ELD Lab 8 HomeWork

Aditya Gautam

2023043

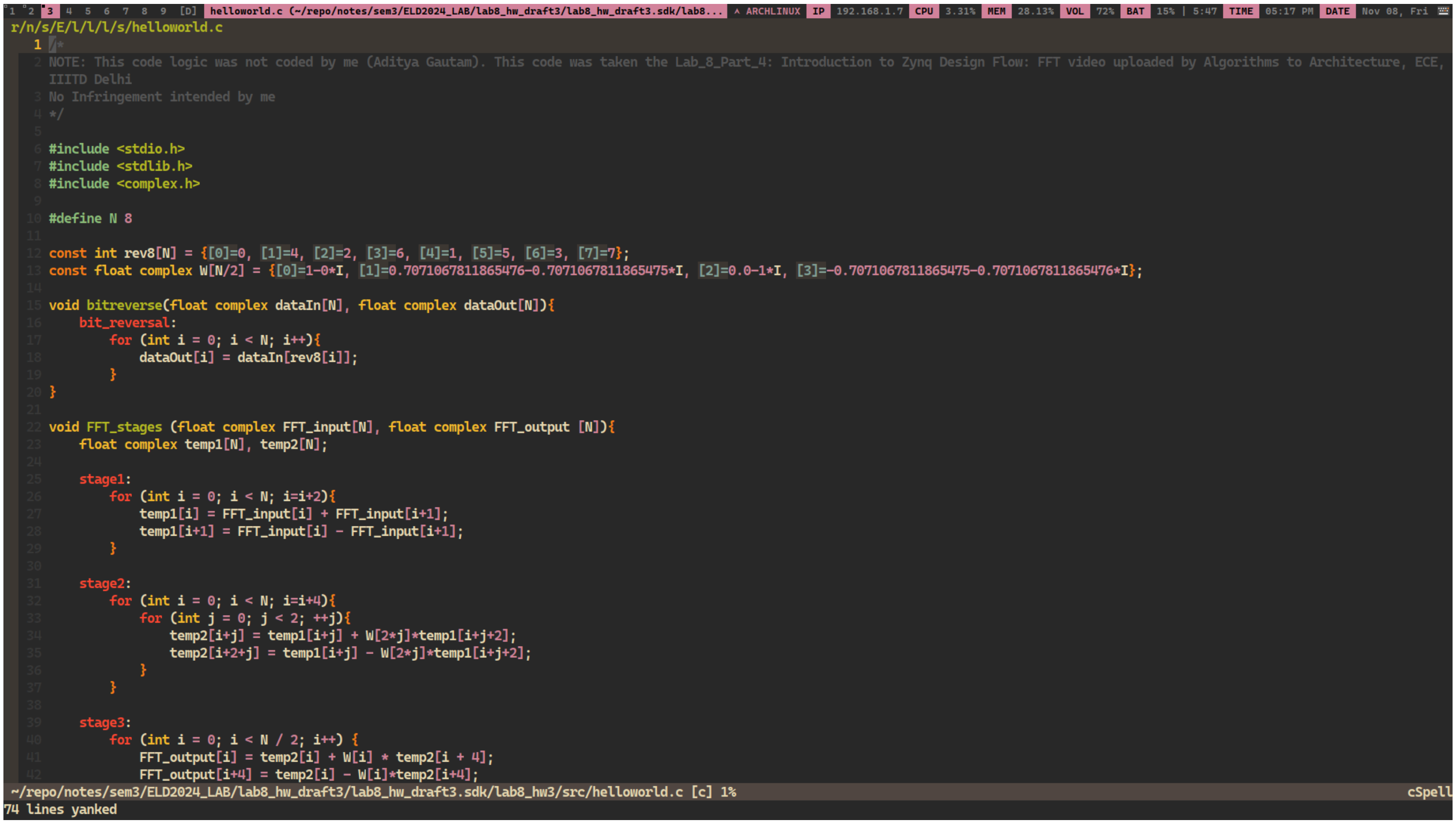
Demo Video

https://drive.google.com/file/d/1TPh79x7N6YC14X4DZXXVuKRcQTkj_jBU/view?usp=sharing

> Note: Make sure to change the video quality to the highest since it is not set to that by default

Source Code

```
/*
NOTE: This code logic was not coded by me (Aditya Gautam). This code was taken the
Lab_8_Part_4: Introduction to Zynq Design Flow: FFT video uploaded by Algorithms to
Architecture, ECE, IIITD Delhi
No Infringement intended by me
*/

#include <stdio.h>
#include <stdlib.h>
#include <complex.h>

#define N 8

const int rev8[N] = {0, 4, 2, 6, 1, 5, 3, 7};
const float complex W[N/2] = {1-0*I, 0.7071067811865476-0.7071067811865475*I, 0.0-1*I,
-0.7071067811865475-0.7071067811865476*I};

void bitreverse(float complex dataIn[N], float complex dataOut[N]){
      bit_reversal:
      for (int i = 0; i < N; i++){
            dataOut[i] = dataIn[rev8[i]];
      }
}

void FFT_stages (float complex FFT_input[N], float complex FFT_output [N]){
      float complex temp1[N], temp2[N];

      stage1:
      for (int i = 0; i < N; i=i+2){
            temp1[i] = FFT_input[i] + FFT_input[i+1];
            temp1[i+1] = FFT_input[i] - FFT_input[i+1];
      }

      stage2:
      for (int i = 0; i < N; i=i+4){
            for (int j = 0; j < 2; ++j){
            temp2[i+j] = temp1[i+j] + W[2*j]*temp1[i+j+2];
            temp2[i+2+j] = temp1[i+j] - W[2*j]*temp1[i+j+2];
            }
      }

      stage3:
      for (int i = 0; i < N / 2; i++) {
            FFT_output[i] = temp2[i] + W[i] * temp2[i + 4];
            FFT_output[i+4] = temp2[i] - W[i]*temp2[i+4];
      }
}

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

  // note: these "I" are provided by complex.h to represent Iota
  const float complex FFT_input[N] = {
      11 + 23 * I,
      32 + 10 * I,
      91 + 94 * I,
      15 + 69 * I,
      47 + 96 * I,
      44 + 12 * I,
      96 + 17 * I,
      49 + 58 * I,
      };

  float complex FFT_output[N];
  float complex FFT_rev[N];

  bitreverse(FFT_input, FFT_rev);
  FFT_stages(FFT_rev, FFT_output);

  printf("\nInput: \n");
  for (int i = 0; i < N; i++) {
      printf("%f %f\n", crealf(FFT_input[i]), cimagf(FFT_input[i]));
  }
  printf("\nOutput: \n");
  for (int i = 0; i < N; i++) {
      printf("%f %f\n", crealf(FFT_output[i]), cimagf(FFT_output[i]));
  }
}
```



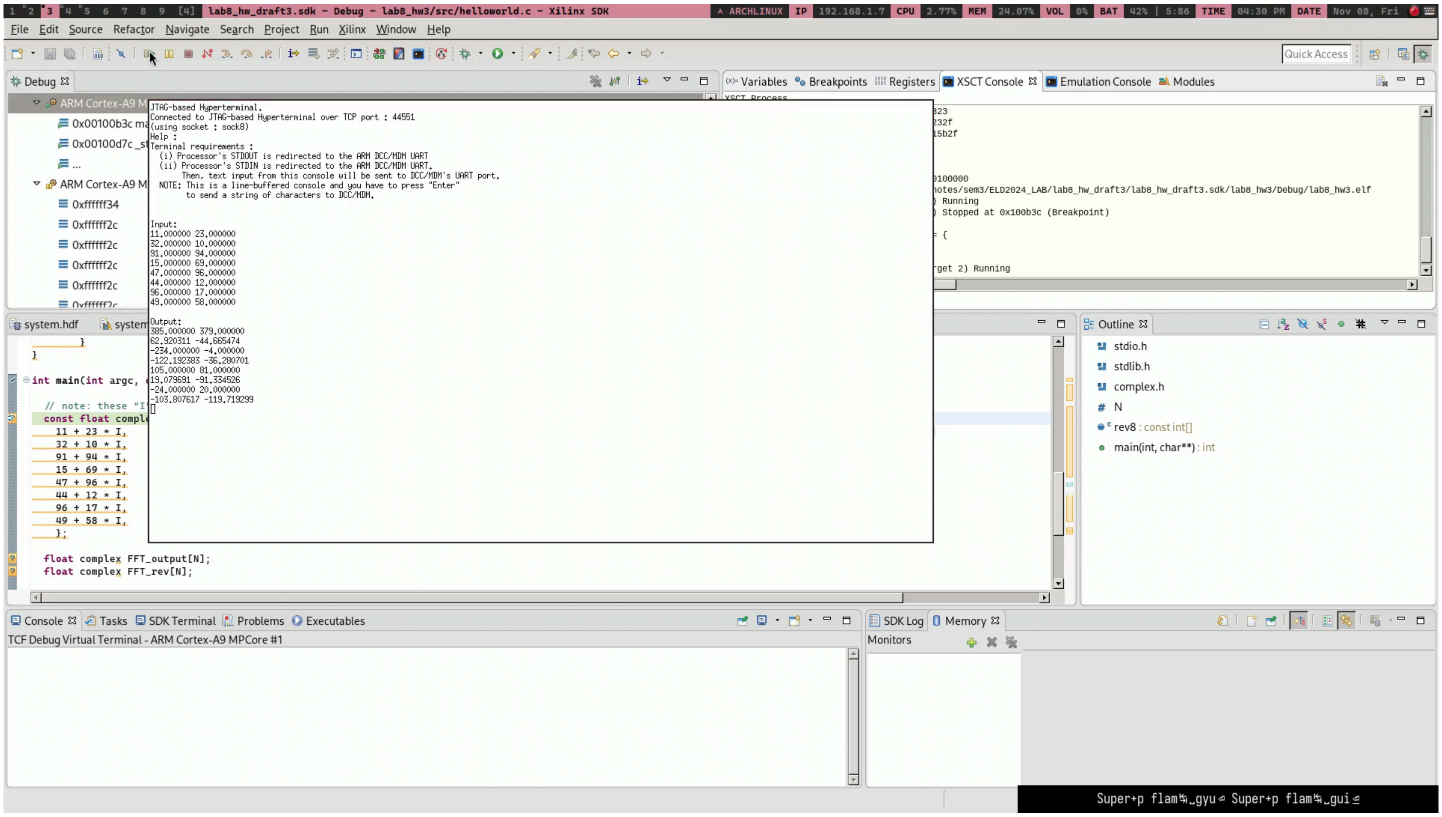
1 2 3 4 5 6 7 8 9 [D] helloworld.c (~/repo/notes/sem3/ELD2024_LAB/lab8_hw_draft3/lab8_hw_draft3.sdk/lab8... ARCHLINUX IP 192.168.1.7 CPU 3.31% MEM 28.13% VOL 72% BAT 15% | 5:47 TIME 05:17 PM DATE Nov 08, Fri
r/n/s/E/l/l/l/s/helloworld.c

```c
/*
NOTE: This code logic was not coded by me (Aditya Gautam). This code was taken the Lab_8_Part_4: Introduction to Zynq Design Flow: FFT video uploaded by Algorithms to Architecture, ECE, IIITD Delhi
No Infringement intended by me
*/

#include <stdio.h>
#include <stdlib.h>
#include <complex.h>

#define N 8

const int rev8[N] = {[0]=0, [1]=4, [2]=2, [3]=6, [4]=1, [5]=5, [6]=3, [7]=7};
const float complex W[N/2] = {[0]=1-0*I, [1]=0.7071067811865476-0.7071067811865475*I, [2]=0.0-1*I, [3]=-0.7071067811865475-0.7071067811865476*I};

void bitreverse(float complex dataIn[N], float complex dataOut[N]){
    bit_reversal:
        for (int i = 0; i < N; i++){
            dataOut[i] = dataIn[rev8[i]];
        }
}

void FFT_stages (float complex FFT_input[N], float complex FFT_output [N]){
    float complex temp1[N], temp2[N];

    stage1:
        for (int i = 0; i < N; i=i+2){
            temp1[i] = FFT_input[i] + FFT_input[i+1];
            temp1[i+1] = FFT_input[i] - FFT_input[i+1];
        }

    stage2:
        for (int i = 0; i < N; i=i+4){
            for (int j = 0; j < 2; ++j){
                temp2[i+j] = temp1[i+j] + W[2*j]*temp1[i+j+2];
                temp2[i+2+j] = temp1[i+j] - W[2*j]*temp1[i+j+2];
            }
        }

    stage3:
        for (int i = 0; i < N / 2; i++) {
            FFT_output[i] = temp2[i] + W[i] * temp2[i + 4];
            FFT_output[i+4] = temp2[i] - W[i]*temp2[i+4];
```

~/repo/notes/sem3/ELD2024_LAB/lab8_hw_draft3/lab8_hw_draft3.sdk/lab8_hw3/src/helloworld.c [c] 1%

```
74 lines yanked
```

```
/* IMPORTANT NOTE
 * I am not the original author of the code presented in this pdf
 * It has been copied by me from the YouTube video linked by Prof. Sumit Sir for us to refer
 * to for lab 8. I do not claim ownership of this code.
/*
```



1 2 3 4 5 6 7 8 9 [4] lab8_hw_draft3.sdk - Debug - lab8_hw3/src/helloworld.c - Xilinx SDK ARCHLINUX IP 192.168.1.7 CPU 2.77% MEM 24.07% VOL 0% BAT 42% | 5:86 TIME 04:30 PM DATE Nov 08, Fri
File Edit Source Refactor Navigate Search Project Run Xilinx Window Help

Variables Breakpoints Registers XSCT Console Emulation Console Modules

```
JTAG-based Hyperterminal.
Connected to JTAG-based Hyperterminal over TCP port : 44551
(using socket : sock8)
Help :
Terminal requirements :
  (i) Processor's STDOUT is redirected to the ARM DCC/MDM UART
  (ii) Processor's STDIN is redirected to the ARM DCC/MDM UART.
       Then, text input from this console will be sent to DCC/MDM's UART port.
  NOTE: This is a line-buffered console and you have to press "Enter"
        to send a string of characters to DCC/MDM.


Input:
11.000000 23.000000
32.000000 10.000000
91.000000 94.000000
15.000000 69.000000
47.000000 96.000000
44.000000 12.000000
96.000000 17.000000
49.000000 58.000000

Output:
385.000000 379.000000
62.920311 -44.665474
-234.000000 -4.000000
-122.192383 -36.280701
105.000000 81.000000
19.079691 -91.334526
-24.000000 20.000000
-103.807617 -119.719299
```

```
323
232f
15b2f



0100000
notes/sem3/ELD2024_LAB/lab8_hw_draft3/lab8_hw_draft3.sdk/lab8_hw3/Debug/lab8_hw3.elf
) Running
) Stopped at 0x100b3c (Breakpoint)

= {


rget 2) Running
```

```c
            }
}

int main(int argc, c

  // note: these "I
  const float comple
      11 + 23 * I,
      32 + 10 * I,
      91 + 94 * I,
      15 + 69 * I,
      47 + 96 * I,
      44 + 12 * I,
      96 + 17 * I,
      49 + 58 * I,
      };

  float complex FFT_output[N];
  float complex FFT_rev[N];
```

Input:

```
11 + 23j
32 + 10j
91 + 94j
15 + 69j
47 + 96j
44 + 12j
96 + 17j
49 + 58j
```

Output:

```
385 + 379j
62.92 - 44.66j
-234 - 4j
-122.19 - 36.28j
105 + 81j
19.07 - 91.33j
-24 + 20j
-103.8 - 119.71j
```