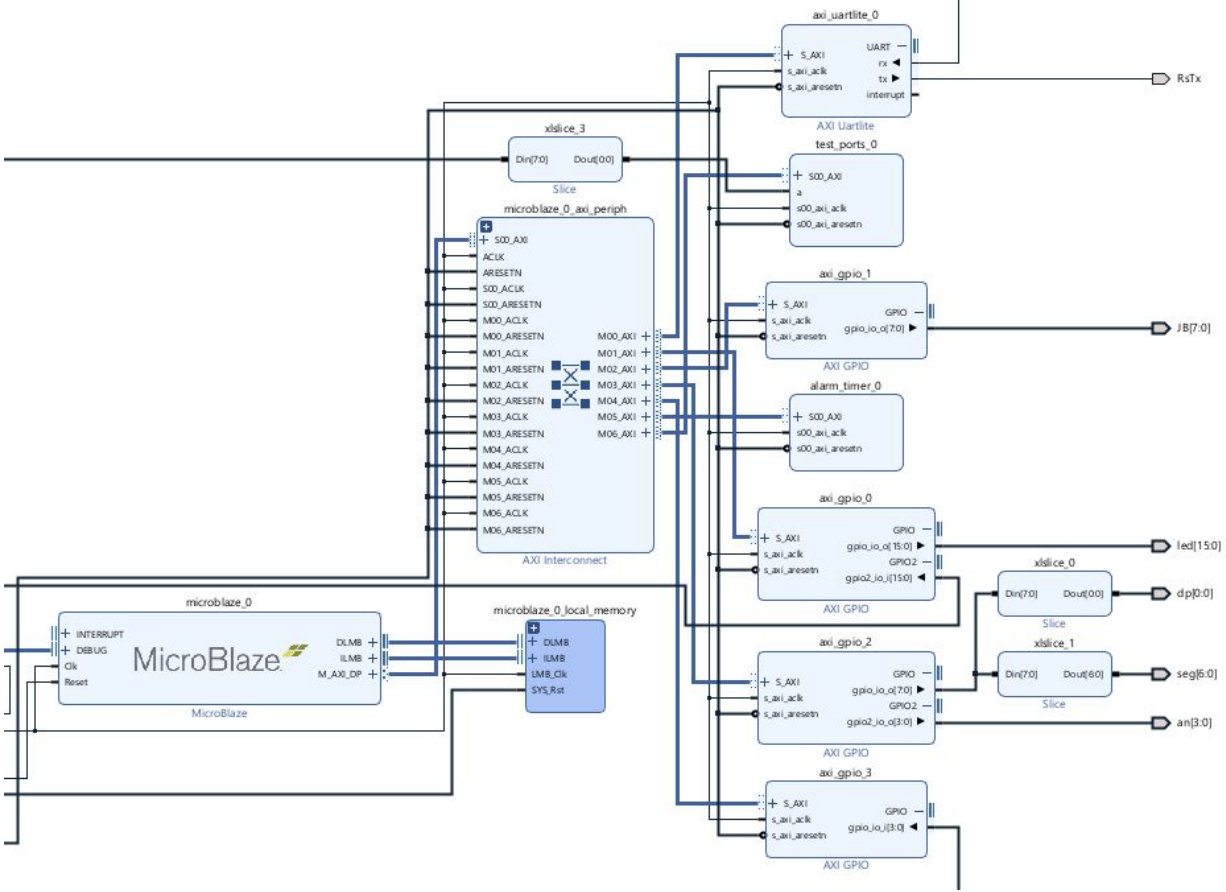
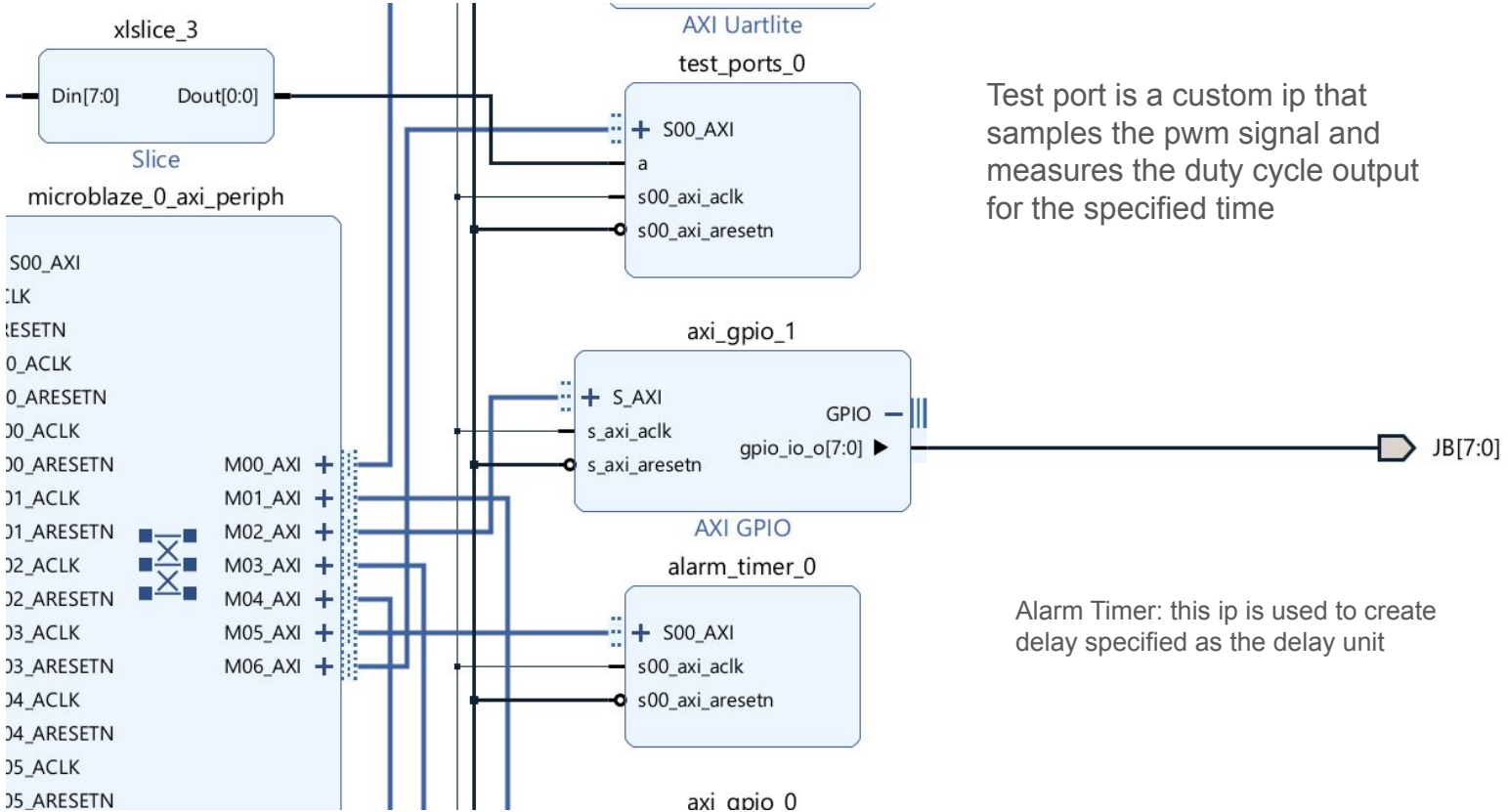

axi_uartlite_0
S_AXI
s_axi_aclk
s_axi_aresetn
UART
rx
tx
interrupt
AXI Uartlite
RsTx
xlslice_3
Din[7:0]
Dout[0:0]
Slice
test_ports_0
S00_AXI
a
s00_axi_aclk
s00_axi_aresetn
microblaze_0_axi_periph
S00_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
M01_ACLK
M01_ARESETN
M02_ACLK
M02_ARESETN
M03_ACLK
M03_ARESETN
M04_ACLK
M04_ARESETN
M05_ACLK
M05_ARESETN
M06_ACLK
M06_ARESETN
M00_AXI
M01_AXI
M02_AXI
M03_AXI
M04_AXI
M05_AXI
M06_AXI
AXI Interconnect
axi_gpio_1
GPIO
gpio_io_o[7:0]
AXI GPIO
JB[7:0]
alarm_timer_0
axi_gpio_0
GPIO
gpio_io_o[15:0]
GPIO2
gpio2_io_i[15:0]
led[15:0]
xlslice_0
dp[0:0]
microblaze_0
INTERRUPT
DEBUG
Clk
Reset
MicroBlaze
DLMB
ILMB
M_AXI_DP
microblaze_0_local_memory
LMB_Clk
SYS_Rst
axi_gpio_2
gpio_io_o[7:0]
gpio2_io_o[3:0]
xlslice_1
seg[6:0]
an[3:0]
axi_gpio_3
gpio_io_i[3:0]



Test port is a custom ip that samples the pwm signal and measures the duty cycle output for the specified time

Alarm Timer: this ip is used to create delay specified as the delay unit

Custom ip diagrams

```
#include "xil_printf.h"

#define ONE_US 100 // 10ns * 100
#define ONE_MS 100*1000 // 1us * 100
#define INCH_CONST 1

#define ALARM_CNTR         (* (volatile unsigned *)0x44a00000)
#define ALARM1       (* (volatile unsigned *)0x44a00005)
#define ALARM0_VALUE       (* (volatile unsigned *)0x44a00008)
#define ALT_CNTR   (* (volatile unsigned *)0x44a10000)

#define DELAY_UNIT 81

#define LEDS   (*(unsigned volatile *)0x40000000 ) //GPIO-0 16-bit
#define SW     (*(unsigned volatile *)0x40000008 ) //GPIO-0 16-bit
#define JB     (*(unsigned volatile *)0x40010000 ) //GPIO-1 8-bit
#define DPSEG  (*(unsigned volatile *)0x40020000 ) //GPIO-2 {DP,
SEG[6:0]}
#define AN     (*(unsigned volatile *)0x40020008 ) //GPIO-2 4-bit
#define BTN    (*(unsigned volatile *)0x40030000 ) //GPIO-3 4-bit,
{btnR, btnL, btnD, btnU};

void delay_ms(unsigned t){
unsigned cntr1, cntr2;
while(t--)
for (cntr1 = 0; cntr1<100; cntr1++){
for (cntr2 = 0; cntr2<DELAY_UNIT; cntr2++){}
}
}
```

First we define all the variables and enter their address from the vivado=>block design=>address editor

```
void delay_ms2 (uint32_t val){
while ((ALARM1 & (1<<0)) == 0){}
ALARM0_VALUE = 100000000; //set alarm offset to loop time
}
```

Create a delay period using the alarm ip

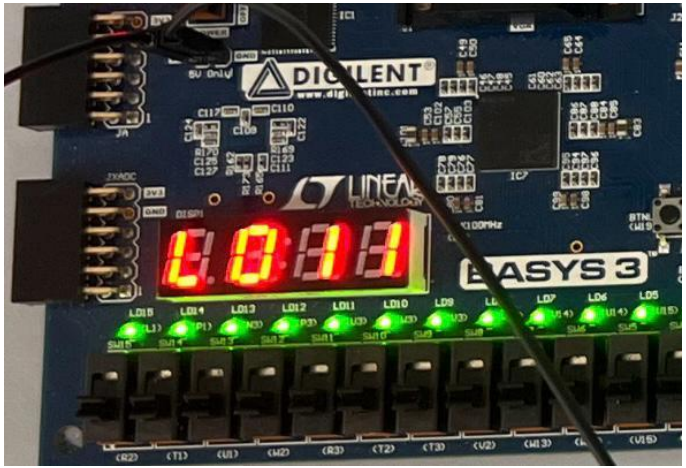
```
void seg_disp(uint8_t data[4]){
const uint8_t disp_lut[16] = {
0b00111111, //0
0b00000110, //1
0b01011011, //2
0b01001111, //3
0b01100110, //4
0b01101101, //5
0b01111101, //6
0b00000111, //7
0b01111111, //8
0b01101111, //9
0b01110111, //10
0b01101101,//0b01111100, //11 small metal
0b00110111,//0b00111001,//12 med metal
0b00111000,//0b01011110,//13 large  metal
0b01111001,//14
0b01110001,//15
};
```

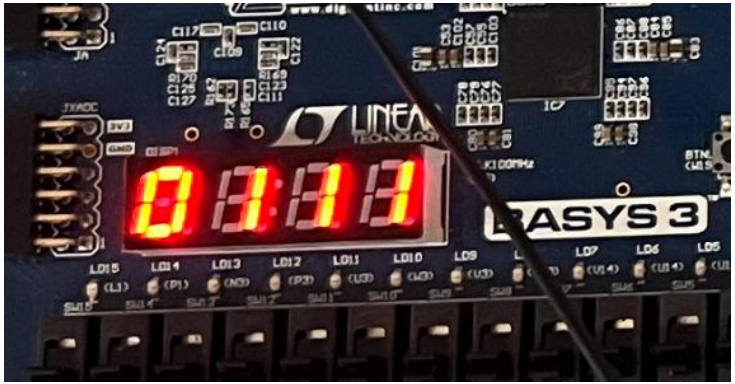
Write a lookup table for the reference of the seven segment display module.

Code for the seven segment display

```
static uint8_t digit = 0;

AN = ~(1<<(3-digit));
DPSEG = ~disp_lut[data[digit]];

if(digit == 3){
digit = 0;
}
else{
digit++;
}

}
```

This code snippet is used to intialize the anodes and the seven segment LED modules





```
int main (){
uint8_t data[4];
uint8_t left = 0;
uint8_t right = 0;
uint8_t leftmiddle = 0;
uint8_t rightmiddle = 0;
int32_t metal_ctr = 0;
int32_t count = 0;
int32_t prev_count = 0;
int32_t difference = 0;
uint8_t speed = 0;
_Bool display;
int32_t metal;
int32_t metal_p=0;
int32_t small = 0;
int32_t med = 0;
int32_t large = 0;
int32_t small_p = 0;
int32_t med_p = 0;
int32_t large_p = 0;
int32_t small_count = 0;
int32_t med_count = 0;
int32_t large_count = 0;
print("final Launched!\n\r");
```

First, we initialize all the variables

```
while(1){
   delay_ms(1);
   LEDS &= ~(1<<0);
   LEDS &= ~(1<<1);
   LEDS &= ~(1<<2);
   LEDS &= ~(1<<3);
   LEDS &= ~(1<<4);
   LEDS &= ~(1<<5);
   LEDS &= ~(1<<6);
   LEDS &= ~(1<<7);
   LEDS &= ~(1<<8);
   LEDS &= ~(1<<9);
   LEDS &= ~(1<<10);
   LEDS &= ~(1<<11);
   LEDS &= ~(1<<12);
   LEDS &= ~(1<<13);
   LEDS &= ~(1<<14);
   LEDS &= ~(1<<15);
   if(speed == 100){
   count = ALT_CNTR;
   difference = count - prev_count;
   prev_count = count;
   xil_printf("%d\n",difference);
   speed =0;
   }
   speed++;
```

We turn off all the LEDs at the start

This is the code for sampling the value of duty cycle of the pwm signal for every 100 ms

//led driver

Large metal

```
if(difference>=9935596)
{
   LEDS |= (1<<0);
   LEDS |= (1<<1);
   LEDS |= (1<<2);
   LEDS |= (1<<3);
   LEDS |= (1<<4);
   LEDS |= (1<<5);
   LEDS |= (1<<6);
   LEDS |= (1<<7);
   LEDS |= (1<<8);
   LEDS |= (1<<9);
   LEDS |= (1<<10);
   LEDS |= (1<<11);
   LEDS |= (1<<12);
   LEDS |= (1<<13);
   LEDS |= (1<<14);
   LEDS |= (1<<15);
   metal= 1;
   large = 1;
     }
```

Proximity sensor for all the metal and this code also raises a flag for metal,large, medium, and small metal

Medium metal

```
else
if(difference>=8478275)
{
   LEDS |= (1<<0);
   LEDS |= (1<<1);
   LEDS |= (1<<2);
   LEDS |= (1<<3);
   LEDS |= (1<<4);
   LEDS |= (1<<5);
   LEDS |= (1<<6);
   LEDS |= (1<<7);
   LEDS |= (1<<8);
   LEDS |= (1<<9);
   LEDS |= (1<<10);
   metal= 1;
   med = 1;
```

Small metal

```
else
if(difference>6700000){
      LEDS |= (1<<0);
      LEDS |= (1<<1);
      LEDS |= (1<<2);
      metal= 1;
      small = 1;
```

No metal

```
else
if(difference>=6540000){
      LEDS |= (1<<0);
      metal= 0;
      small =0;
      med = 0;
      large =0;

            }
   else {
   metal =0;
   small =0;
   med = 0;
   large =0;
   }
```

```
//total metal counter
   if(metal_p == 1){
   if(metal == 0){
   metal_ctr++;

   }

   }
   metal_p= metal;

//small metal counter
      if(small_p == 1){
      if(small == 0){
      small_count++;

      }

      }
      small_p= small;

//med metal counter
         if(med_p == 1){
         if(med == 0){
         med_count++;
         small_count--;

         }

         }
         med_p= med;
```

```
//large metal counter
         if(large_p == 1){
            if(large == 0){
            large_count++;
            med_count--;


            }

            }
            large_p= large;
```

This nested if statements uses the flags that are raised in the previous step to increment the metal counter.

```
//total display
   if(SW & (1<<0)){

   left = (metal_ctr / 1000) % 10;
   leftmiddle = (metal_ctr / 100) % 10;
   rightmiddle = (metal_ctr / 10) % 10;
   right = metal_ctr % 10;
   }
  else if(SW & (1<<15)){

         left = (difference / 1000000) % 10;
         leftmiddle = (difference / 100000) % 10;
         rightmiddle = (difference  / 10000) % 10;
         right = (difference  / 1000) % 10;
   }
else{
   if(large==1){
   Left = 13;
   }
   else if(med==1){
   left = 12;
   }
   else if(small==1){
   Left = 11;
   }
```

```
else{
   Left = 0;
   }
   leftmiddle = large_count % 10;
   rightmiddle = med_count % 10;
   right = small_count % 10;
   }

 //display data
   data[3] = right;
   data[2] = rightmiddle;
   data[1] = leftmiddle;
   data[0] = left;
   seg_disp(data);

   }
```

The first if else loop shows the total value of the metal counter which is total no of
the sw0 is turned on and the next else if loop is used to show the value of the duty
credits) when sw15 is turned on and finally if all the switches are turned off the da
symbol of the metal being detected(s,n or L) and the data(1) for large metal coun
metal counter and data(4) for small metal counter