RTL Example: Video Compression – Sum of Absolute Differences

- Video is a series of frames (e.g., 30 per second)
- Most frames similar to previous frame
- Compression idea: just send difference from previous frame

- Need to quickly determine whether two frames are similar enough to just send difference for second frame
  - Compare corresponding 16x16 "blocks"
  - Treat 16x16 block as 256-byte array
  - Compute the absolute value of the difference of each array item
  - Sum those differences – if above a threshold, send complete frame for second frame; if below, can use difference method (using another technique, not described)

RTL Example: Video Compression – Sum of Absolute Differences

- Want fast sum-of-absolute-differences (SAD) component
  - When go=1, sums the differences of element pairs in arrays A and B, outputs that sum

RTL Example: Video Compression – Sum of Absolute Differences

- Inputs: A, B (256 byte memory); go (bit)
- Outputs: sad (32 bits)
- Local registers: sum, sad_reg (32 bits); i (9 bits)
RTL Example: Video Compression – Sum of Absolute Differences

Step 2: Create datapath

- Inputs: A, B (256 byte memory); go (bit)
- Outputs: sad (32 bits)
- Local registers: sum, sad_reg (32 bits); i (9 bits)

- Step 3: Connect to controller
- Step 4: Replace high-level state machine by FSM

Comparing software and custom circuit SAD

- Circuit: Two states (S2 & S3) for each i, 256 is/9 512 clock cycles
- Software: Loop (for i = 1 to 256), but for each i, must move memory to local registers, subtract, compute absolute value, add to sum, increment i = ~ 6 cycles per array item \( \rightarrow 256*6 = 1536 \) cycles
- Circuit is about 3 times (300%) faster
- Later, we’ll see how to build SAD circuit that is even faster

RTL Design Pitfalls and Good Practice

- Common pitfall: Reading outputs
  - Outputs can only be written
  - Solution: Introduce additional register, which can be written and read

Solutions

- Read register in following state (Q=R)
- Insert extra state so that conditions use updated value
- Other solutions are possible, depends on the example
RTL Design Pitfalls and Good Practice

- Good practice: Register all data outputs
  - In fig (a), output P would show spurious values as addition computes
    - Furthermore, longest register-to-register path, which determines clock period, is not known until that output is connected to another component
  - In fig (b), spurious outputs reduced, and longest register-to-register path is clear

Design Challenge

- Design Challenge
  - Using the RTL design method, convert the following high-level state machine to datapath and controller. Design the datapath to structure, but design the controller to the point of an FSM only.

Due Next Lecture (as announced in class)