencing of datapath operations is up to you.
- You cannot use more than 16 states in your FSM.

Datapath - bifilt.gdf Interface

- Inputs
  - Clk, reset : clock and asynchronous reset
  - addb/[5..0] : drives address bus to SRAM when datapath is not in operation
  - din[8..0] : 9-bit input bus used to initialized u_reg, v_reg, SRAM contents
  - ld_v : when asserted, writing to v_reg (addr0 = '0') or u_reg (addr0 = '1')
  - we : when asserted, writing to SRAM using addr, din. Assume only asserted if datapath is not in operation
  - start : when asserted, start bifilt operation starting at SRAM location 0 and processing all 32 values in SRAM.
- SRAM, u_reg, v_reg are initialized externally to FSM control.
Datapath - bifilt.gdf Interface (cont)

- Outputs
  - Busy: asserted for duration of bifiltering operation
  - o_rdy: asserted when dout bus contains Tnew value
  - dout[7..0]: 8-bit output bus for Tnew value
- It is very important that o_rdy only be asserted when dout bus contains a valid value for Tnew.
- When o_rdy is negated, the value dout is undefined
  - Will depend on your particular implementation

Testing Your Design

- Cannot do a waveform compare against the golden waveform because you may have a different number of clock cycles
- The tb_bifilt.gdf schematic and tb_bifilt.scf is a testbench that can be used for checking.
- Includes a counter that will record the number of clock cycles that busy remains high during bifilter operation
- Includes an XOR-checksum that will checksum all values on dout when o_rdy is asserted
  - you can use this as a quick check – if your checksum matches the golden checksum then your design is functional

The Next Assignment

- This lab is worth 200 pts and is the first part of a 2-part series
- In the next part, you will be able to add more multipliers/adders to reduce the number of clocks
  - Single SRAM is still a constraint
  - Interface does not change
  - You will have to change the datapath and your FSM
- 2nd part is also worth 200 pts

Due Dates

- Schedule:
  - Week 0 (Sept 30): Demo Lab #5, begin working on Lab #6, Part #1
  - Week 1 (Oct 7): Must have ASM chart ready for checkoff and FSM VHDL code written/compiled and debug in datapath in progress. Part #2 will be assigned at this time.
  - Week 2 (Oct 14): Complete checkoff for Part #1 at beginning of lab.
  - Week 3 (Oct 21): Must demo Part #2 at the beginning of the lab period. Begin work on Lab #7.
- You must attend lab session for entire time each week until Lab #6 (both parts) is completed.
  - If you have a laptop, bring it to the lab. If you work on a desktop at home, then ftp the files to ECE machines. If you do not show up for lab, or do not remain for the entire time you will lose 30% credit of the 400 total points.
- No late labs accepted for either parts #1 or #2.

A Guaranteed Way to get a 0 for both labs and perhaps wreck your lab grade

- Don’t do anything the first week.
- Show up for lab in week #1 not even having thought about the first part.
  - Now you have only 2 weeks to complete two tough labs
- In week #2, won’t have first part working and understanding the first part is the key to performing the second part
  - You madly try to finish the 2nd part in week #3 but are clueless, so at the end of the 3 weeks you have 0 pts.
  - Total lab points for first 5 labs = 600 pts, so lab average is 600/1000 = 60% (assuming perfect scores on first 5 labs).
  - Remember that you must have at least a 60% on all out of class material to pass the course.