ECE 337: ASIC Design Lab 4

Week of 9/9/2013

Submitting Lab 4

- Submit Commands:
  - Submit the Lab 4
    - submit Lab4AVG
  - Check number of attempts used & prior grades
    - submit Lab4AVG -c
- Submission notes:
  - Can use up to 5 times (this lab ONLY)
  - Grade based on last <u>mapped</u> submission. Use SVN!
  - Must achieve 50% to pass this lab
  - Top level port names and module name <u>needs to be IDENTICAL to what is listed in the handout</u> or you will waste a submission attempt.

Lab 4 Test Benches

- Can be shared via Blackboard
  - Only the test benches for the 4 Sample Averaging Filter Design
- Don’t depend exclusively on posted test benches
  - There’s no guarantee to their accuracy.
- Extra credit available for original test benches
  - At Instructor/TA Discretion (Maximum of 5 points toward lab)
  - Based on level of commenting
  - Structuring
  - Using advanced test benching concepts
  - Efficient testing (command usage and corner cases)

Reminders

- Collaboration
  - Can work with one other person – your choice for this lab
  - Only test benches/debugging is collaborative
  - Sharing code is prohibited!
    - <u>And we do check!</u>
  - See *Team Reports* for more information and instructions
- Class details
  - Midterm 1 is Thurs. 9:30-10:20 EE 117

State Machine Syntax

```
typedef enum bit [N:0] {st_state1,
  st_state2,...} stateType;
stateType state;
stateType nxt_state;

always @ (negedge n_rst, posedge clk)
begin : REG_LOGIC
  if(1'b0 == n_rst)
    state <= st_state1;
  else
    state <= nxt_state;
end
```

State Machine Syntax Continued

```
always @ (state, params)
begin : NXT_LOGIC
  // Set the default value(s)
  nxt_state = state;

  // Define the override/special case conditions
  case(state)
    st_state1:
    begin
      nxt_state = st_state2;
    end
    …
  endcase
end
```

State Machine Syntax Continued

```
always @ (state)
begin : OUT_LOGIC
  // Set the default value(s)
  flag = 1’b0;

  // Define the override/special case conditions
  case(state)
    st_state2:
    begin
      flag = 1’b1;
    end
    …
  endcase
end
```

Test Bench Clock Generation

```
always
begin : CLK_GEN
  tb_clk = 1’b0;
  #(CLK_PERIOD/2);
  tb_clk = 1’b1;
  #(CLK_PERIOD/2);
end
```

Preventing Latches – Default Case

```
-- perform 2^i
always (i)
begin
  case(i)
    2’b00:
    begin
      o = 4’d1;
    end
    …
    default:
    begin
      o = 4’d0;
    end
  endcase
end
```

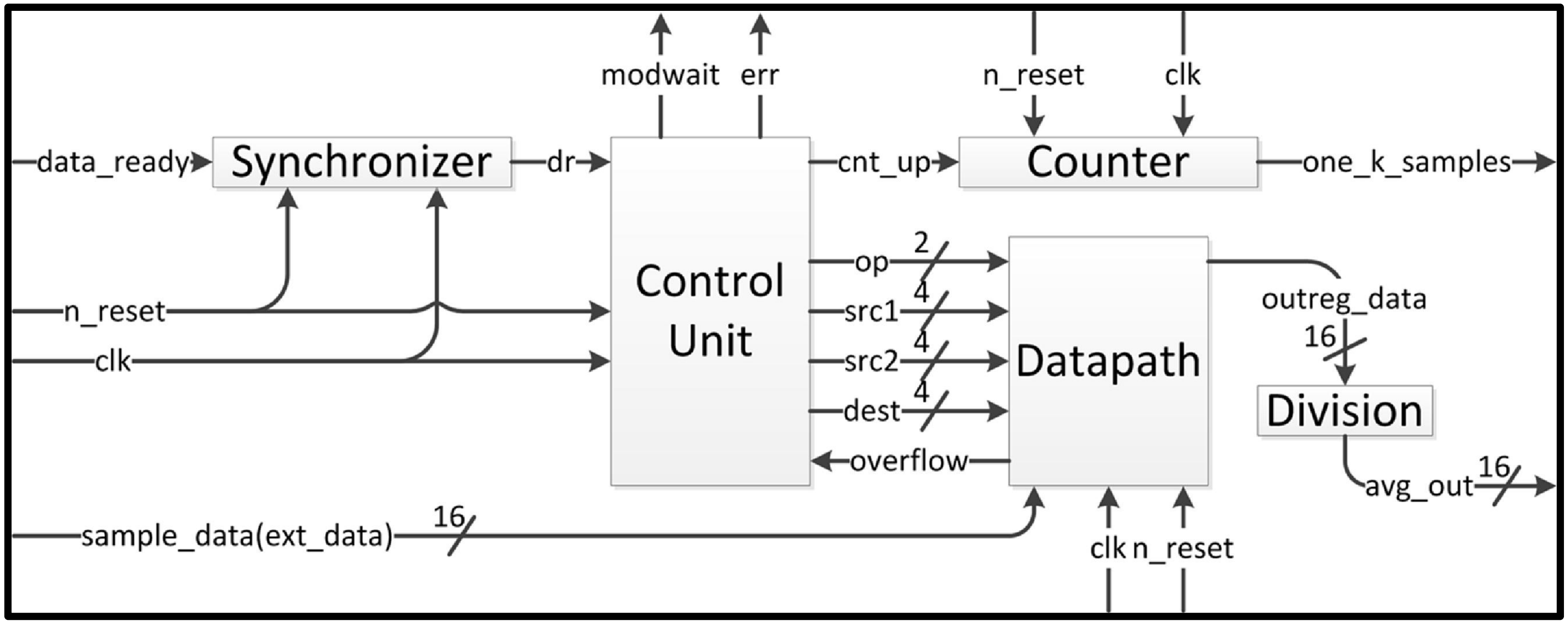
- Only prevents latches that would be caused by not specifying a possible case for ‘i’

Preventing Latches - Default Values

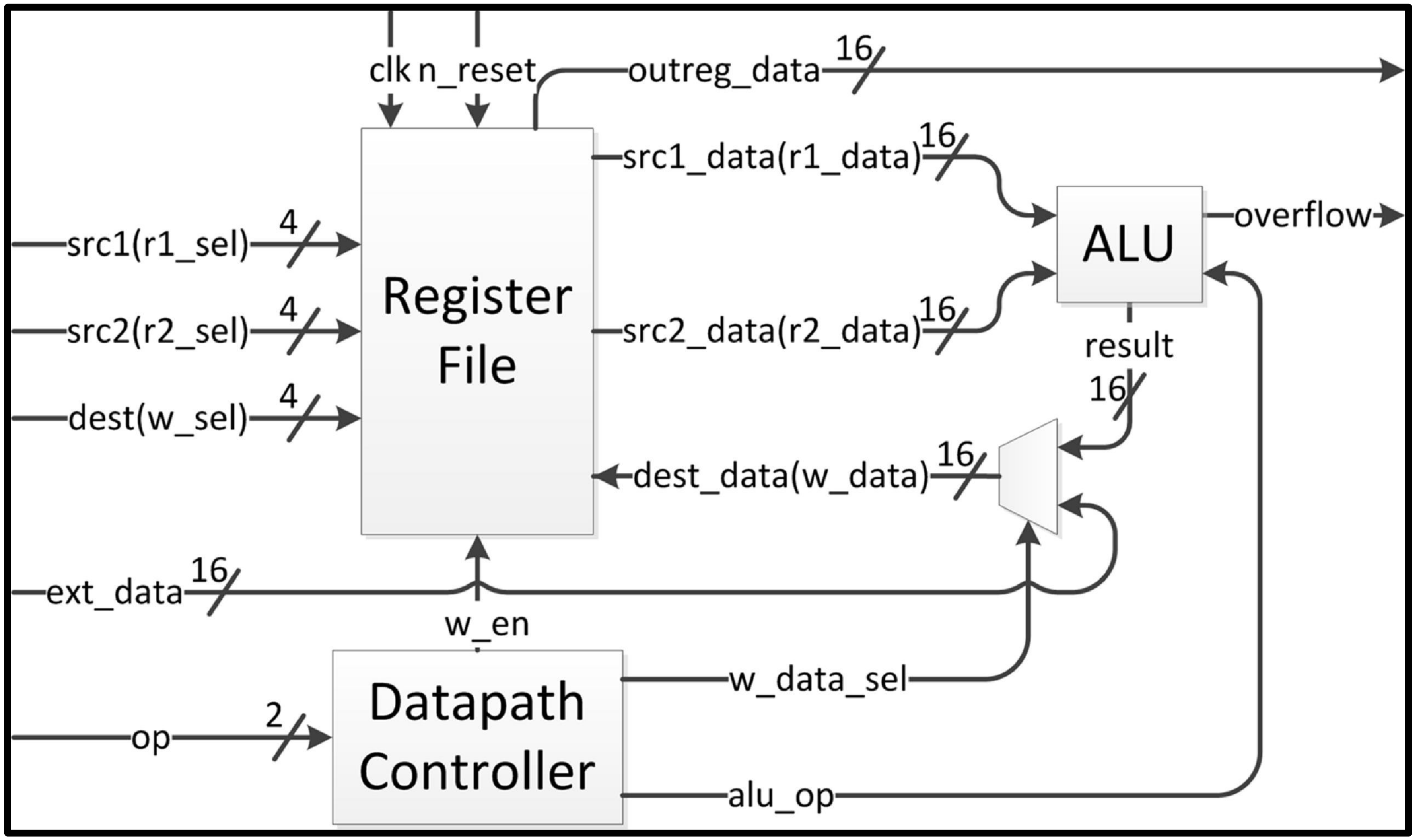
```
-- perform 2^i
always (i)
begin
  o = 4’d0;

  case(i)
    2’b00:
    begin
      o = 4’d1;
    end
    …
  endcase
end
```

- Will always prevent latches from happening
- Guarantees a value for all possible situations since later lines simply override a prior value assignment

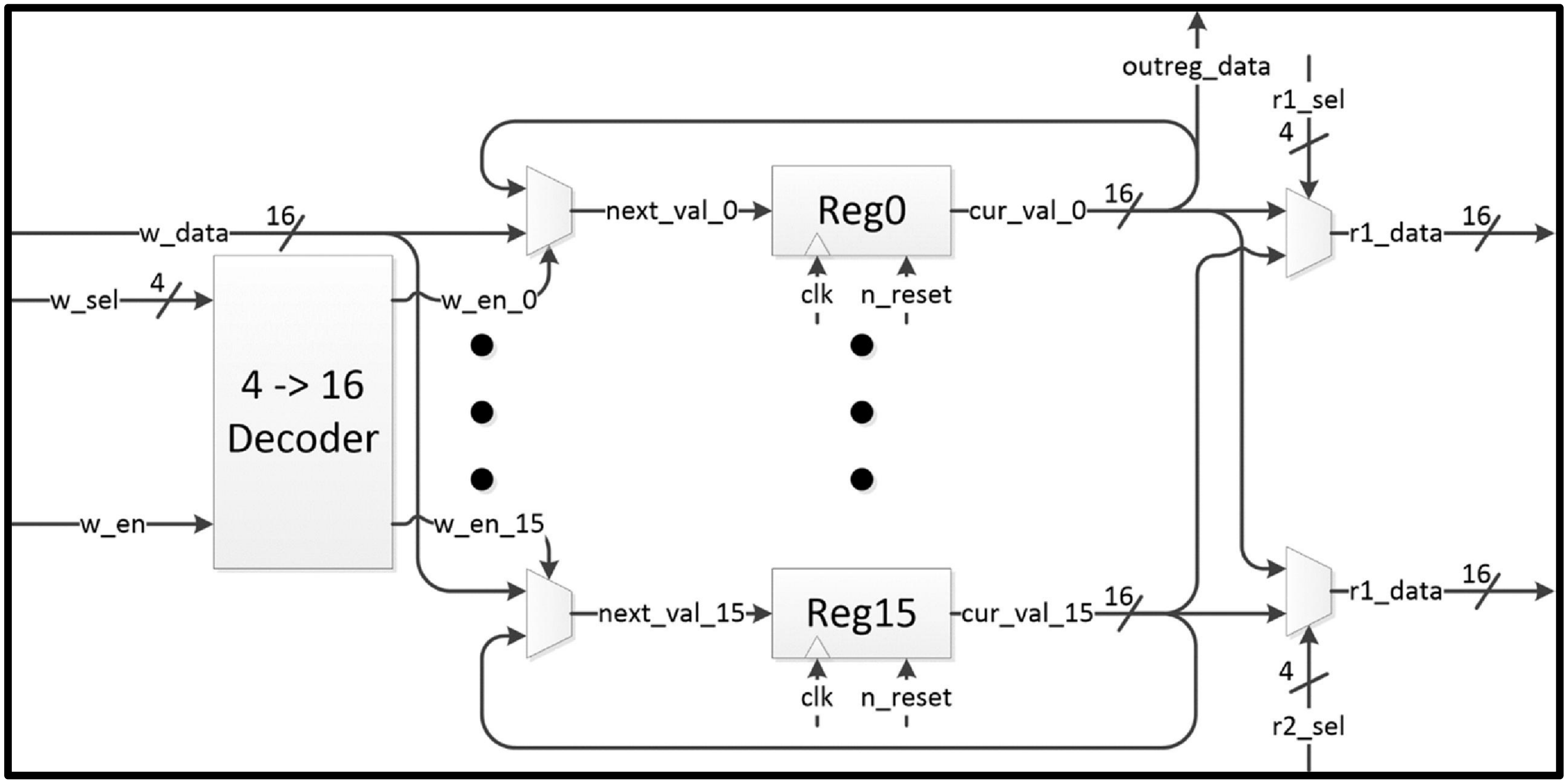
Lab 4: Sliding Window Average Chip



Inside the Datapath



Inside the Register File



Verilog File IO

- Very similar to C
- Personal file IO test bench example for part of postlab
  - Reads in a 24-bit bitmap file
  - Extracts the image data
  - Feeds the image data through 3 instances of the lab 4 design
    - 1 for each color slice of the pixel data
  - Creates a result file with the averaged pixel data
- File IO is a very useful and often necessary for testing complex designs (such as your projects)

Simulating Hierarchical Designs

- All sub-modules must be compiled in addition to top level and test bench files
  - Source simulations -> have to compile each file
  - Mapped simulations -> mapped file contains entire design
- Let the makefile do the work for you
  - Fill in the first 3 variables in the makefile
  - Use “make sim_full_source” to compile & simulate source
  - Use “make sim_full_mapped” to synthesize, compile, & simulate mapped versions

SVN Management

- `svn status`
  - **Gives a list of all changes to the working directory since the last revision**
- `svn diff <fname>`
  - **Displays changes in *fname* since the last revision. Leave blank to get an entire readout of changes made to a file**
- `svn revert <fname>`
  - **Discards changes made to *fname* since the last revision and restores it to the latest revision.**
  - **Leave blank to revert all files to the last version.**
  - ***Side Note: svn update merges changes, vs. svn revert discarding changes***
- `svn cat <fname>`
  - **Displays repository version of *fname***
- `svn log`
  - **Displays revision log messages for the current directory. This is useful for identifying the number of an old revision that you want to revert to.**
- `svn <command> -r <Revision Number>`
  - **Runs given command in relation to a given revision number.**