

**San Jose State University  
Department of Electrical Engineering**

**Laboratory Assignment #2**

**WITH SIMD CODE :**

/\*\*\*\*\*

\*

\* Copyright (C) 2009 - 2014 Xilinx, Inc. All rights reserved.

\*

\* Permission is hereby granted, free of charge, to any person obtaining a copy

\* of this software and associated documentation files (the "Software"), to deal

\* in the Software without restriction, including without limitation the rights

\* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

\* copies of the Software, and to permit persons to whom the Software is

\* furnished to do so, subject to the following conditions:

\*

\* The above copyright notice and this permission notice shall be included in

\* all copies or substantial portions of the Software.

\*

\* Use of the Software is limited solely to applications:

\* (a) running on a Xilinx device, or

\* (b) that interact with a Xilinx device through a bus or interconnect.

\*

\* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

\* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

\* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL

\* XILINX BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,

\* WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF

\* OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

\* SOFTWARE.

\*

\* Except as contained in this notice, the name of the Xilinx shall not be used

\* in advertising or otherwise to promote the sale, use or other dealings in

\* this Software without prior written authorization from Xilinx.

\*

\*\*\*\*\*/

/\*

\* helloworld.c: simple test application

\*

\* This application configures UART 16550 to baud rate 9600.

\* PS7 UART (Zynq) is not initialized by this application, since

\* bootrom/bsp configures it to baud rate 115200

\*

\* -----

\* | UART TYPE BAUD RATE |

\* -----

\* uartns550 9600

\* uartlite Configurable only in HW design

\* ps7\_uart 115200 (configured by bootrom/bsp)

\*/

#include <stdio.h>

#include <stdlib.h>

#include <arm\_neon.h>

#include "platform.h"

#include "xil\_printf.h"

#include "xtime\_l.h"

#define ROWS 144

#define COLS 176

```

static unsigned char image1[ROWS][COLS] __attribute__((aligned(16)));
static unsigned char image2[ROWS][COLS] __attribute__((aligned(16)));

int sad(const unsigned char *im1_p, const unsigned char *im2_p, int numcols);

int sad(const unsigned char *im1_p, const unsigned char *im2_p, int numcols) {
    XTime begin_time, end_time;

    static unsigned int someones_an_idiot;

    int safety_count = 2;

    XTime_GetTime(&begin_time); /* operation with SIMD will start here */
    if (im1_p == NULL) {
        safety_count--;
    }
    if (im2_p == NULL) {
        safety_count--;
    }
    if (safety_count != 2) {
        someones_an_idiot++;
    }

    int i; /* defining i for while loop */

    uint16_t Sum_of_absolute_differences = 0; /* using 16 bit reg defining
    Sum_of_absolute_differences for final result */

    i = 0; /* initialize i to 0 for start of operation */
    uint8x16_t absolute; /* using 128 bit reg defining absolute */
    while(i<16){ /* start of while loop for SAD operation */
        uint8x16_t Load__Image1, Load__Image2; /* defining 128 bit reg for images */
        Load__Image1 = vld1q_u8 (im1_p); /* loading rows to the reg */

```

```

        Load__Image2 = vld1q_u8 (im2_p); /* loading cols to the reg */

/* Load our custom data into the vector register. */
        absolute = vabdq_u8(Load__Image1, Load__Image2); /* getting absolute value of
two images */

        /* final sum of all absolutes */

        Sum_of_absolute_differences+=absolute[0]+absolute[1]+absolute[2]+absolute[3]+absolute[
4]+absolute[5]+absolute[6]+absolute[7]+absolute[8]+absolute[9]+absolute[10]+absolute[11]+absolu
te[12]+absolute[13]+absolute[14]+absolute[15];

        im1_p = im1_p + 160 + 16; /* Going to column 0 */
        im2_p = im2_p + 160 + 16; /* Going to next row */
        i++; /* incrementing i for next operation */
    }

    XTime_GetTime(&end_time); /* End of SIMD operation */

    //uses the global timer in the Zynq SoC whose counter increases every two clock cycles.
    printf("Output took %llu clock cycles.\n", 2*(end_time - begin_time));

    printf("Output took %.2f us.\n", 1.0 * (end_time - begin_time) /
(COUNTS_PER_SECOND/1000000));

    return(Sum_of_absolute_differences);
}

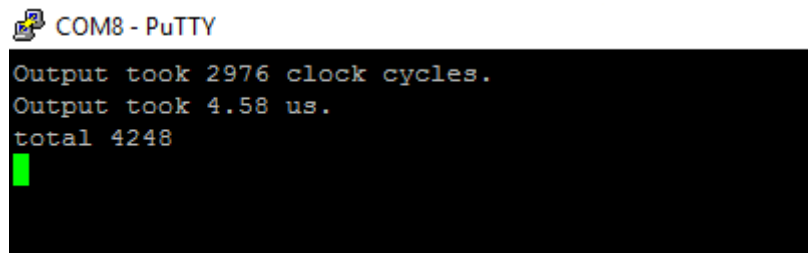
int main(int argc, char *argv[]) {

    unsigned int total;
    unsigned char *im1_p, *im2_p;
    int row, col;
    int block_row, block_col;

    /* initialize source data (and warm up caches) */

```

```
for (row = 0; row < ROWS; row++) {  
    im1_p = image1[row]; /* point to first pixel in row */  
    im2_p = image2[row];  
    for (col = 0; col < COLS; col++) {  
        unsigned char temp;  
  
        temp = ((row+col+120) % 256); /* sort of a random number */  
        *im1_p++ = temp;  
        *im2_p++ = 255-temp;  
    } /* column loop */  
} /* row loop */  
  
block_row = 0;  
block_col = 0;  
  
/* point to first pixel in each block */  
im1_p = &image1[16*block_row][16*block_col];  
im2_p = &image2[16*block_row][16*block_col];  
  
total = sad(im1_p, im2_p, COLS);  
  
/* total == 4248 */  
printf("total %d\n", total);  
return(0);  
} /* end of main */
```

**SCREESHOT OF THE OUTPUT AT PUTTY :**

COM8 - PuTTY

```
Output took 2976 clock cycles.  
Output took 4.58 us.  
total 4248  
█
```

**FINAL OUPUTU TABLE FOR WITH & WITHOUT SIMD RESULTS:**

SR. NUMBER	TYPE	RESULT	CLOCK CYCLES	TIME (us.)
1	WITHOUT SIMD	4248	6162	9.48
2	WITH SIMD	4248	2976	4.58