REAL-TIME DIGITAL SYSTEMS DESIGN AND VERIFICATION WITH FPGAS ECE 387 – LECTURE 6

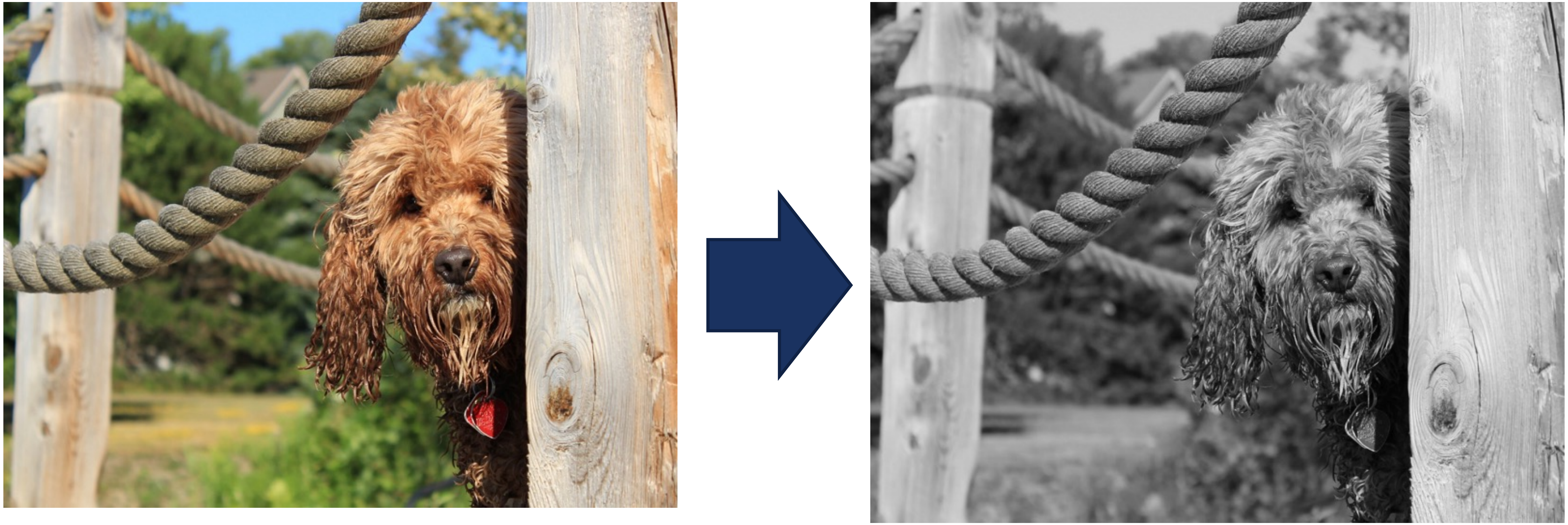
PROF. DAVID ZARETSKY

DAVID.ZARETSKY@NORTHWESTERN.EDU

AGENDA

- Streaming Architectures
- Image Processing Applications: Grayscale Conversion
- Homework 3: Motion Detection

IMAGE PROCESSING EXAMPLE: GRAYSCALE CONVERSION



GRAYSCALE CONVERSION IN SOFTWARE

- Grayscale is the average of RGB pixels
- Use streaming architecture to iterate over image pixels in sequence

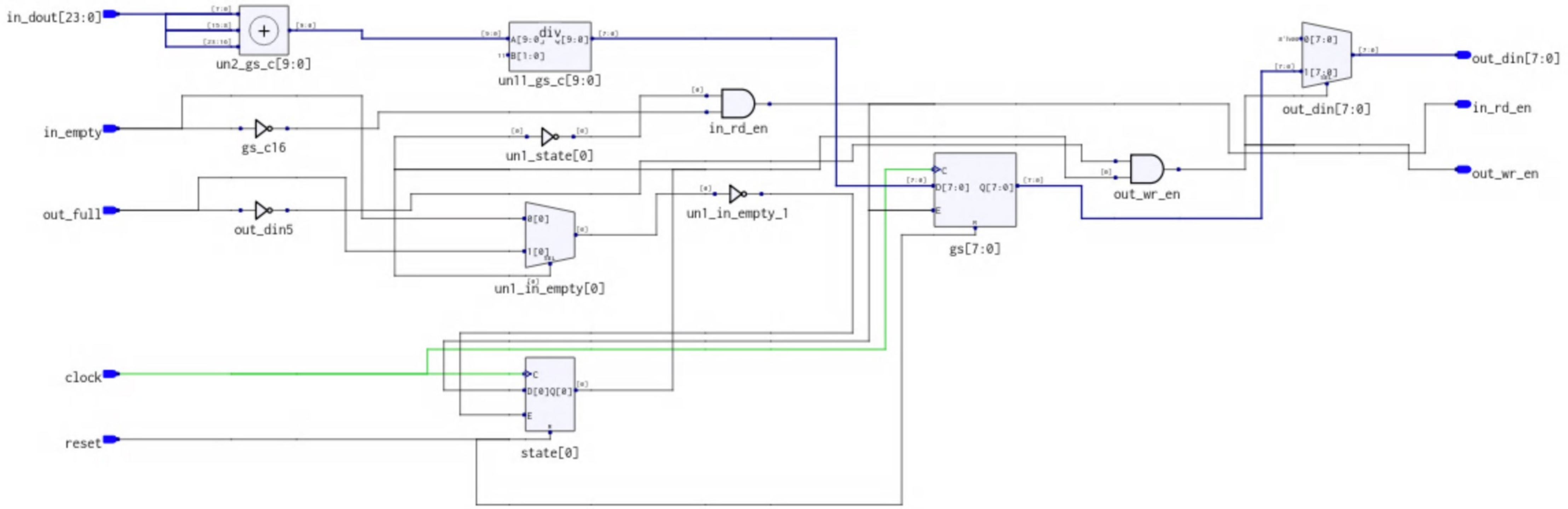
```
void convert_to_grayscale(struct pixel * data, int height, int width, unsigned char *grayscale_data)
{
    for (int i = 0; i < width*height; i++)
    {
         grayscale_data[i] = (data[i].r + data[i].g + data[i].b) / 3;
    }
}
```

GRAYSCALE CONVERSION IN SYSTEM VERILOG

```
module grayscale (
  input logic clock,
  input logic reset,
  output logic in_rd_en,
  input logic in_empty,
  input logic [23:0] in_dout,
  output logic out_wr_en,
  input logic out_full,
  output logic [7:0] out_din
);

typedef enum logic [0:0] {s0, s1} state_types;
state_types state, state_c;
logic [7:0] gs, gs_c;

always_ff @(posedge clock or posedge reset) begin
  if (reset == 1'b1) begin
    state <= s0;
    gs <= 8'h0;
  end else begin
    state <= state_c;
    gs <= gs_c;
  end
end
```

```
always_comb begin
  in_rd_en = 1'b0;
  out_wr_en = 1'b0;
  out_din = 8'b0;
  state_c = state;
  gs_c = gs;

  case (state)
    s0: begin
      if (in_empty == 1'b0) begin
        gs_c = 8'(($unsigned({2'b0, in_dout[23:16]}) +
                   $unsigned({2'b0, in_dout[15:8]}) +
                   $unsigned({2'b0, in_dout[7:0]})) / $unsigned(10'd3));
        in_rd_en = 1'b1;
        state_c = s1;
      end
    end

    s1: begin
       if (out_full == 1'b0) begin
          out_din = gs;
          out_wr_en = 1'b1;
          state_c = s0;
       end
    end

  endcase
end
endmodule
```

GRAYSCALE ARCHITECTURE



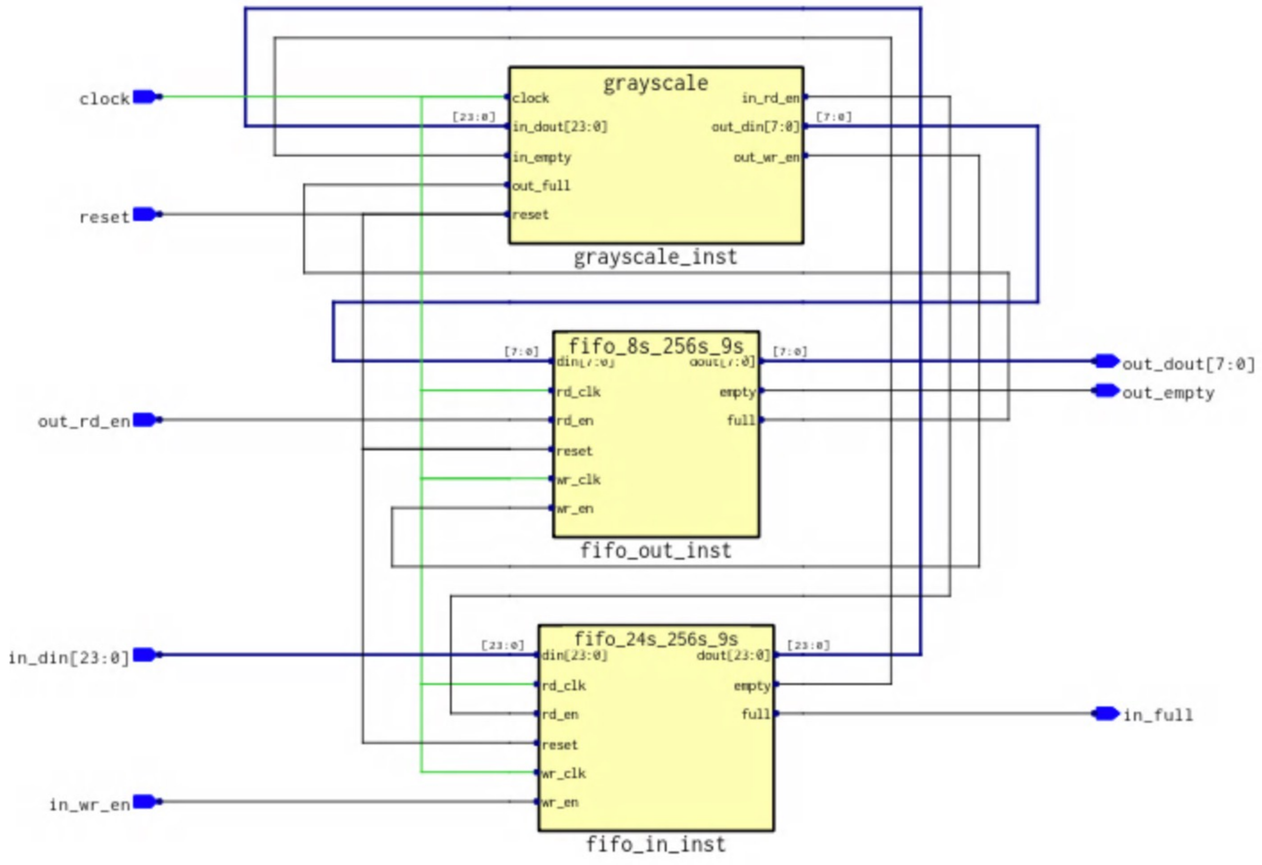
GRAYSCALE WRAPPER

```
module grayscale_top #(
   parameter WIDTH = 720,
   parameter HEIGHT = 540
) (
  input logic clock,
  input logic reset,
  output logic in_full,
  input logic in_wr_en,
  input logic [23:0] in_din,
  output logic out_empty,
  input logic out_rd_en,
  output logic [7:0] out_dout
);

logic [23:0] in_dout;
logic in_empty;
logic in_rd_en;
logic [7:0] out_din;
logic out_full;
logic out_wr_en;
```

```
grayscale #(
) grayscale_inst (
  .clock(clock),
  .reset(reset),
  .in_dout(in_dout),
  .in_rd_en(in_rd_en),
  .in_empty(in_empty),
  .out_din(out_din),
  .out_full(out_full),
  .out_wr_en(out_wr_en)
);

fifo #(
  .FIFO_BUFFER_SIZE(256),
  .FIFO_DATA_WIDTH(24)
) fifo_in_inst (
  .reset(reset),
  .wr_clk(clock),
  .wr_en(in_wr_en),
  .din(in_din),
  .full(in_full),
  .rd_clk(clock),
  .rd_en(in_rd_en),
  .dout(in_dout),
  .empty(in_empty)
);
```

```
fifo #(
  .FIFO_BUFFER_SIZE(256),
  .FIFO_DATA_WIDTH(8)
) fifo_out_inst (
  .reset(reset),
  .wr_clk(clock),
  .wr_en(out_wr_en),
  .din(out_din),
  .full(out_full),
  .rd_clk(clock),
  .rd_en(out_rd_en),
  .dout(out_dout),
  .empty(out_empty)
);

endmodule
```

GRAYSCALE WRAPPER TECHNOLOGY



GRAYSCALE TESTEBENCH

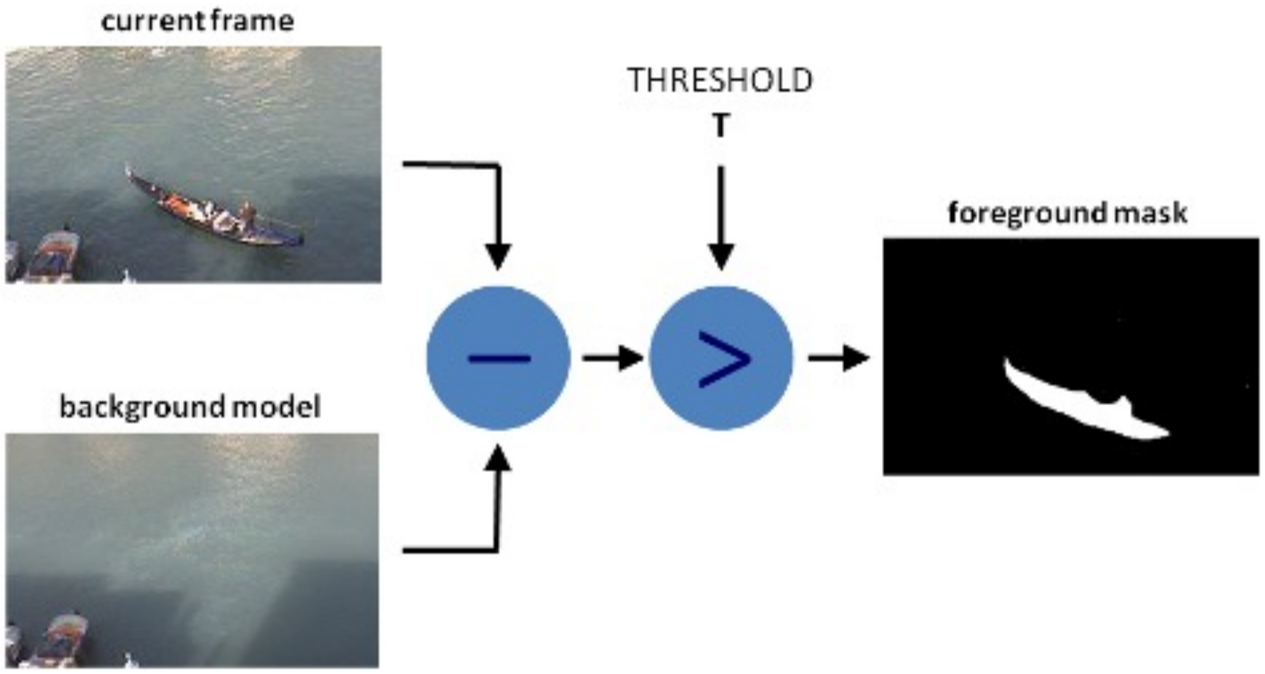
```
initial begin : img_read_process
  int i, r;
  int in_file;
  logic [7:0] bmp_header [0:BMP_HEADER_SIZE-1];

  @(negedge reset);
  $display("@ %0t: Loading file %s...", $time, IMG_IN_NAME);

  in_file = $fopen(IMG_IN_NAME, "rb");
  in_wr_en = 1'b0;

  // Skip BMP header
  r = $fread(bmp_header, in_file, 0, BMP_HEADER_SIZE);

  // Read data from image file
  i = 0;
  while ( i < BMP_DATA_SIZE ) begin
     @(negedge clock);
     in_wr_en = 1'b0;
     if (in_full == 1'b0) begin
        r = $fread(in_din, in_file, BMP_HEADER_SIZE+i, BYTES_PER_PIXEL);
        in_wr_en = 1'b1;
        i += BYTES_PER_PIXEL;
     end
  end

  @(negedge clock);
  in_wr_en = 1'b0;
  $fclose(in_file);
  in_write_done = 1'b1;
end
```

```
initial begin : img_write_process
  int i, r, out_file, cmp_file;
  logic [23:0] cmp_dout;
  logic [7:0] bmp_header [0:BMP_HEADER_SIZE-1];

   @(negedge reset);

   $display("@ %0t: Comparing file %s...", $time, IMG_OUT_NAME);
   out_file = $fopen(IMG_OUT_NAME, "wb");
   cmp_file = $fopen(IMG_CMP_NAME, "rb");
   out_rd_en = 1'b0;

   // Copy the BMP header
    r = $fread(bmp_header, cmp_file, 0, BMP_HEADER_SIZE);
    for (i = 0; i < BMP_HEADER_SIZE; i++) begin
       $fwrite(out_file, "%c", bmp_header[i]);
     end

    i = 0;
    while (i < BMP_DATA_SIZE) begin
       @(negedge clock);
       out_rd_en = 1'b0;
       if (out_empty == 1'b0) begin
          r = $fread(cmp_dout, cmp_file, BMP_HEADER_SIZE+i, BYTES_PER_PIXEL);
          $fwrite(out_file, "%c%c%c", out_dout, out_dout, out_dout);
          if (cmp_dout != {3{out_dout}}) begin
             out_errors += 1;
             $write("@ %0t: %s(%0d): ERROR: %x != %x.\n", $time, IMG_OUT_NAME, i+1, {3{out_dout}}, cmp_dout);
          end
          out_rd_en = 1'b1;
          i += BYTES_PER_PIXEL;
      end
   end

   @(negedge clock);
   out_rd_en = 1’b0;
   $fclose(out_file);
   $fclose(cmp_file);
   out_read_done = 1'b1;
end
```

HOMEWORK 3: MOTION DETECTION

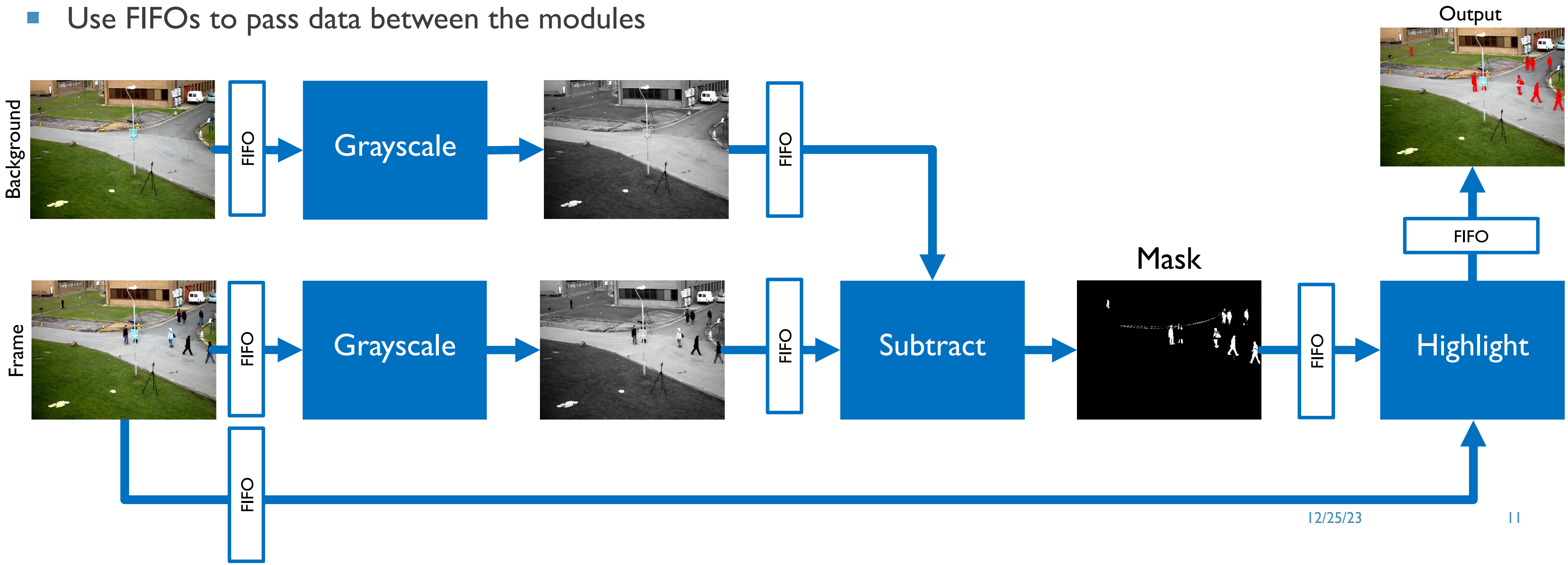
- Background subtraction is a common and widely used technique for identifying moving objects in a scene
- A threshold parameter is used to account for variance between images and background model
- Generally, a Gaussian filter or image averaging is used to reduce noise.

```
void subtract_background(unsigned char *base, unsigned char *img,
                         int height, int width, unsigned char *img_out)
{
    for (int y = 0; y < height; y++) {
        for (int x = 0; x < width; x++) {
           unsigned char data = (unsigned char)abs(img[y*width + x]
                   - base[y*width + x]);
           img_out[y * width + x] = data > THRESHOLD ? 0xFF : 0x00;
        }
    }
}
```



MOTION DETECT ARCHITECTURE

- Implement motion detection using background subtraction and then highlight motion.
- Use FIFOs to pass data between the modules



NEXT…

- HW #3: Motion Detection