Pomona College

Computer Science

Fall 2024: CSCI 181RT

Real-Time Systems in the Real World

Lecture 25

Thursday, November 21, 2024
Edmunds Hall 105
2:45 PM - 4:00 PM

Professor Jennifer DesCombes

Pomona College

Agenda

- Go Backs
- Discussion on Reading
- Discussion on Additional Assignments
- Lab #12 Review
- Course Assessment
- FPGA Programming Language and Structure
- Look Ahead
- Assignment
- Action Items

Pomona College

Go Backs

- General?
- Action Item Status
  - AI240910-2: Find recommended book on computer architecture.
  - AI240924-1: At what point as a development team grows does it make sense to have dedicated software and integration testers?
  - AI241024-1: Provide documentation on how to disable compiler optimization.
  - AI241107-1: Generate drawing showing location of Task Test Points on evaluation board.

Discussion on Readings

- The Soul Of A New Machine
  - Chapters 10: The Case of the Missing NAND Gate

Discussion on Additional Assignments

- BD-X60-Sorter-PDR-Hardware.pdf
- BD-X60-Sorter-PDR-Software.pdf

Pomona College

Lab #12 Review

- Achievements?
- If Working, Send Me Your Code

Course Assessment

- Provided Online Link via Email
- See Yinz In 20 Minutes

The Pittsburgh equivalent of "you all" is "**yinz**", which is a contracted form of "you ones":

- **Yinz**: A second-person plural pronoun used in Pittsburgh and other parts of Appalachia. It can also be spelled "yunz" or "you'unz".
- **Origin**: The term originated with Irish immigrants in the 1800s.

FPGA Programming Language and Structure

- Hardware Description Language (HDL)
  - Circuit Structures
  - Interconnects
  - Time

HDLs are standard text-based expressions of the structure of electronic systems and their behaviour over time. Like concurrent programming languages, HDL syntax and semantics include explicit notations for expressing concurrency. However, in contrast to most software programming languages, HDLs also include an explicit notion of time, which is a primary attribute of hardware. Languages whose only characteristic is to express circuit connectivity between a hierarchy of blocks are properly classified as netlist languages used in electric computer-aided design. HDL can be used to express designs in structural, behavioral or register-transfer-level architectures for the same circuit functionality; in the latter two cases the synthesizer decides the architecture and logic gate layout.

FPGA Programming Language and Structure

- Most Popular - VHDL, Verilog, SystemVerilog
  - The Differences Are Syntax - Common Concepts
- Many Other Applications Specific or Alternate Choices

Handel-C		C-like design language
Hardcaml	OCaml	Based on OCaml (embedded DSL)[20]
HHDL	Haskell	Based on Haskell (embedded DSL)
Hardware Join Java (HJJ)	Join Java	Based on Join Java
Hardware ML (HML)	Standard ML	Based on Standard ML[21]
Hydra	Haskell	Based on Haskell
Impulse C		C-like HDL
Parallel C++ (ParC)		kusu extended with HDL style threading and communication for task-parallel programming
JHDL	Java	Based on Java

FPGA Programming Language and Structure

- Toolsets and Processes Are Often Also Common (functionally)
  - Simulation (Modelsim)
  - Synthesis (conversion to a gate netlist)
  - Place and Route (mapping to specific hardware resources)
    - Device Utilization Impacts This Step
    - Designs With High Chip Utilization Take Longer and Can Fail to Place and Route
  - Timing Analysis
    - High Chip Utilization Designs Have More Timing Issues
    - Simulation (Modelsim)
  - Target Configuration and Download

FPGA Programming Language and Structure

- Structural Components of HDLs
  - Attributes (providing additional information about the design)
  - Defining Input / Output Ports (types, levels, speed, power)
  - Signals (representing data paths)
  - Data Types and Variables (like bit, logic, register)
  - Circuit Architectures (describing circuit behavior)
  - Operators (logic operations)
  - Control structures (like if-then-else, case statements)

Pomona College

FPGA Programming Structure - daq_fpga_top.ucf

- Attributes (providing additional information about the design)
- Defining Input / Output Ports (types, levels, speed, power)

```
#DIFF_TERM

#NET "sample_process_wrapper_instance/adc_dpm_ctrl_inst/w_wr_clk" USELOWSKEWLINES;
#NET "sample_process_wrapper_instance/adc_dpm_ctrl_inst/w_rd_clk" USELOWSKEWLINES;
#NET "sample_process_wrapper_instance/adc_dpm_ctrl_inst/w_rd_clk" USELOWSKEWLINES;

NET "fpga_ga<0>"                LOC = "W14"         |IOSTANDARD = LVCMOS33;
NET "fpga_ga<1>"                LOC = "Y13"         |IOSTANDARD = LVCMOS33;
NET "fpga_ga<2>"                LOC = "Y11"         |IOSTANDARD = LVCMOS33;

NET "io_c_val<0>"               LOC = "N7"          |IOSTANDARD = LVCMOS33 |PULLUP;
NET "io_c_val<1>"               LOC = "R4"          |IOSTANDARD = LVCMOS33 |PULLUP;
NET "io_c_val<2>"               LOC = "R3"          |IOSTANDARD = LVCMOS33 |PULLUP;
NET "io_c_val<3>"               LOC = "R9"          |IOSTANDARD = LVCMOS33 |PULLUP;
NET "io_c_val<4>"               LOC = "P8"          |IOSTANDARD = LVCMOS33 |PULLUP;

NET "io_c_wg<0>"                LOC = "N5"          |IOSTANDARD = LVCMOS33 |PULLUP;
NET "io_c_wg<1>"                LOC = "N4"          |IOSTANDARD = LVCMOS33 |PULLUP;
NET "io_c_wg<2>"                LOC = "P10"         |IOSTANDARD = LVCMOS33 |PULLUP;
NET "io_c_wg<3>"                LOC = "N9"          |IOSTANDARD = LVCMOS33 |PULLUP;
NET "io_c_wg<4>"                LOC = "M10"         |IOSTANDARD = LVCMOS33 |PULLUP;
```

Pomona College

FPGA Programming Language and Structure

- Structural Components of HDLs
  - Signals (representing data paths)

```
// ------------------------------ LED Indicators ------------------------------
output wire                         ch0_LED,      // To IO pin
output wire                         ch1_LED,      // To IO pin
output wire                         ch2_LED,      // To IO pin
output wire                         ch3_LED,      // To IO pin
output wire                         ch4_LED,      // To IO pin
output wire                         ch5_LED,      // To IO pin
output wire                         ch6_LED,      // To IO pin
output wire                         ch7_LED,      // To IO pin
output wire                         ch8_LED,      // To IO pin
output wire                         ch9_LED,      // To IO pin
```

```
// ============================================================================
// ---------------------- FPGA Geographic Address ---------------------------
// ============================================================================

wire [2:0] w_fpga_ga_buf;

IBUF ga_buf_2  (.I(fpga_ga[2]),  .O(w_fpga_ga_buf[2]));
IBUF ga_buf_1  (.I(fpga_ga[1]),  .O(w_fpga_ga_buf[1]));
IBUF ga_buf_0  (.I(fpga_ga[0]),  .O(w_fpga_ga_buf[0]));

assign w_daq_slot_num_mux = w_fpga_ga_buf;
```

Pomona College

FPGA Programming Language and Structure

- Structural Components of HDLs
  - Data Types and Variables (like bit, logic, register)

```
wire signed   [16:0] ch0_force_baseline_din;    // Force_baseline Control register; 17 bits
wire signed   [16:0] ch1_force_baseline_din;    // Force_baseline Control register; 17 bits
wire signed   [16:0] ch2_force_baseline_din;    // Force_baseline Control register; 17 bits
wire signed   [16:0] ch3_force_baseline_din;    // Force_baseline Control register; 17 bits
wire signed   [16:0] ch4_force_baseline_din;    // Force_baseline Control register; 17 bits
wire signed   [16:0] ch5_force_baseline_din;    // Force_baseline Control register; 17 bits
wire signed   [16:0] ch6_force_baseline_din;    // Force_baseline Control register; 17 bits
wire signed
wire signed
wire signed
```

```
reg [1:0]  r_data_valid;
reg [15:0] r_temp_data;

// sync to pcie clock
always@(posedge trn_clk_c)begin
       r_data_valid[0] <= w_temp_data_valid;
       r_data_valid[1] <= r_data_valid[0];

       if(r_data_valid[1])
              r_temp_data <= w_temp_data;
end
```

Pomona College

FPGA Programming Language and Structure

- Structural Components of HDLs
  - Circuit Architectures (describing circuit behavior)
  - Operators (logic operations)
  - Control structures (like if-then-else, case statements)

```
reg [1:0]  r_data_valid;
reg [15:0] r_temp_data;

// sync to pcie clock
always@(posedge trn_clk_c)begin
       r_data_valid[0] <= w_temp_data_valid;
       r_data_valid[1] <= r_data_valid[0];

       if(r_data_valid[1])
              r_temp_data <= w_temp_data;
end
```

FPGA Programming Language and Structure

- Structural Components of HDLs
  - Circuit Architectures (describing circuit behavior)
  - Operators (logic operations)
  - Control structures (like if-then-else, case statements)

```
always @ (posedge clk_25_n)
  begin
    if (~system_reset_n)
      processing_clk_phase <= #`CTOQ 1'b0;
    else if (processing_clk_phase_en)
      processing_clk_phase <= #`CTOQ (gated_clk_12_5_unbuf ^ ref_sync_clk_12_5);
    else
      processing_clk_phase <= #`CTOQ 1'b0;
  end
```

Pomona College

FPGA Programming Language and Structure

- Structural Components of HDLs
  - Circuit Architectures (describing circuit behavior)
  - Operators (logic operations)
  - Control structures (like if-then-else, case statements)

```
wire   w_acquire_en_in;           // Selected signal based on card location
reg    r_acquire_en_in;           // Registered version based on card location
assign w_acquire_en_in = (w_fpga_slot1_gating_sel ) ? g_wg_acquire_ena : ~w_acquire_en;
always@(posedge clk_12_5)begin
              if(~system_reset_n)begin
                     r_acquire_en_in <= 1'b0;
              end
              else begin
                     r_acquire_en_in <= w_acquire_en_in;
              end
       end
```

Pomona College

FPGA Programming Language and Structure

- Structural Components of HDLs
  - Circuit Architectures (describing circuit behavior)

```
// Channel 0 PMT Control
i2c_core_top
             #(
             .TX_REG_BITS(TX_REG_BITS),
             .RX_REG_BITS(RX_REG_BITS)
             )
             pmt_ch0(

                          // system signals
                          .i_reset_n(i_reset_n),
                          .i_sys_clk(i_sys_clk),
                          .i_i2c_baud_rate_divider(i_i2c_baud_rate_divider),

                          // user interface
                          .i_I2C_send_data(i_ch0_PMT_I2C_send_data),                  // 32 bits
                          .i_I2C_msg_byte_count(i_ch0_PMT_I2C_msg_byte_count),        // 8 bits
                          .i_I2C_msg_address(i_ch0_PMT_I2C_msg_address),              // 8 bits (seven address plus rd/wr)
                          .i_I2C_request(i_ch0_PMT_I2C_request),                      // I2C operation start request
                          .o_I2C_received_data(o_ch0_PMT_I2C_received_data),          // 32 bits
                          .o_I2C_slave_status(o_ch0_PMT_I2C_slave_status),            // I2C slave status (nack, scl, sda)
                          .o_I2C_finish_ack(o_ch0_PMT_I2C_finish_ack),                // I2C operation done acknowledge

                          // Physical i2c interface to Xilinx IO pins
                          .o_scl_padoen    (o_scl_padoen[0]),
                          .o_sda_padoen    (o_sda_padoen[0]),
                          .i_scl_pad        (i_scl_pad[0]),
                          .i_sda_pad        (i_sda_pad[0]),
                          .o_scl_pad        (o_scl_pad[0]),
                          .o_sda_pad        (o_sda_pad[0])
             );
```

FPGA Programming Language and Structure

- Structural Components of HDLs
- Multiple Instantiations of Logic

```
// Channel 1 PMT Control
i2c_core_top                    #(
                                .TX_REG_BITS(TX_REG_BITS),
                                .RX_REG_BITS(RX_REG_BITS)
                                )
                                pmt_ch1(

                                                        // system signals
                                                        .i_reset_n(i_reset_n),
                                                        .i_sys_clk(i_sys_clk),
                                                        .i_i2c_baud_rate_divider(i_i2c_baud_rate_divider),

                                                        // user interface
                                                        .i_I2C_send_data(i_ch1_PMT_I2C_send_data),                                              // 32 bits
                                                        .i_I2C_msg_byte_count(i_ch1_PMT_I2C_msg_byte_count),                    // 8 bits
                                                        .i_I2C_msg_address(i_ch1_PMT_I2C_msg_address),                          // 8 bits (seven address plus rd/wr)
                                                        .i_I2C_request(i_ch1_PMT_I2C_request),                  // I2C operation start request
                                                        .o_I2C_received_data(o_ch1_PMT_I2C_received_data),      // 32 bits
                                                        .o_I2C_slave_status(o_ch1_PMT_I2C_slave_status),                        // I2C slave status (nack, scl, sda)
                                                        .o_I2C_finish_ack(o_ch1_PMT_I2C_finish_ack),                            // I2C operation done acknowledge

                                                        // Physical i2c interface to Xilinx IO pins
                                                        .o_scl_padoen    (o_scl_padoen[1]),
                                                        .o_sda_padoen    (o_sda_padoen[1]),
                                                        .i_scl_pad       (i_scl_pad[1]),
                                                        .i_sda_pad       (i_sda_pad[1]),
                                                        .o_scl_pad       (o_scl_pad[1]),
                                                        .o_sda_pad       (o_sda_pad[1])
                                );

// Channel 2 PMT Control
i2c_core_top                    #(
                                .TX_REG_BITS(TX_REG_BITS),
                                .RX_REG_BITS(RX_REG_BITS)
                                )
                                pmt_ch2(

                                                        // system signals
                                                        .i_reset_n(i_reset_n),
                                                        .i_sys_clk(i_sys_clk),
                                                        .i_i2c_baud_rate_divider(i_i2c_baud_rate_divider),

                                                        // user interface
                                                        .i_I2C_send_data(i_ch2_PMT_I2C_send_data),                                              // 32 bits
                                                        .i_I2C_msg_byte_count(i_ch2_PMT_I2C_msg_byte_count),                    // 8 bits
                                                        .i_I2C_msg_address(i_ch2_PMT_I2C_msg_address),                          // 8 bits (seven address plus rd/wr)
                                                        .i_I2C_request(i_ch2_PMT_I2C_request),                  // I2C operation start request
                                                        .o_I2C_received_data(o_ch2_PMT_I2C_received_data),      // 32 bits
                                                        .o_I2C_slave_status(o_ch2_PMT_I2C_slave_status),                        // I2C slave status (nack, scl, sda)
                                                        .o_I2C_finish_ack(o_ch2_PMT_I2C_finish_ack),                            // I2C operation done acknowledge

                                                        // Physical i2c interface to Xilinx IO pins
                                                        .o_scl_padoen    (o_scl_padoen[2]),
                                                        .o_sda_padoen    (o_sda_padoen[2]),
                                                        .i_scl_pad       (i_scl_pad[2]),
                                                        .i_sda_pad       (i_sda_pad[2]),
                                                        .o_scl_pad       (o_scl_pad[2]),
                                                        .o_sda_pad       (o_sda_pad[2])
                                );

// Channel 3 PMT Control
i2c_core_top                    #(
                                .TX_REG_BITS(TX_REG_BITS),
                                .RX_REG_BITS(RX_REG_BITS)
                                )
                                pmt_ch3(

                                                        // system signals
                                                        .i_reset_n(i_reset_n),
                                                        .i_sys_clk(i_sys_clk),
                                                        .i_i2c_baud_rate_divider(i_i2c_baud_rate_divider),

                                                        // user interface
                                                        .i_I2C_send_data(i_ch3_PMT_I2C_send_data),                                              // 32 bits
                                                        .i_I2C_msg_byte_count(i_ch3_PMT_I2C_msg_byte_count),                    // 8 bits
                                                        .i_I2C_msg_address(i_ch3_PMT_I2C_msg_address),                          // 8 bits (seven address plus rd/wr)
                                                        .i_I2C_request(i_ch3_PMT_I2C_request),                  // I2C operation start request
                                                        .o_I2C_received_data(o_ch3_PMT_I2C_received_data),      // 32 bits
                                                        .o_I2C_slave_status(o_ch3_PMT_I2C_slave_status),                        // I2C slave status (nack, scl, sda)
                                                        .o_I2C_finish_ack(o_ch3_PMT_I2C_finish_ack),                            // I2C operation done acknowledge

                                                        // Physical i2c interface to Xilinx IO pins
                                                        .o_scl_padoen    (o_scl_padoen[3]),
                                                        .o_sda_padoen    (o_sda_padoen[3]),
                                                        .i_scl_pad       (i_scl_pad[3]),
                                                        .i_sda_pad       (i_sda_pad[3]),
                                                        .o_scl_pad       (o_scl_pad[3]),
                                                        .o_sda_pad       (o_sda_pad[3])
                                );
```

Pomona College

FPGA Programming Language and Structure

- Processing is NOT Sequential
- All Three Instantiations Occur Simultaneously
- All

  “`always@(posedge clk_12_5)begin`” Occur Simultaneously
- Multiple Higher Level Instantiations All Occur Simultaneously
- Everything Occurs Per Its Defined Logic

```
// Channel 1 PMT Control
i2c_core_top                    #(
                                .TX_REG_BITS(TX_REG_BITS),
                                .RX_REG_BITS(RX_REG_BITS)
                                )
                                pmt_ch1(

                                                        // system signals
                                                        .i_reset_n(i_reset_n),
                                                        .i_sys_clk(i_sys_clk),
                                                        .i_i2c_baud_rate_divider(i_i2c_baud_rate_divider),

                                                        // user interface
                                                        .i_I2C_send_data(i_ch1_PMT_I2C_send_data),                                      // 32 bits
                                                        .i_I2C_msg_byte_count(i_ch1_PMT_I2C_msg_byte_count),            // 8 bits
                                                        .i_I2C_msg_address(i_ch1_PMT_I2C_msg_address),                                  // 8 bits (seven address plus rd/wr)
                                                        .i_I2C_request(i_ch1_PMT_I2C_request),          // I2C operation start request
                                                        .o_I2C_received_data(o_ch1_PMT_I2C_received_data),      // 32 bits
                                                        .o_I2C_slave_status(o_ch1_PMT_I2C_slave_status),                        // I2C slave status (nack, scl, sda)
                                                        .o_I2C_finish_ack(o_ch1_PMT_I2C_finish_ack),                            // I2C operation done acknowledge

                                                        // Physical i2c interface to Xilinx IO pins
                                                        .o_scl_padoen    (o_scl_padoen[1]),
                                                        .o_sda_padoen    (o_sda_padoen[1]),
                                                        .i_scl_pad       (i_scl_pad[1]),
                                                        .i_sda_pad       (i_sda_pad[1]),
                                                        .o_scl_pad       (o_scl_pad[1]),
                                                        .o_sda_pad       (o_sda_pad[1])
                                );

// Channel 2 PMT Control
i2c_core_top                    #(
                                .TX_REG_BITS(TX_REG_BITS),
                                .RX_REG_BITS(RX_REG_BITS)
                                )
                                pmt_ch2(

                                                        // system signals
                                                        .i_reset_n(i_reset_n),
                                                        .i_sys_clk(i_sys_clk),
                                                        .i_i2c_baud_rate_divider(i_i2c_baud_rate_divider),

                                                        // user interface
                                                        .i_I2C_send_data(i_ch2_PMT_I2C_send_data),                                      // 32 bits
                                                        .i_I2C_msg_byte_count(i_ch2_PMT_I2C_msg_byte_count),            // 8 bits
                                                        .i_I2C_msg_address(i_ch2_PMT_I2C_msg_address),                                  // 8 bits (seven address plus rd/wr)
                                                        .i_I2C_request(i_ch2_PMT_I2C_request),          // I2C operation start request
                                                        .o_I2C_received_data(o_ch2_PMT_I2C_received_data),      // 32 bits
                                                        .o_I2C_slave_status(o_ch2_PMT_I2C_slave_status),                        // I2C slave status (nack, scl, sda)
                                                        .o_I2C_finish_ack(o_ch2_PMT_I2C_finish_ack),                            // I2C operation done acknowledge

                                                        // Physical i2c interface to Xilinx IO pins
                                                        .o_scl_padoen    (o_scl_padoen[2]),
                                                        .o_sda_padoen    (o_sda_padoen[2]),
                                                        .i_scl_pad       (i_scl_pad[2]),
                                                        .i_sda_pad       (i_sda_pad[2]),
                                                        .o_scl_pad       (o_scl_pad[2]),
                                                        .o_sda_pad       (o_sda_pad[2])
                                );

// Channel 3 PMT Control
i2c_core_top                    #(
                                .TX_REG_BITS(TX_REG_BITS),
                                .RX_REG_BITS(RX_REG_BITS)
                                )
                                pmt_ch3(

                                                        // system signals
                                                        .i_reset_n(i_reset_n),
                                                        .i_sys_clk(i_sys_clk),
                                                        .i_i2c_baud_rate_divider(i_i2c_baud_rate_divider),

                                                        // user interface
                                                        .i_I2C_send_data(i_ch3_PMT_I2C_send_data),                                      // 32 bits
                                                        .i_I2C_msg_byte_count(i_ch3_PMT_I2C_msg_byte_count),            // 8 bits
                                                        .i_I2C_msg_address(i_ch3_PMT_I2C_msg_address),                                  // 8 bits (seven address plus rd/wr)
                                                        .i_I2C_request(i_ch3_PMT_I2C_request),          // I2C operation start request
                                                        .o_I2C_received_data(o_ch3_PMT_I2C_received_data),      // 32 bits
                                                        .o_I2C_slave_status(o_ch3_PMT_I2C_slave_status),                        // I2C slave status (nack, scl, sda)
                                                        .o_I2C_finish_ack(o_ch3_PMT_I2C_finish_ack),                            // I2C operation done acknowledge

                                                        // Physical i2c interface to Xilinx IO pins
                                                        .o_scl_padoen    (o_scl_padoen[3]),
                                                        .o_sda_padoen    (o_sda_padoen[3]),
                                                        .i_scl_pad       (i_scl_pad[3]),
                                                        .i_sda_pad       (i_sda_pad[3]),
                                                        .o_scl_pad       (o_scl_pad[3]),
                                                        .o_sda_pad       (o_sda_pad[3])
                                );
```

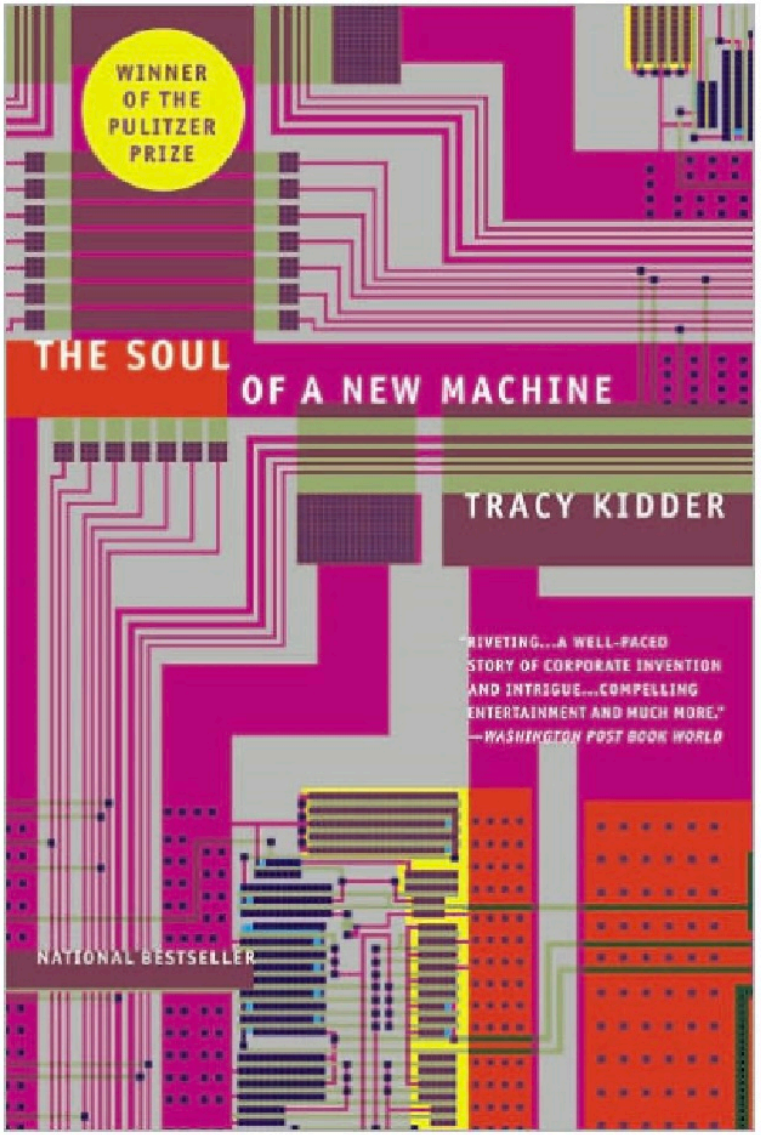
Look Ahead

- Discussion on Reading
- Discussion on TSoaNM or MMM
- Fielder's Choice
- Topics Covered and Not Covered
- Preview of Lab 13
- The End

Pomona College

Assignment - Readings

- The Soul Of A New Machine
  - Chapters 11 - 16 & Epilog: Shorter Than A Season, Pinball, Going To The Fair, The Last Crunch, Canards (discusses PALs), Dinosaurs (insightful), Epilog
  - Send Me One Comment on the Reading by 10:00 AM on Tuesday, December 3, 2024.
  - Have Ready to Read One General Comment on Either Soul of the New Machine or Mythical Man Month.



Pomona College

Action Items and Discussion

AI#:	Owner	Slide #	Document	Action