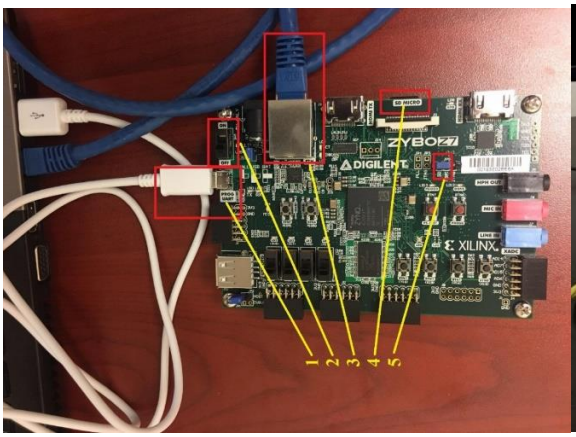
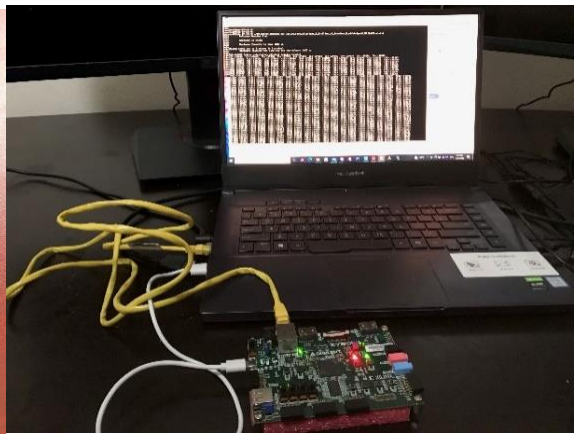
Fall 2022: EE 409/556 Hardware/Software Codesign

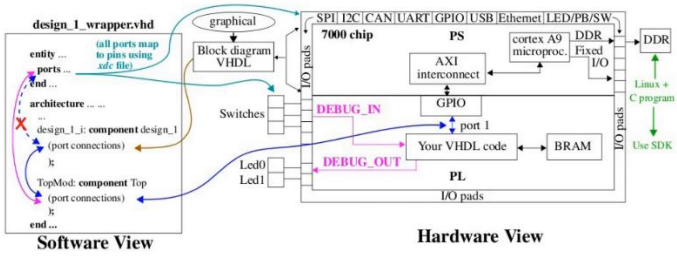
This course introduces design techniques of embedded and computing systems that integrate both hardware and software. Topics include dataflow modeling, software and hardware implementations of dataflows, analysis of control flows and dataflows, FSM with Datapath, microprogramming, embedded cores, etc. **Students will gain hands-on experiences and skills in implementing hardware & software co-design solutions for real-world problems on commercial SoC-FPGA: the Xilinx Zynq System-on-Chip (SoC) platform that integrates a dual-core ARM Cortex-A9 processor and FPGA fabrics. This course intends to better prepare student for future industrial and academic careers in this field.**

Lab and Project Setup

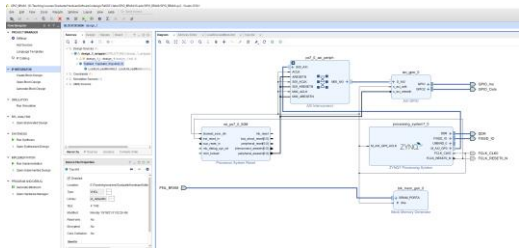




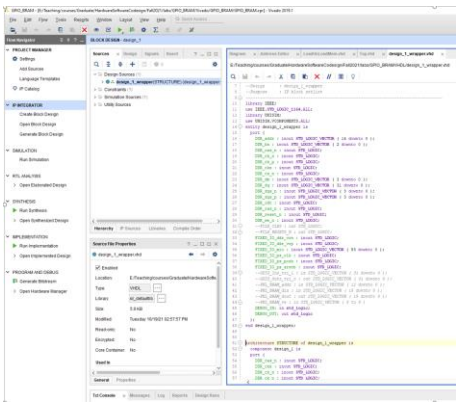
Zynq SoC Overview:



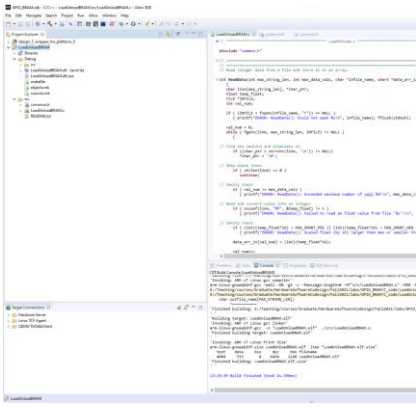
Vivado Zynq FPGA-SoC Block Diagram Design:



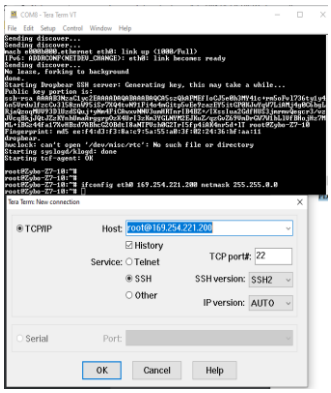
Vivado HW Design Wrapper



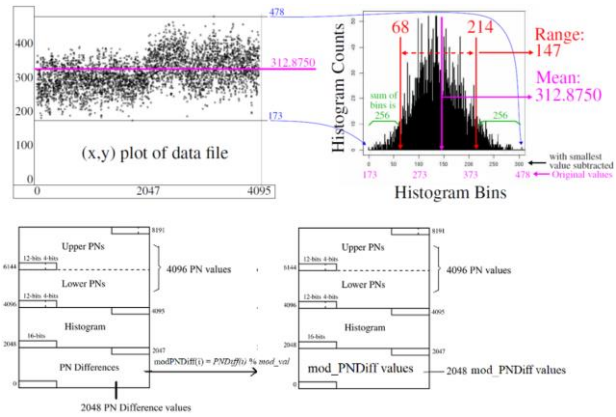
Xilinx SDK (SW Development)

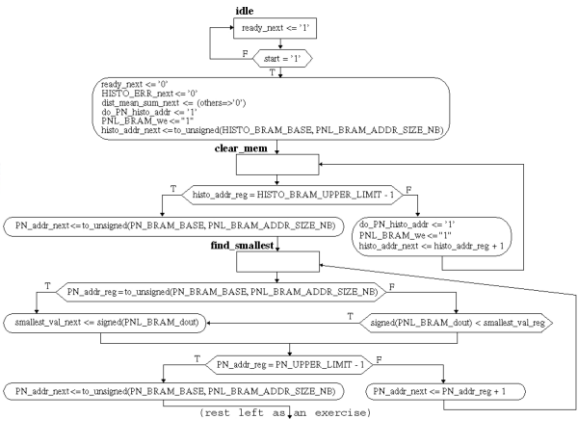


Zynq Linux Ethernet Config.

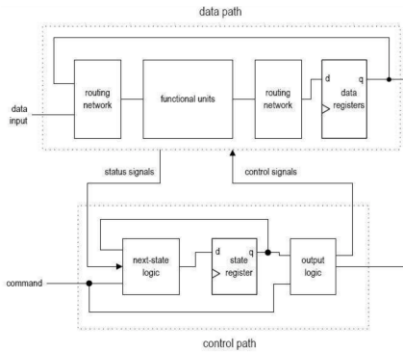


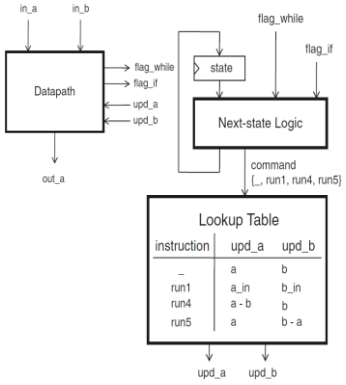
Labs and Projects Snapshots:





Data Flow and Control Flow Design in Hardware:





Software Design Snapshot (C code)

Histo C code

```
for ( PN_num = 0; PN_num < num_vals; PN_num++ )
   {
   dist_mean_sum += (int)vals[PN_num];
   temp_val = vals[PN_num]/precision_scaler - smallest_val;
   if ( temp_val >= DIST_range )
      HISTO_ERR = 1;
   software_histo[temp_val]++;
   }

LV_addr = 0; HV_addr = 0;
LV_set = 0; HV_set = 0;
dist_cnt_sum = 0;
for ( bin_num = 0; bin_num < DIST_range; bin_num++ )
   {
   dist_cnt_sum += software_histo[bin_num];
   if ( LV_set == 0 && dist_cnt_sum >= LV_bound )
      {
      LV_addr = bin_num;
      LV_set = 1;
      }
   if ( dist_cnt_sum <= HV_bound )
      {
      HV_addr = bin_num;
      HV_set = 1;
      }
   }
range = HV_addr - LV_addr + 1;
```

Hardware Design Snapshot (VHDL RTL)

```
- ======================
- With all the counts computed and stored in the upper portion of the PNL_BRAM, commense the parse from left to right
- to determine the range of the distribution.
        when sweep_BRAM =>

- Select histo memory
            do_PN_histo_addr <= '1';

- Add the count in the histo cell to the sum. NOTE: The counts are unsigned values. Note we resize to 13 bit to accomodate
- the value 4096
            dist_cnt_sum_next <= dist_cnt_sum_reg + resize(unsigned(PNL_BRAM_dout), NUM_PNS_NB+1);

- Assign the LV address just when dist_cnt_sum_next becomes >= than the LV bound. LV_set also used to handle case
- where the dist_sum never becomes >= LV_bound and is therefore, never assigned (which would happen if ALL the PMs are the
- same value or span a very small range).
            if ( LV_set_reg = '0' and dist_cnt_sum_next >= LV_bound ) then
              LV_addr_next <= histo_addr_reg;
              LV_set_next <= '1';
            end if;

- Keep assigning HV address while dist_cnt_sum_next is smaller than the bound. If it happens that the first address has a
- count that's larger than the HV_bound, then HV_set is never set.
            if ( dist_cnt_sum_next <= HV_bound ) then
              HV_addr_next <= histo_addr_reg;
              HV_set_next <= '1';
            end if;

- Check if entire distribution has been swept.
            if ( histo_addr_reg = HISTO_BRAM_UPPER_LIMIT - 1 ) then
              state_next <= check_histo_error;
            else
              histo_addr_next <= histo_addr_reg + 1;
            end if;

- ======================
- Check if the distribution is TOO narrow to be characterized by our bounds. Set the error flag if true.
        when check_histo_error =>
            if ( LV_set_reg = '0' or HV_set_reg = '0' ) then
              HISTO_ERR_next <= '1';
              state_next <= idle;

- Just store the mean and range in the lowest portion of memory (addresses 0 and 1) for transfer to C program.
            else
              do_PN_histo_addr <= '1';
              histo_addr_next <= to_unsigned(HISTO_BRAM_UPPER_LIMIT - 2, PNL_BRAM_ADDR_SIZE_NB);
              PNL_BRAM_we <= "1";
              PNL_BRAM_din <= dist_mean;
              state_next <= write_range;
            end if;

- ======================
- Write range at address 1
        when write_range =>
```