INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Department
of
Electronics & Communication Engineering

Embedded Logic Design
Lab 9 Submission

SHIVAM SHUKLA
2022478

Source Code

8 Point FFT:

Lab9.sdk - Debug - Lab9/src/helloworld.c - Xilinx SDK

```c
#include <complex.h>
#include <stdio.h>
#include <stdlib.h>
#include <xtime_l.h>
#include <math.h>

#define N 8

const int rev8[N] = {0, 4, 2, 6, 1, 5, 3, 7};
const float complex W1[N] = {1 - 0 * I, 0 - 1 * I, 0 , 0, 1 - 0 * I, 0 - 1 * I, 0 , 0};
const float complex W2[N / 2] = {1 - 0 * I, (1 - 1 * I) / sqrt(2) , 0 - 1 * I , (-1 - 1 * I) / sqrt(2)};

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 < 2; i = i + 1) {
    temp2[i] = temp1[i] + W1[i] * temp1[i + 2];
    temp2[i + 2] = temp1[i] - W1[i] * temp1[i + 2];
  }
  for (int i = 4; i < 6; i = i + 1) {
    temp2[i] = temp1[i] + W1[i] * temp1[i + 2];
    temp2[i + 2] = temp1[i] - W1[i] * temp1[i + 2];
  }
stage3:
  for (int i = 0; i < N / 2; i = i + 1) {
    FFT_output[i] = temp2[i] + W2[i] * temp2[i + 4];
    FFT_output[i + 4] = temp2[i] - W2[i] * temp2[i + 4];
    // printf("i=%d, %f %f\n", i, creal(FFT_output[i]), cimagf(FFT_output[i]));
    // printf("i=%d, %f %f\n", i+4, creal(FFT_output[i+4]), cimagf(FFT_output[i+4]));
  }

//     printf("\n Printinf temp1\r\n");
// for (int i = 0; i < N; i++) {
//   printf("%f %f\n", creal(temp1[i]), cimagf(temp1[i]));
// }
//       printf("\n Printinf temp2\r\n");
// for (int i = 0; i < N; i++) {
//   printf("%f %f\n", creal(temp2[i]), cimagf(temp2[i]));
// }
```

Lab9.sdk - Debug - Lab9/src/helloworld.c - Xilinx SDK

```c
    temp2[i + 2] = temp1[i] - W1[i] * temp1[i + 2];
  }
stage3:
  for (int i = 0; i < N / 2; i = i + 1) {
    FFT_output[i] = temp2[i] + W2[i] * temp2[i + 4];
    FFT_output[i + 4] = temp2[i] - W2[i] * temp2[i + 4];
    // printf("i=%d, %f %f\n", i, creal(FFT_output[i]), cimagf(FFT_output[i]));
    // printf("i=%d, %f %f\n", i+4, creal(FFT_output[i+4]), cimagf(FFT_output[i+4]));
  }

//     printf("\n Printinf temp1\r\n");
// for (int i = 0; i < N; i++) {
//   printf("%f %f\n", creal(temp1[i]), cimagf(temp1[i]));
// }
//       printf("\n Printinf temp2\r\n");
// for (int i = 0; i < N; i++) {
//   printf("%f %f\n", creal(temp2[i]), cimagf(temp2[i]));
// }
}

int main() {
  float complex FFT_input[N] = {1,2,3,4,5,6,7,8};

  float complex FFT_output[N];

  float complex FFT_rev[N];
XTime PS_start_time, PS_end_time;
XTime_SetTime(0);
XTime_GetTime(&PS_start_time);
  bitreverse(FFT_input, FFT_rev);
  FFT_stages(FFT_rev, FFT_output);
XTime_GetTime(&PS_end_time);

  printf("\n Printinf FFT input\r\n");
  for (int i = 0; i < N; i++) {
    printf("%f %f\n", creal(FFT_input[i]), cimagf(FFT_input[i]));
  }

  printf("\n Printinf FFT output\r\n");
  for (int i = 0; i < N; i++) {
      printf("%f %f\n", creal(FFT_output[i]), cimagf(FFT_output[i]));
  }

  float time = 0;
  time = (float)1.0 * (PS_end_time-PS_start_time)/(COUNTS_PER_SECOND/1000000);
  printf("\n\rExecution time for PS in Micro-seconds: %f", time);
  return 0;
}
```

16 Point FFT:

Lab9.sdk - Debug - Lab9/src/helloworld.c - Xilinx SDK

File Edit Source Refactor Navigate Search Project Run Xilinx Window Help

system.hdf | system.mss | *helloworld.c | asm_vectors.S | _exit.c | asm_vectors.S

```c
#include <complex.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <xtime_l.h>

#define N 16

const int rev16[N] = {0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15};
const float complex W1[N] = {1 - 0 * I, 0 - 1 * I, 0 , 0, 1 - 0 * I, 0 - 1 * I, 0 , 0, 1 - 0 * I, 0 - 1 * I, 0 , 0, 1 - 0 * I, 0 - 1 * I, 0 , 0};
const float complex W2[N] = {1 - 0 * I, (1 - 1 * I) / sqrt(2) , 0 - 1 * I , -0.707107 -0.707107*I,0,0,0,0,1 - 0 * I, (1 - 1 * I) / sqrt(2) , 0 - 1 * I , -0.707107 -0.707107*I,0,0,0,0};
const float complex W3[N / 2] = {1 - 0*I, 0.923879532511 - 0.382683432365*I, 0.707106781187 - 0.707106781187*I, 0.382683432365 - 0.923879532511*I, 0 - 1*I, -0.382683432365 -0.923879532511*I,
        -0.707106781187 -0.707106781187*I, -0.923879532511 -0.382683432365*I};
// for(int i=0; i < N/2; i++){
//   W3[i] = cos((i*M_PI)/8) - sin((i*M_PI)/8)*I;
//   //printf("%f %f\n",creal(W3[i]), cimagf(W3[i]));
// }

void bitreverse(float complex dataIn[N], float complex dataOut[N]) {
bit_reversal:
  for (int i = 0; i < N; i++) {
    dataOut[i] = dataIn[rev16[i]];
  }
}

void FFT_stages(float complex FFT_input[N], float complex FFT_output[N]) {
  float complex temp1[N], temp2[N], temp3[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 < 2; i = i + 1) {
    temp2[i] = temp1[i] + W1[i] * temp1[i + 2];
    temp2[i + 2] = temp1[i] - W1[i] * temp1[i + 2];
  }
  for (int i = 4; i < 6; i = i + 1) {
    temp2[i] = temp1[i] + W1[i] * temp1[i + 2];
    temp2[i + 2] = temp1[i] - W1[i] * temp1[i + 2];
  }
  for (int i = 8; i < 10; i = i + 1) {
    temp2[i] = temp1[i] + W1[i] * temp1[i + 2];
    temp2[i + 2] = temp1[i] - W1[i] * temp1[i + 2];
  }
  for (int i = 12; i < 14; i = i + 1) {
    temp2[i] = temp1[i] + W1[i] * temp1[i + 2];
    temp2[i + 2] = temp1[i] - W1[i] * temp1[i + 2];
  }
stage3:
  for (int i = 0; i < N / 4; i = i + 1) {
```

Writable | Smart Insert | 65 : 3

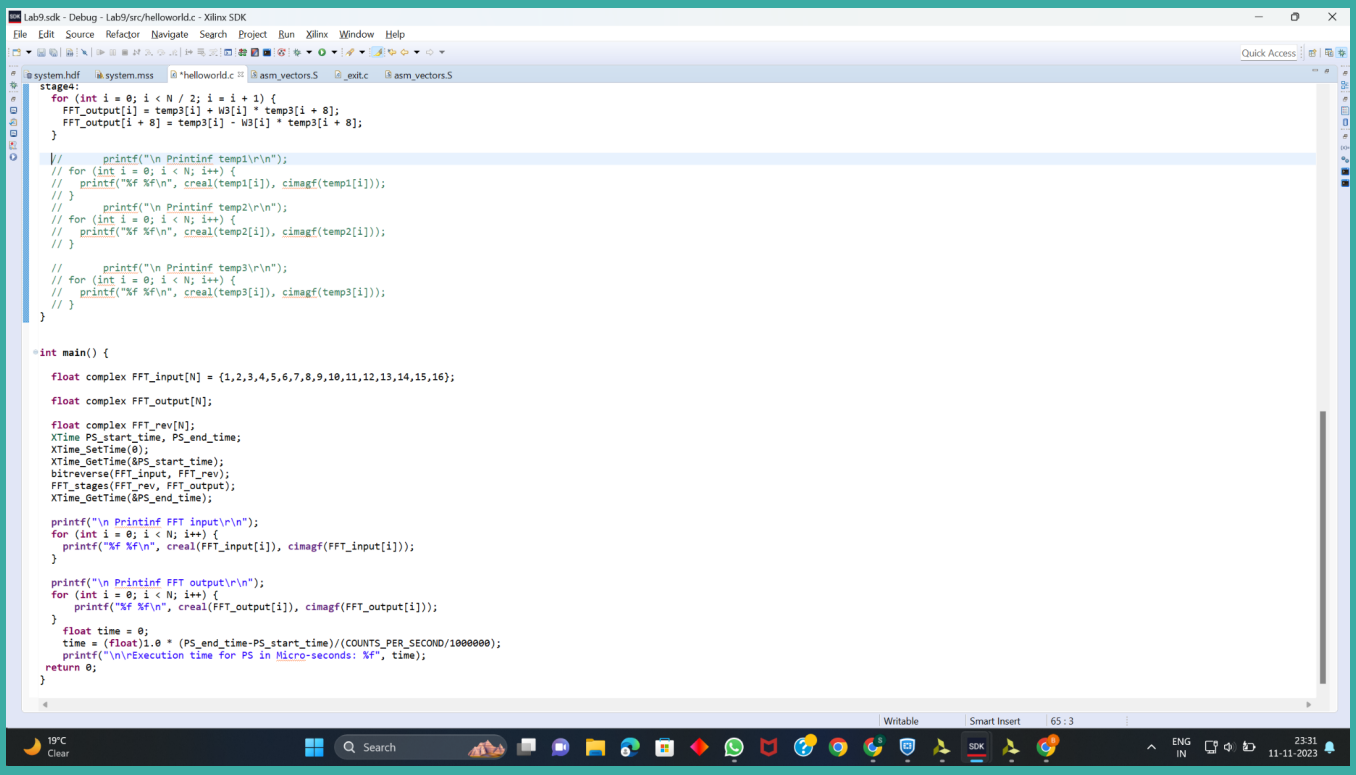
Lab9.sdk - Debug - Lab9/src/helloworld.c - Xilinx SDK

File Edit Source Refactor Navigate Search Project Run Xilinx Window Help

system.hdf | system.mss | *helloworld.c | asm_vectors.S | _exit.c | asm_vectors.S

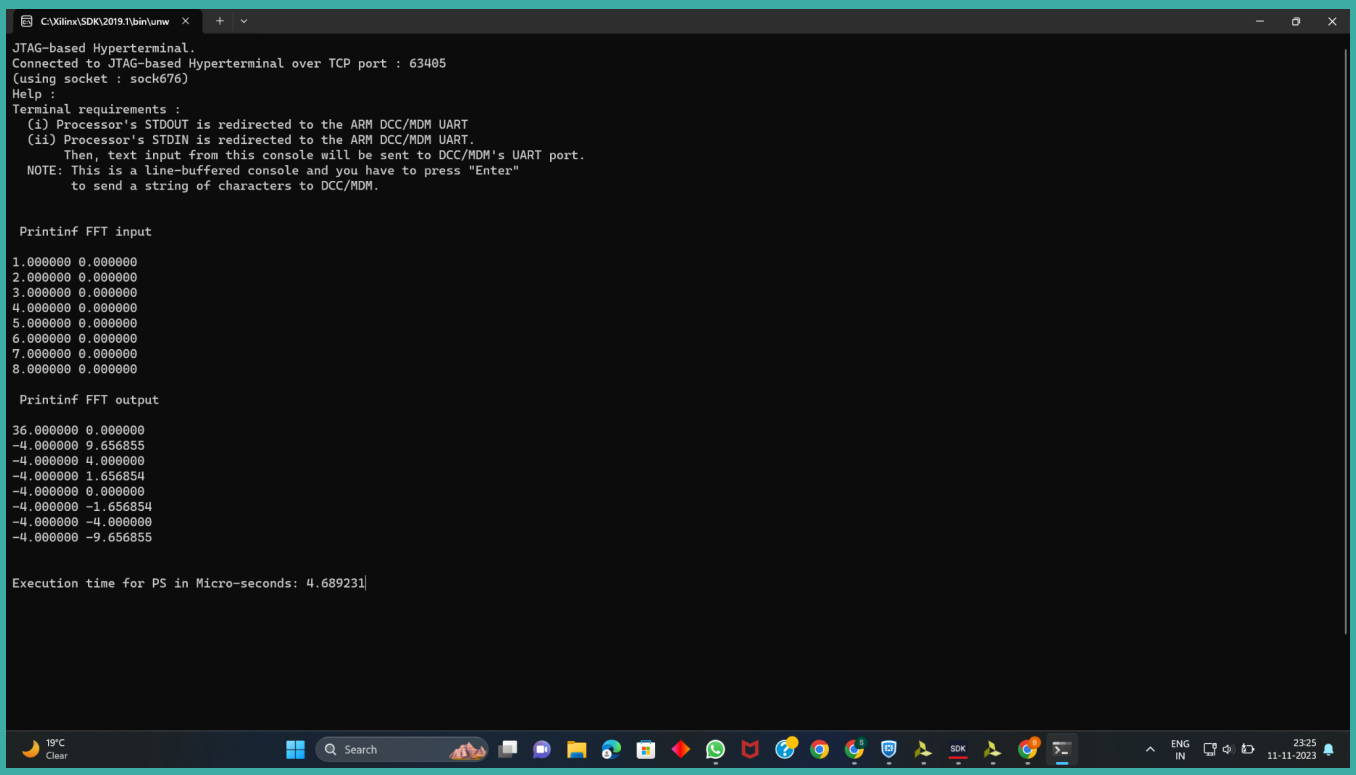
```c
stage3:
  for (int i = 0; i < N / 4; i = i + 1) {
    temp3[i] = temp2[i] + W2[i] * temp2[i + 4];
    temp3[i + 4] = temp2[i] - W2[i] * temp2[i + 4];
  }
  for (int i = 8; i < 12; i = i + 1) {
    temp3[i] = temp2[i] + W2[i] * temp2[i + 4];
    temp3[i + 4] = temp2[i] - W2[i] * temp2[i + 4];
  }
stage4:
  for (int i = 0; i < N / 2; i = i + 1) {
    FFT_output[i] = temp3[i] + W3[i] * temp3[i + 8];
    FFT_output[i + 8] = temp3[i] - W3[i] * temp3[i + 8];
  }

  //       printf("\n Printinf temp1\r\n");
  // for (int i = 0; i < N; i++) {
  //   printf("%f %f\n", creal(temp1[i]), cimagf(temp1[i]));
  // }
  //       printf("\n Printinf temp2\r\n");
  // for (int i = 0; i < N; i++) {
  //   printf("%f %f\n", creal(temp2[i]), cimagf(temp2[i]));
  // }

  //       printf("\n Printinf temp3\r\n");
  // for (int i = 0; i < N; i++) {
  //   printf("%f %f\n", creal(temp3[i]), cimagf(temp3[i]));
  // }
}


int main() {

  float complex FFT_input[N] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};

  float complex FFT_output[N];

  float complex FFT_rev[N];
  XTime PS_start_time, PS_end_time;
  XTime_SetTime(0);
  XTime_GetTime(&PS_start_time);
  bitreverse(FFT_input, FFT_rev);
  FFT_stages(FFT_rev, FFT_output);
  XTime_GetTime(&PS_end_time);

  printf("\n Printinf FFT input\r\n");
  for (int i = 0; i < N; i++) {
    printf("%f %f\n", creal(FFT_input[i]), cimagf(FFT_input[i]));
  }

  printf("\n Printinf FFT output\r\n");
```

Writable | Smart Insert | 65 : 3



jtag terminal output

8 Point FFT:



(ZOOMED OUTPUT)

```
C:\Xilinx\SDK\2019.1\bin\unw

JTAG-based Hyperterminal.
Connected to JTAG-based Hyperterminal over TCP port : 63405
(using socket : sock676)
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.


 Printinf FFT input

1.000000 0.000000
2.000000 0.000000
3.000000 0.000000
4.000000 0.000000
5.000000 0.000000
6.000000 0.000000
7.000000 0.000000
8.000000 0.000000

 Printinf FFT output

36.000000 0.000000
-4.000000 9.656855
-4.000000 4.000000
-4.000000 1.656854
-4.000000 0.000000
-4.000000 -1.656854
-4.000000 -4.000000
-4.000000 -9.656855


Execution time for PS in Micro-seconds: 4.689231
```

Expected output from Matlab

```
>> x = [1,2,3,4,5,6,7,8];
>> fft(x)

ans =

  36.0000 + 0.0000i  -4.0000 + 9.6569i  -4.0000 + 4.0000i  -4.0000 + 1.6569i  -4.0000 + 0.0000i  -4.0000 - 1.6569i  -4.0000 - 4.0000i  -4.0000 - 9.6569i
```

jtag terminal output

16 Point FFT:

```
C:\Xilinx\SDK\2019.1\bin\unw
JTAG-based Hyperterminal.
Connected to JTAG-based Hyperterminal over TCP port : 63367
(using socket : sock684)
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.


 Printinf FFT input

1.000000 0.000000
2.000000 0.000000
3.000000 0.000000
4.000000 0.000000
5.000000 0.000000
6.000000 0.000000
7.000000 0.000000
8.000000 0.000000
9.000000 0.000000
10.000000 0.000000
11.000000 0.000000
12.000000 0.000000
13.000000 0.000000
14.000000 0.000000
15.000000 0.000000
16.000000 0.000000

 Printinf FFT output

136.000000 0.000000
-8.000000 40.218719
-8.000000 19.313709
-7.999997 11.972851
-8.000000 8.000000
-8.000000 5.345429
-8.000000 3.313708
-8.000002 1.591299
-8.000000 0.000000
-8.000000 -1.591299
-8.000000 -3.313708
```

```
C:\Xilinx\SDK\2019.1\bin\unw

 Printinf FFT input

1.000000 0.000000
2.000000 0.000000
3.000000 0.000000
4.000000 0.000000
5.000000 0.000000
6.000000 0.000000
7.000000 0.000000
8.000000 0.000000
9.000000 0.000000
10.000000 0.000000
11.000000 0.000000
12.000000 0.000000
13.000000 0.000000
14.000000 0.000000
15.000000 0.000000
16.000000 0.000000

 Printinf FFT output

136.000000 0.000000
-8.000000 40.218719
-8.000000 19.313709
-7.999997 11.972851
-8.000000 8.000000
-8.000000 5.345429
-8.000000 3.313708
-8.000002 1.591299
-8.000000 0.000000
-8.000000 -1.591299
-8.000000 -3.313708
-8.000004 -5.345427
-8.000000 -8.000000
-8.000000 -11.972846
-8.000000 -19.313709
-7.999998 -40.218727


Execution time for PS in Micro-seconds: 11.618462
```

(ZOOMED OUTPUT)

```
C:\Xilinx\SDK\2019.1\bin\unw

        to send a string of characters to DCC/MDM.


 Printinf FFT input

1.000000 0.000000
2.000000 0.000000
3.000000 0.000000
4.000000 0.000000
5.000000 0.000000
6.000000 0.000000
7.000000 0.000000
8.000000 0.000000
9.000000 0.000000
10.000000 0.000000
11.000000 0.000000
12.000000 0.000000
13.000000 0.000000
14.000000 0.000000
15.000000 0.000000
16.000000 0.000000

 Printinf FFT output

136.000000 0.000000
-8.000000 40.218719
-8.000000 19.313709
-7.999997 11.972851
-8.000000 8.000000
-8.000000 5.345429
-8.000000 3.313708
-8.000002 1.591299
-8.000000 0.000000
-8.000000 -1.591299
-8.000000 -3.313708
-8.000004 -5.345427
-8.000000 -8.000000
-8.000000 -11.972846
-8.000000 -19.313709
-7.999998 -40.218727


Execution time for PS in Micro-seconds: 11.618462
```

19°C
Clear

Search

Expected output from Matlab

```
>> x = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16];
>> fft(x)

ans =

   1.0e+02 *

  Columns 1 through 9

   1.3600 + 0.0000i  -0.0800 + 0.4022i  -0.0800 + 0.1931i  -0.0800 + 0.1197i  -0.0800 + 0.0800i  -0.0800 + 0.0535i  -0.0800 + 0.0331i  -0.0800 + 0.0159i  -0.0800 + 0.0000i

  Columns 10 through 16

  -0.0800 - 0.0159i  -0.0800 - 0.0331i  -0.0800 - 0.0535i  -0.0800 - 0.0800i  -0.0800 - 0.1197i  -0.0800 - 0.1931i  -0.0800 - 0.4022i
```

Thank You