Weather Sensor Interface for FPGA

Andrew Galczyk, August Lile, Adam Marszalek, Andrew Meesseman

ECE 4710: Computer Hardware Design Winter 2020

Professor D. Llamocca

Chip: DF Robot Environmental Sensor (SPI or I2C)

Figure 1a:
Back side of sensor (SPI interface). 6 pin connection. VCC,GND,MOSI,CS,SCLK,MISO

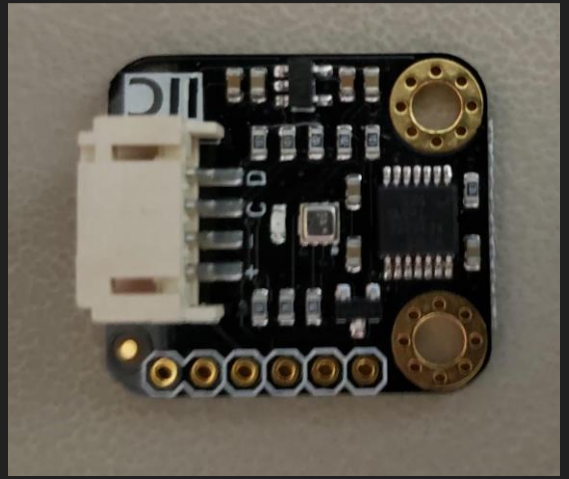


Figure 1b:
Front side of sensor (i2c interface). 4 pin connector (+,-,C,D)

Sensor: Bosch BME280 Combined Humidity and Pressure Sensor

BOSCH

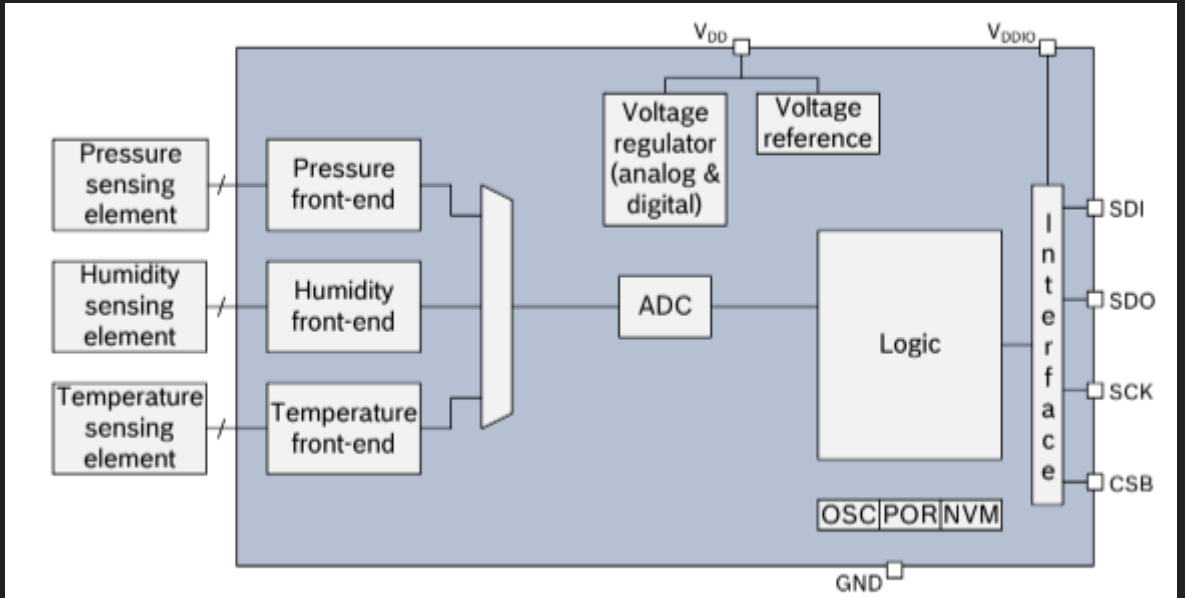
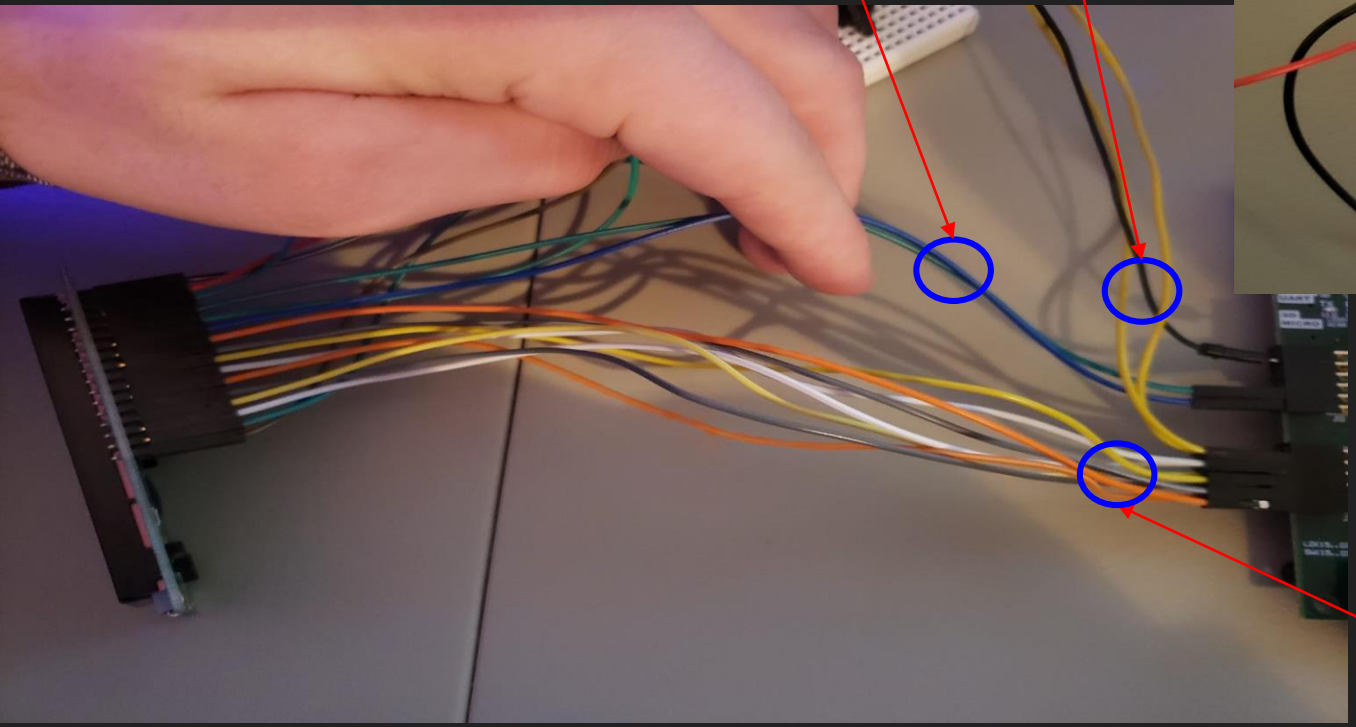
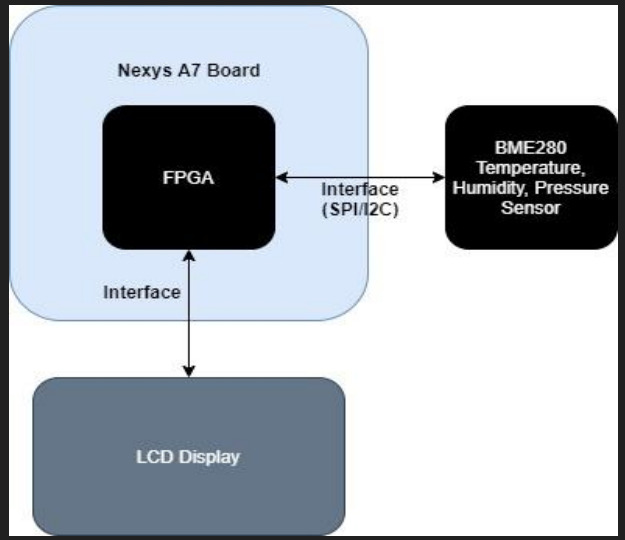


Figure 2: Block diagram of BME280

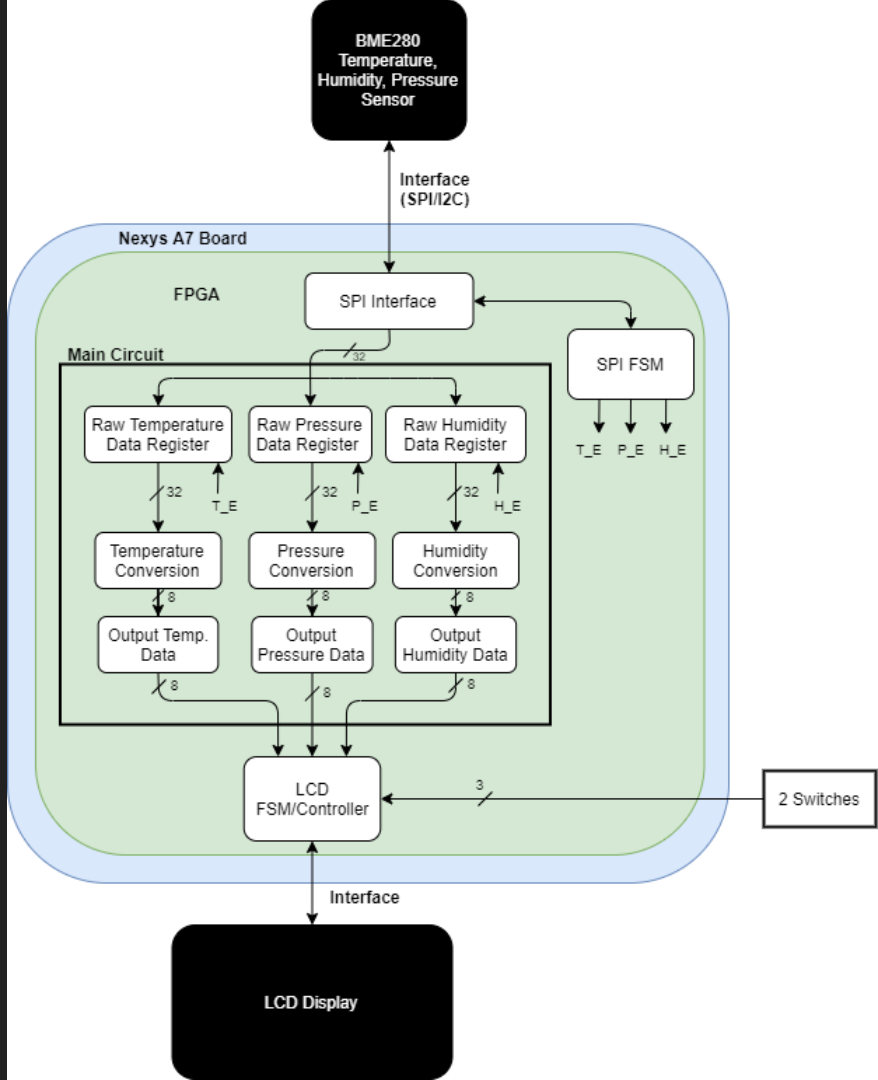
Hitachi HD44780 LCD



High Level Methodology

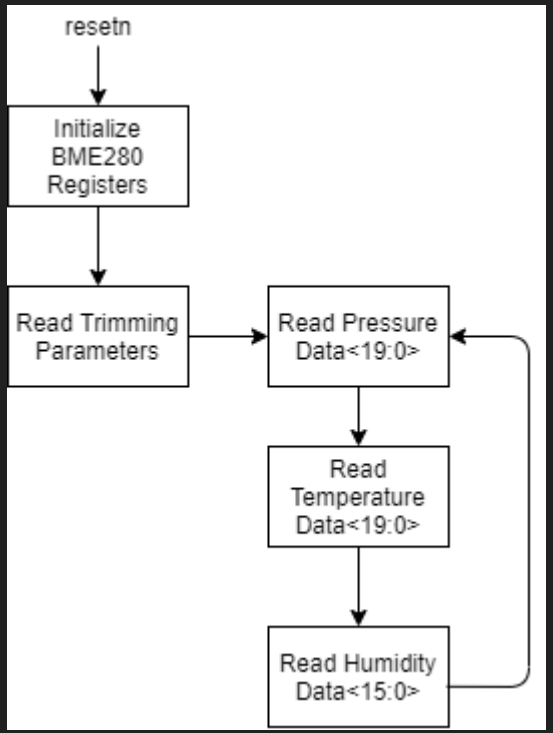


- Raw data will be read from the sensor by SPI
- Data will be converted by FPGA
- Then printed on the LCD



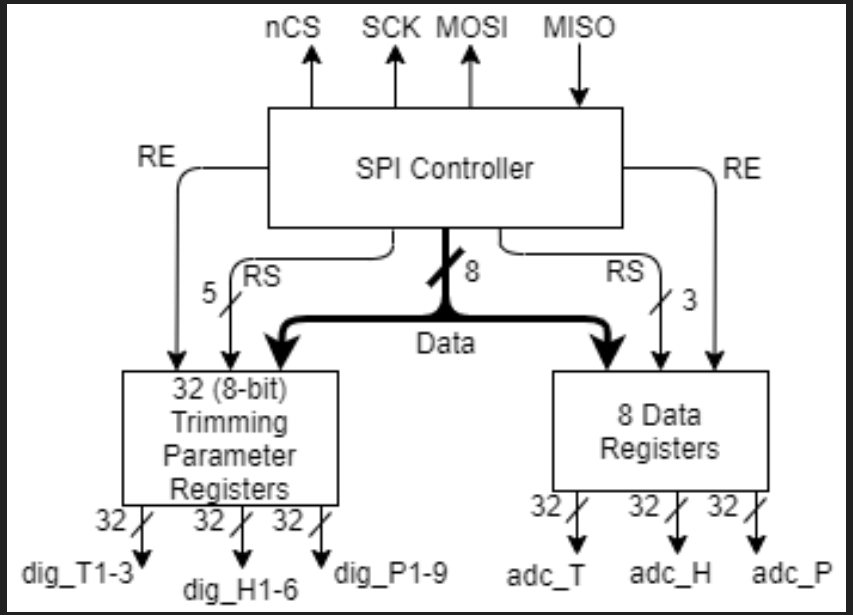
SPI High Level Design

1. Initialize BME 280 registers
    a. Write to 0xE0
        i. Perform Power-on Reset (PoR)
    b. Write to 0xF5
        i. Sampling interval set to 500ms in normal mode
        ii. Filter off
    c. Write to 0xF2
        i. Humidity oversampling*1
    d. Write to 0xF4
        i. Temperature oversampling*1
        ii. Pressure oversampling*1
        iii. Set sensor to normal mode
2. Read the 32 trimming parameters
    a. Used in data conversion circuits
3. Loop to read raw ADC values



SPI Interface Architecture

- Values used for temperature conversion are stored in 8-bit registers
- These are then converted into 32-bit buses accordingly

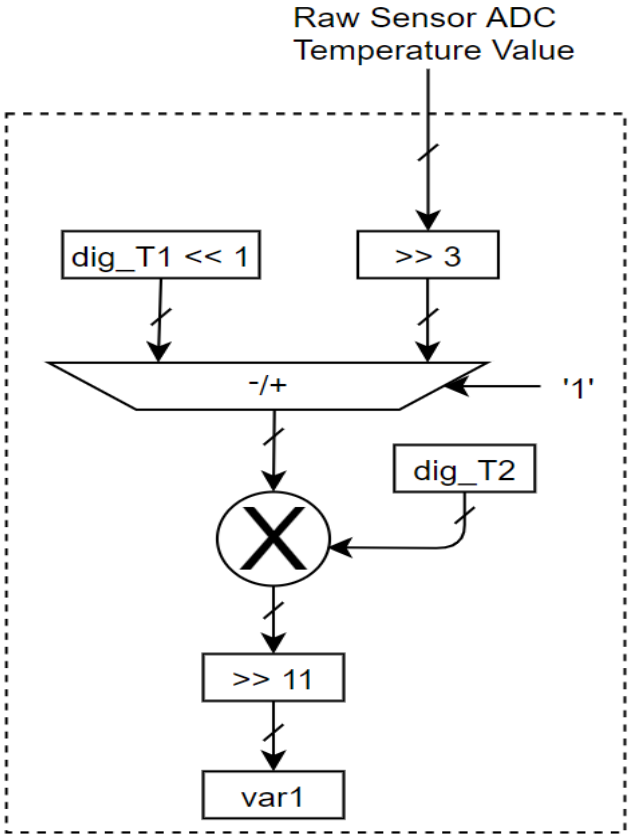


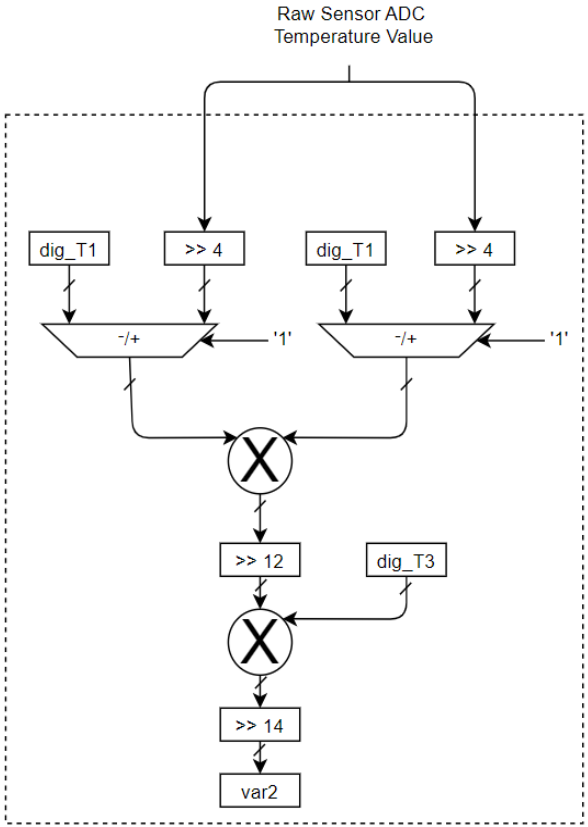
Conversion Formulas from Datasheet (1/3)

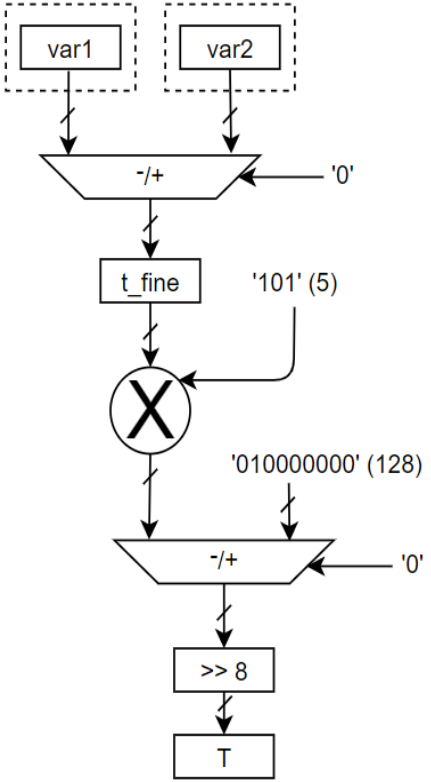
Temperature

```
// Returns temperature in DegC, resolution is 0.01 DegC. Output value of “5123” equals 51.23
DegC.
// t_fine carries fine temperature as global value
BME280_S32_t t_fine;
BME280_S32_t BME280_compensate_T_int32(BME280_S32_t adc_T)
{
   BME280_S32_t var1, var2, T;
   var1  = ((((adc_T>>3) - ((BME280_S32_t)dig_T1<<1))) * ((BME280_S32_t)dig_T2)) >> 11;
   var2  = (((((adc_T>>4) - ((BME280_S32_t)dig_T1)) * ((adc_T>>4) - ((BME280_S32_t)dig_T1)))
>> 12) *
      ((BME280_S32_t)dig_T3)) >> 14;
   t_fine = var1 + var2;
   T  = (t_fine * 5 + 128) >> 8;
   return T;
}
```

Temperature Conversion Digital





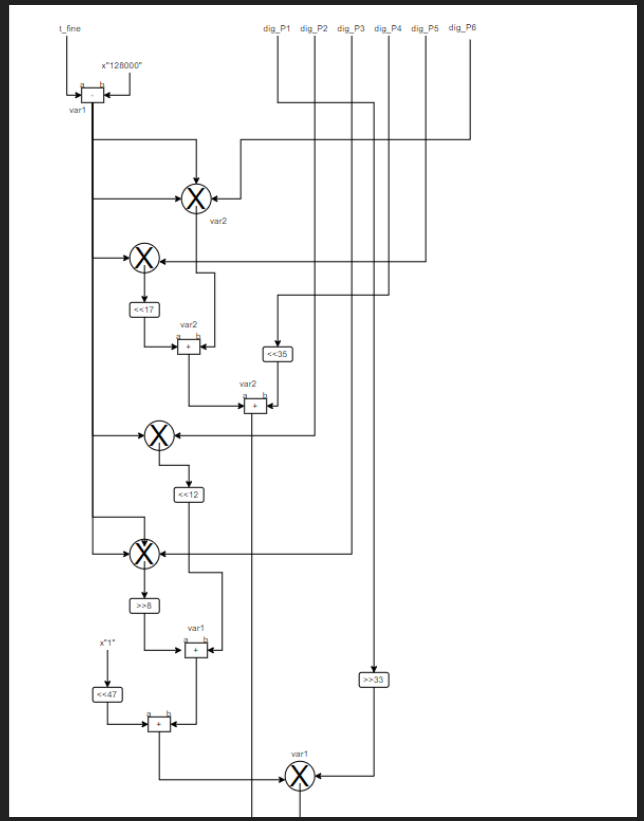


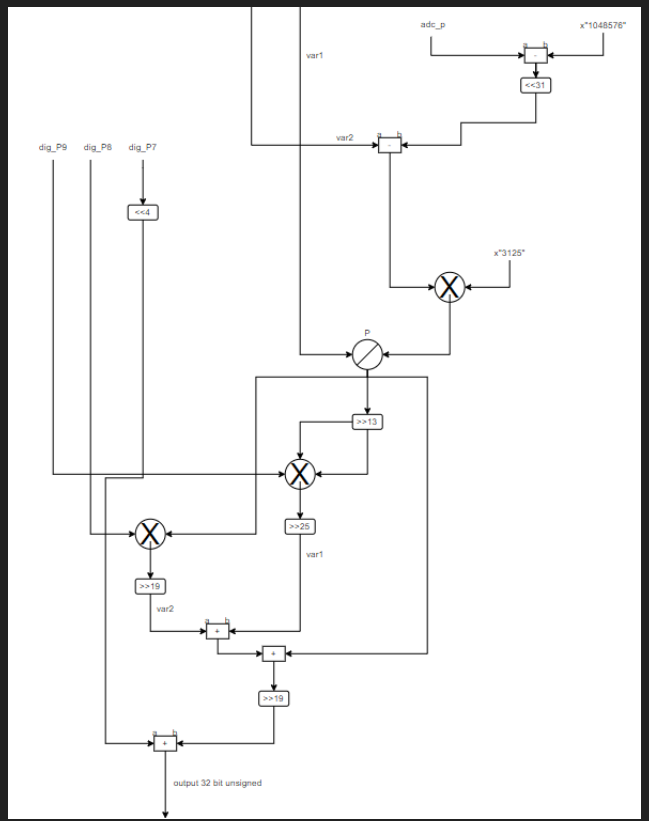
Conversion Formulas from Datasheet (2/3)

Pressure

```
// Returns pressure in Pa as unsigned 32 bit integer in Q24.8 format (24 integer bits and 8
fractional bits).
// Output value of "24674867" represents 24674867/256 = 96386.2 Pa = 963.862 hPa
BME280_U32_t BME280_compensate_P_int64(BME280_S32_t adc_P)
{
   BME280_S64_t var1, var2, p;
   var1 = ((BME280_S64_t)t_fine) - 128000;
   var2 = var1 * var1 * (BME280_S64_t)dig_P6;
   var2 = var2 + ((var1*(BME280_S64_t)dig_P5)<<17);
   var2 = var2 + (((BME280_S64_t)dig_P4)<<35);
   var1 = ((var1 * var1 * (BME280_S64_t)dig_P3)>>8) + ((var1 * (BME280_S64_t)dig_P2)<<12);
   var1 = (((((BME280_S64_t)1)<<47)+var1))*((BME280_S64_t)dig_P1)>>33;
   if (var1 == 0)
   {
      return 0; // avoid exception caused by division by zero
   }
   p = 1048576-adc_P;
   p = (((p<<31)-var2)*3125)/var1;
   var1 = (((BME280_S64_t)dig_P9) * (p>>13) * (p>>13)) >> 25;
   var2 = (((BME280_S64_t)dig_P8) * p) >> 19;
   p = ((p + var1 + var2) >> 8) + (((BME280_S64_t)dig_P7)<<4);
   return (BME280_U32_t)p;
}
```

Pressure Conversion Digital Circuit





Conversion Formulas from Datasheet (3/3)

Humidity

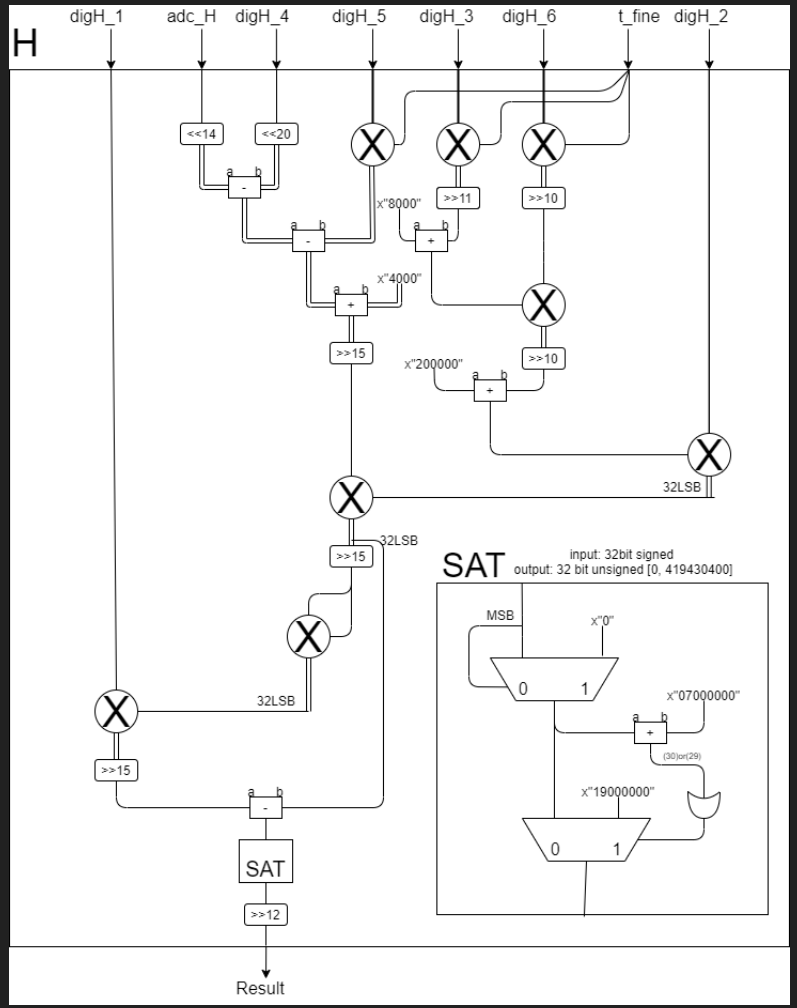
```
// Returns humidity in %RH as unsigned 32 bit integer in Q22.10 format (22 integer and 10
fractional bits).
// Output value of "47445" represents 47445/1024 = 46.333 %RH
BME280_U32_t bme280_compensate_H_int32(BME280_S32_t adc_H)
{
   BME280_S32_t v_x1_u32r;

   v_x1_u32r = (t_fine - ((BME280_S32_t)76800));

   v_x1_u32r = (((((adc_H << 14) - (((BME280_S32_t)dig_H4) << 20) - (((BME280_S32_t)dig_H5) *
      v_x1_u32r)) + ((BME280_S32_t)16384)) >> 15) * (((((((v_x1_u32r *
      ((BME280_S32_t)dig_H6)) >> 10) * (((v_x1_u32r * ((BME280_S32_t)dig_H3)) >> 11) +
      ((BME280_S32_t)32768))) >> 10) + ((BME280_S32_t)2097152)) * ((BME280_S32_t)dig_H2) +
      8192) >> 14));
   v_x1_u32r = (v_x1_u32r - (((((v_x1_u32r >> 15) * (v_x1_u32r >> 15)) >> 7) *
      ((BME280_S32_t)dig_H1)) >> 4));
   v_x1_u32r = (v_x1_u32r < 0 ? 0 : v_x1_u32r);
   v_x1_u32r = (v_x1_u32r > 419430400 ? 419430400 : v_x1_u32r);
   return (BME280_U32_t)(v_x1_u32r>>12);
}
```

Humidity Conversion Digital Circuit

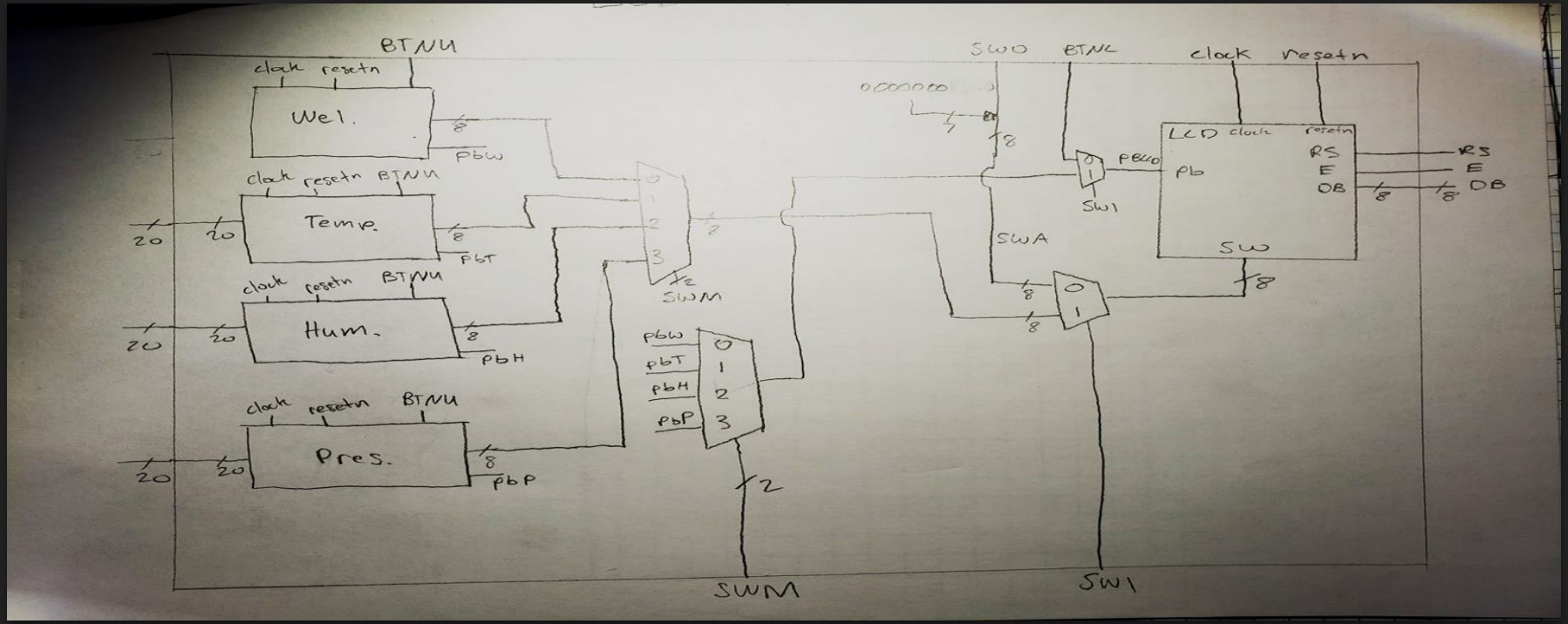
Due to the number of signed multipliers in this circuit, the bus width became too large.

To correct this, inputs to the signed multipliers were restricted to 32 bits, while outputs are restricted to 64 bits, shown as double line.

Outputs of the circuit are saturated at the end, so no loss of data will take place.

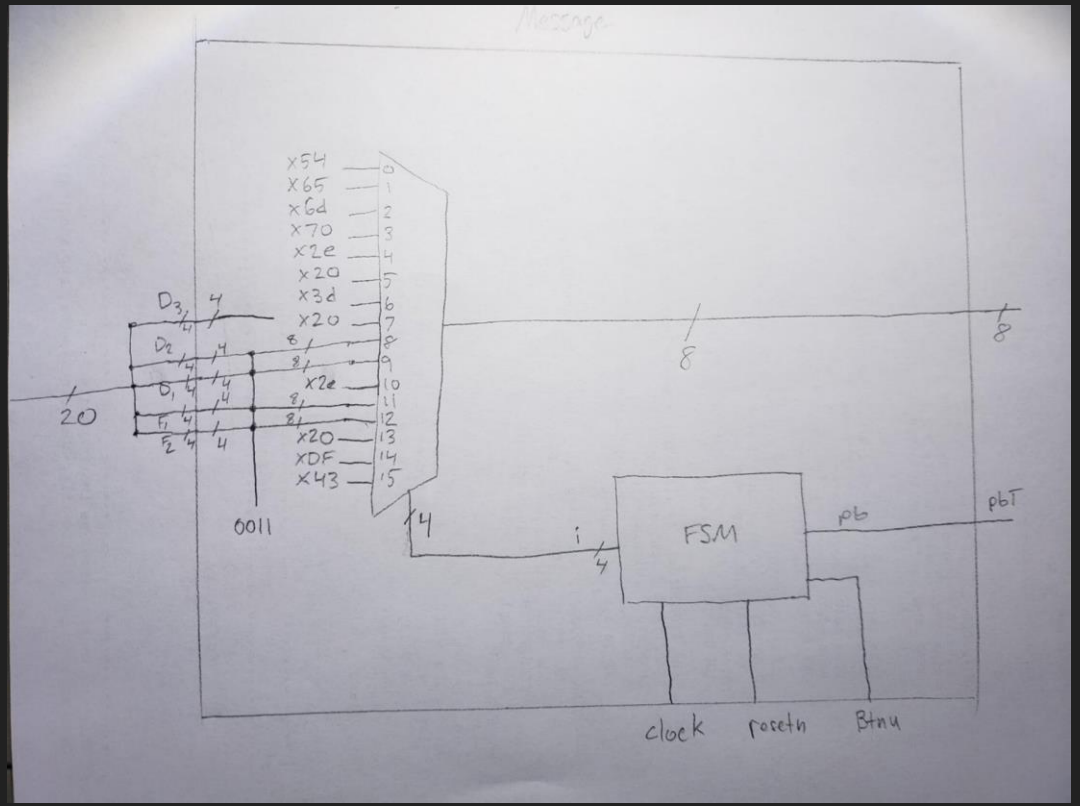


LCD Controller



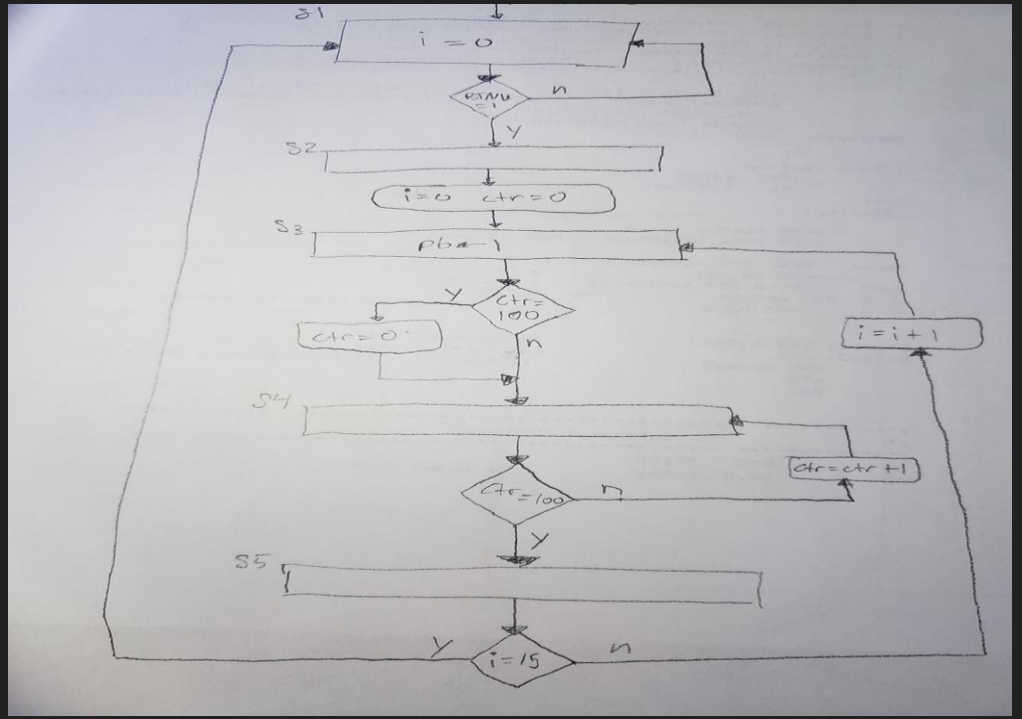
LCD Message

- 20-bit BCD input (from sensor)
- 1 16to1 Mux
  - 4 or 5 channel resolution
- 8-bit output to LCD
- 1 write signal



LCD Message FSM

- 2 Internal Counters (i,ctr)
  - I is for cycling through mux
  - Ctr is for giving enough write time for LCD to display
- 1 Start Signal (BTNU)
  - Wait for button press
- 1 Output Signal (pb)
  - Enable write



Demo

Challenges

The project as a whole was the most challenging part, it was much more than we had anticipated before making purchases and delving into the datasheet.

If we had to pick one thing out it would be the conversion circuits, which were designed based on code that would have been more suited for a microcontroller.

Improvements

We could be able to improve in the conversion of raw data. Since the bit sizes started becoming too high to handle there was a lot of dropping MSB’s. Ideally we wouldn’t want to do that so our numbers can become more exact.

References

[1] BME280 - Bosch Sensortec - Humidity, Moisture | Online Catalog

[2] https://www.dfrobot.com/product-1606.html?gclid=EAIaIQobChMIrL6F4Jui6AIVWv7jBx26BwmrEAkYCiABEgJxAvD_BwE

[3] BME280 I²C or SPI Temperature Humidity Pressure Board - Adafruit Industries LLC - Sensors | Online Catalog