
[SoC Design] Lab 1: SW Design

Chester Sungchung Park (박성정)
SoC Design Lab, Konkuk University
Webpage: <http://soclub.konkuk.ac.kr>

Teaching Assistants

- ❑ Youngho Seo (younghoseo@konkuk.ac.kr), M.S. candidate
- ❑ Sanghun Lee (sanghunlee@konkuk.ac.kr), M.S. candidate

Outline

❑ Objectives

❑ SW design (DFT)

- Creating projects
- Adding math library
- Running C applications
- Debugging C applications

❑ SW optimization (FFT)

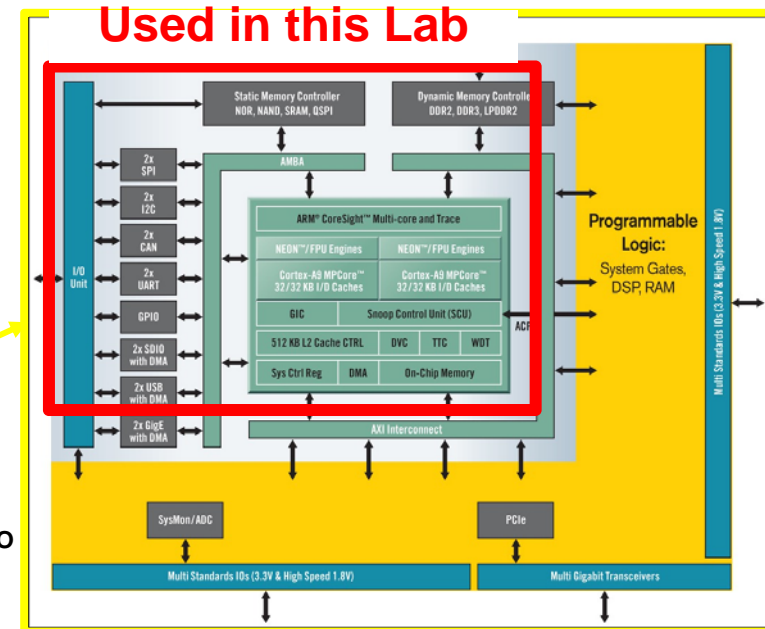
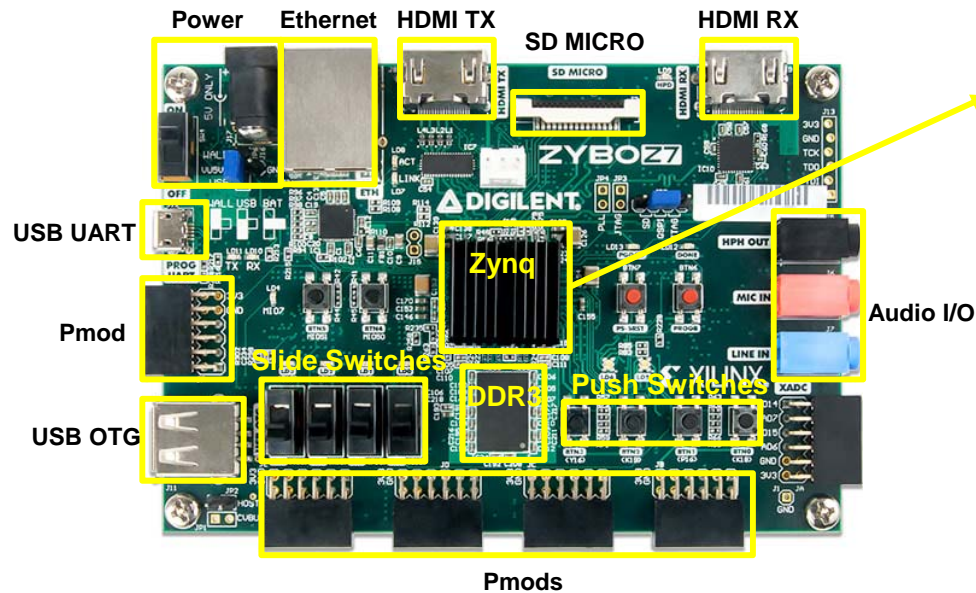
- Programming in C
- Programming in assembly
- Setting optimization level

Objectives

- ❑ After completing this lab, you will be able to :
 - Program an application in either C or assembly
 - Run an application
 - Debug an application
 - Measure the execution time of an application
 - Optimize the performance of an application in either C or assembly

Introduction

□ ZYNQ



Introduction

□ Design flow

Vivado

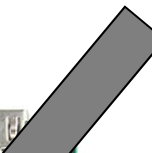
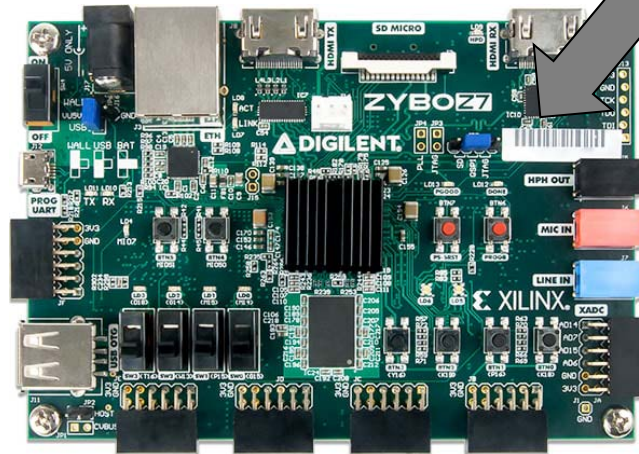


Used in this Lab

SDK



ZYNQ/Zybo



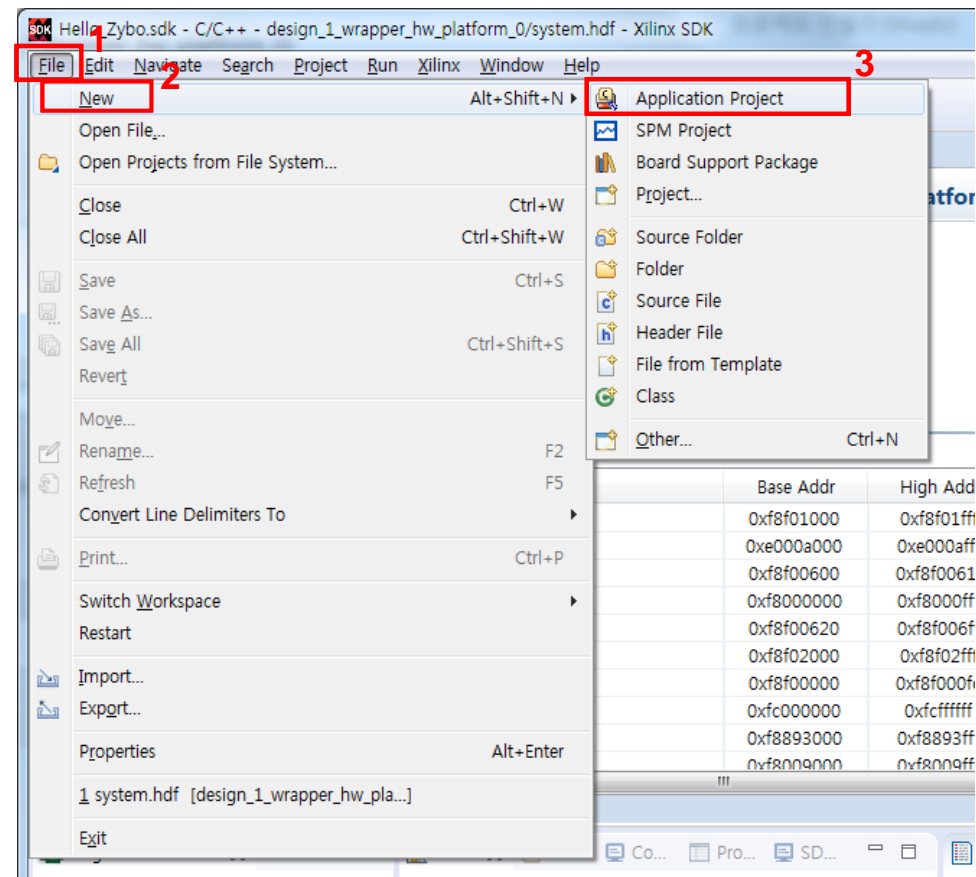
S/W Design (DFT)

Creating Projects

- ❑ Repeat the previous steps
 - Follow pp. 6~26 of the following lab workbook:
[Lab_SD2019_0w.pdf](#)

Running C Applications

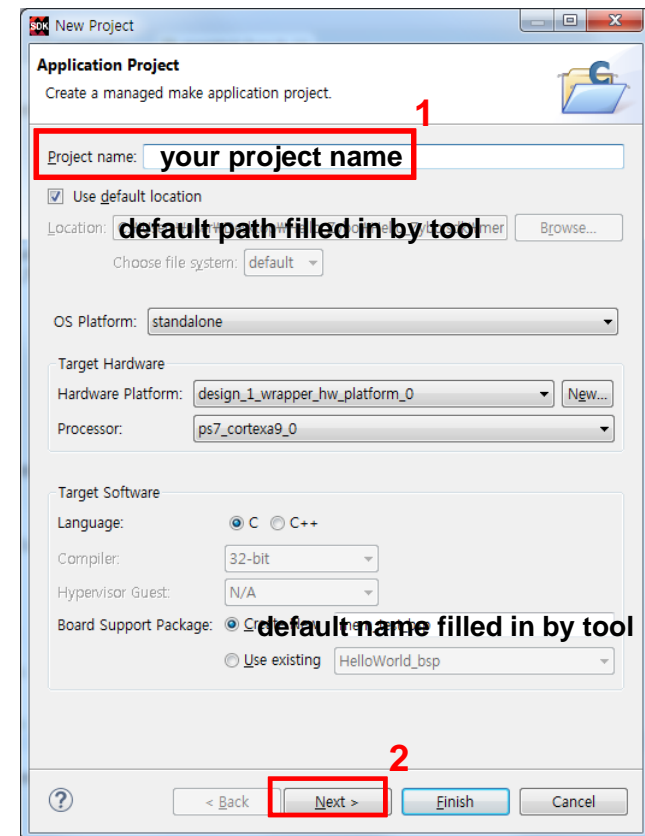
- ❑ Create a C application project
 - Click '**File**' > '**New**' > '**Application Project**'



Running C Applications

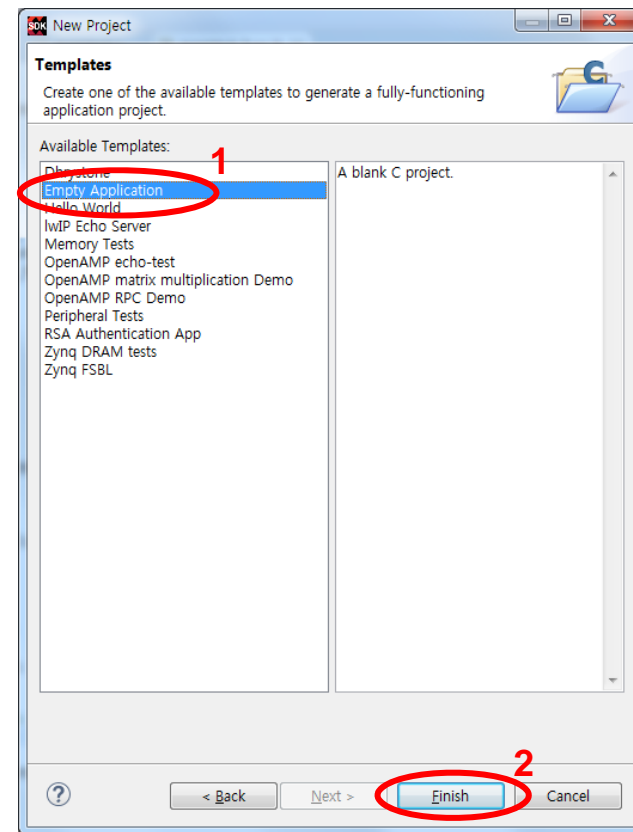
❑ Create a C application project (cont'd)

- Type **<your project name>** in the Project name field
- The **'Board Support Package'** field can be set up to use an existing BSP or a new BSP can be created based on the project name. (Do not modify)



Running C Applications

- ❑ Create a C application project (cont'd)
 - Select '*Empty Application*' from the Template list
 - Click '*Finish*'



Creating Projects

❑ Add source files

- Copy '**main.c.1**' and paste into the '**src**' folder.

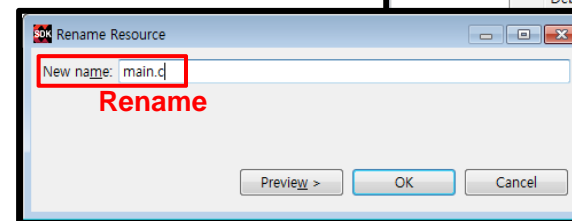
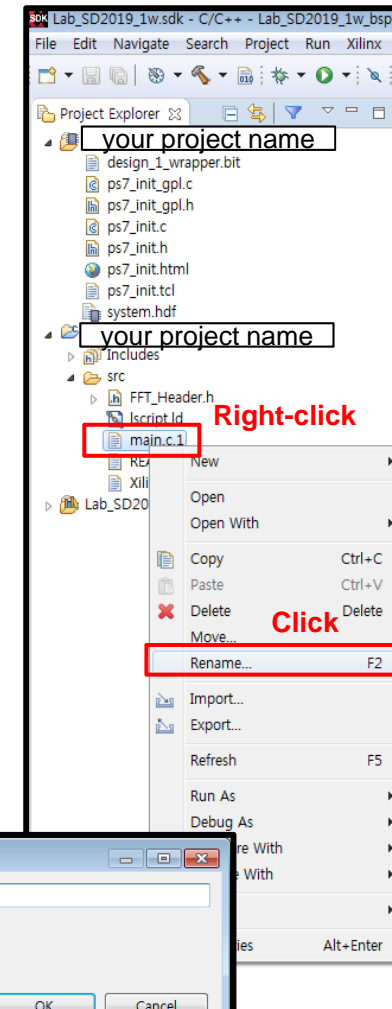


main.c.1

- Change the name from '**main.c.1**' to '**main.c**' after clicking '**Rename**'.
- Copy '**FFT_Header.h**' and paste into the '**src**' folder



FFT_Header.h



Creating Projects

❑ Check the C application program

- Expand '**your project name** > **src**' to see all of the source files included in the project by clicking the arrow next to '**src**'.
- Double-click the '**main.c**' file to open it

Function

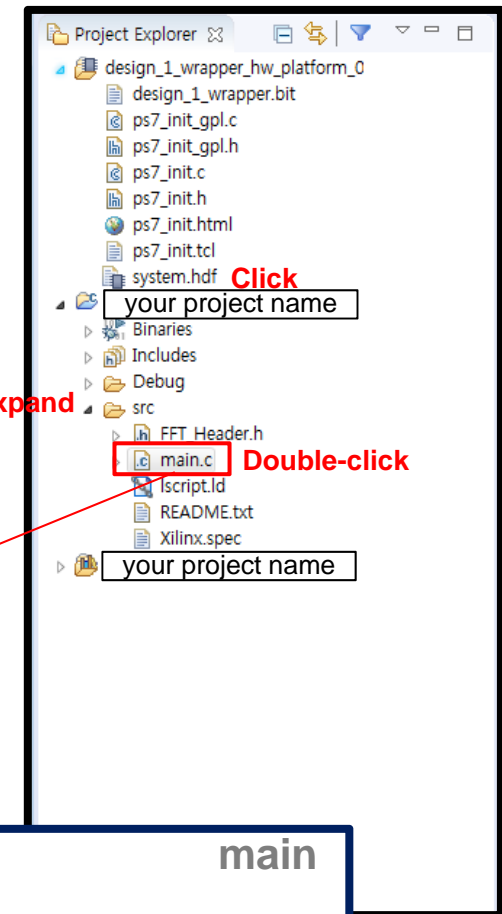
```
int Re_ordering(int x) {  
    return 32 * (x % 2) + 16 * ((x % 4) / 2) + 8 * ((x % 8) / 4)  
        + 4 * ((x % 16) / 8) + 2 * ((x % 32) / 16) + x / 32;  
}  
  
void DFT()  
{  
    int n = 0, i = 0, k = 0;  
  
    complex input[N];  
    complex temp_mult[N];  
  
    int out_re[N] = {0};  
    int out_im[N] = {0};  
  
    for (n = 0; n < N; n++)  
    {  
        input[n].re = in_real[n];  
        input[n].im = in_imag[n];  
    }  
  
    for (i = 0; i < N; i++)  
    {  
        X_DFT[i] = add_cal(init1_int, init2_int);  
        for (k = 0; k < N; k++)  
        {  
            temp_mult[k] = multiple(input[k], W[(k*i)%64]);  
            X_DFT[i] = add_cal(temp_mult[k], X_DFT[i]);  
        }  
    }  
  
    for (n = 0; n < N; n++)  
    {  
        out_re[n] = X_DFT[n].re;  
        out_im[n] = X_DFT[n].im;  
    }  
}
```

Header files & global variables

```
#include <stdio.h>  
#include <xtime.h>  
#include <xil_cache.h>  
#include <math.h>  
#include "FFT_Header.h"  
  
#define N 64  
  
complex X_DFT[N];  
  
int time;
```

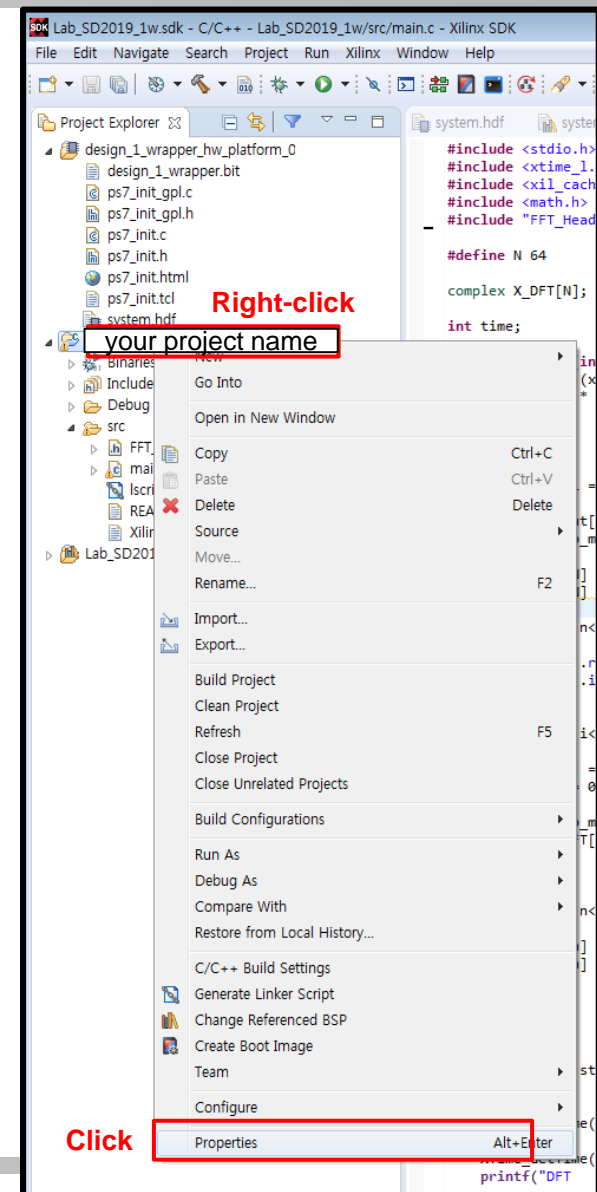
main

```
int main() {  
    XTime start, stop;  
    int i = 0;  
  
    XTime_GetTime((XTime*)&start);  
    DFT();  
    XTime_GetTime((XTime*)&stop);  
    printf("DFT %8.3f us\n", ((float)stop - (float)start) / COUNTS_PER_SECOND * 1000000);  
  
    return 0;  
}
```



Adding Math Library

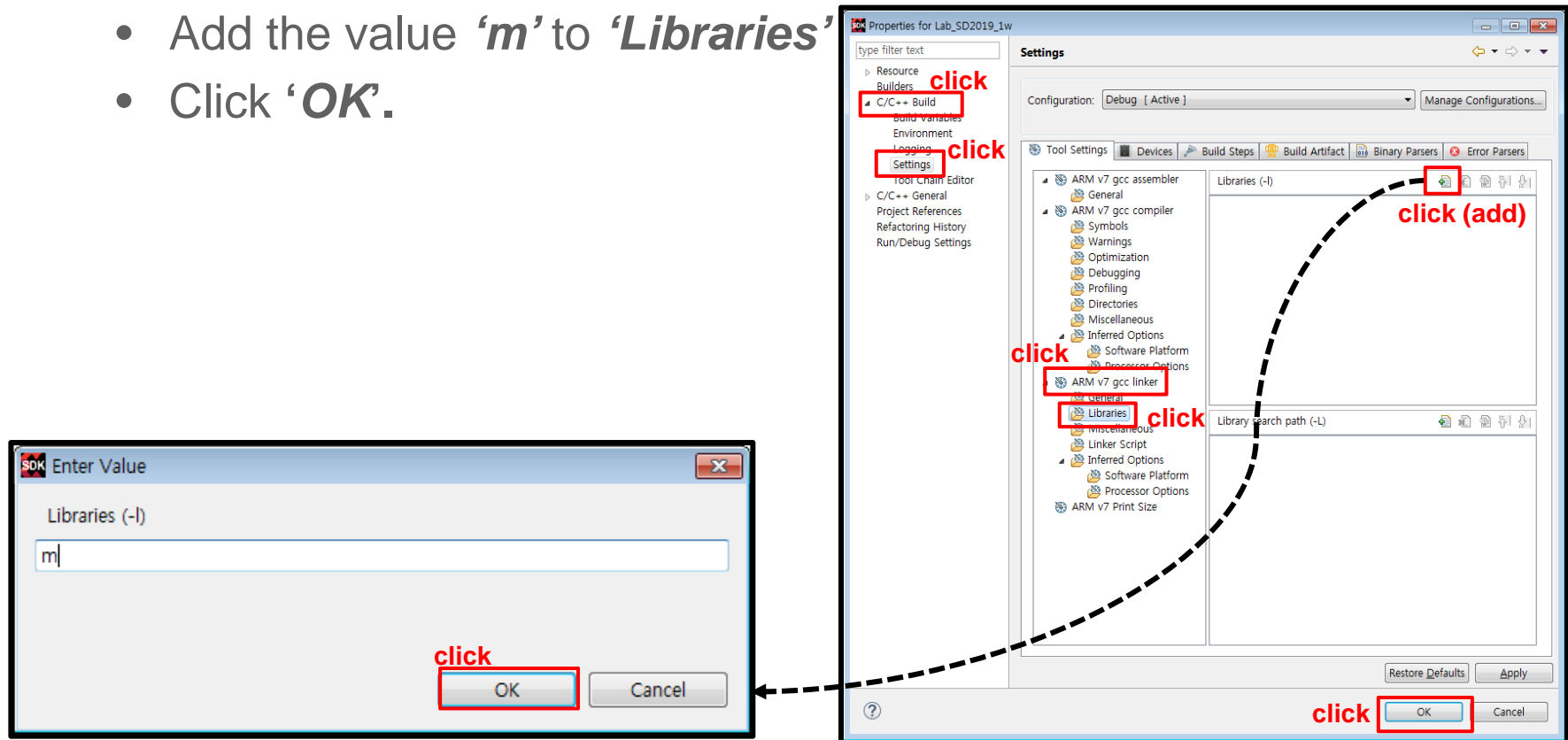
- ❑ Add math library to linker
 - Choose '*your project name*' > **Properties**



Adding Math Library

❑ Add math library to linker (cont'd)

- Click '**C/C++ Build > Settings > ARM v7 gcc linker > Libraries > add**'
- Add the value '**m**' to '**Libraries**'
- Click '**OK**'.



Running C Applications

❑ Review the function: *'main()'*

- ① Measures the start time (*'start'*)
- ② Calls *'DFT()'*
- ③ Measures the stop time (*'stop'*)
- ④ Prints the execution time

```
int main() {  
    XTime start, stop;  
    int i = 0;  
  
    ① XTime_GetTime((XTime*)&start);  
    ② DFT();  
    ③ XTime_GetTime((XTime*)&stop);  
    ④ printf("DFT          %8.3f us\n", ((float)stop - (float)start)/COUNTS_PER_SECOND*1000000);  
  
    return 0;  
}
```


Running C Applications

□ Review the function '*DFT()*'

- ① Takes the input (from '*header*')
- ② Performs DFT
- ③ Generates the output

```
void DFT()
{
    int n = 0, i = 0, k = 0;

    complex input[N];
    complex temp_mult[N];

    int out_re[N] = {0};
    int out_im[N] = {0};

    for (n=0; n<N; n++) ①
    {
        input[n].re=in_real[n];
        input[n].im=in_imag[n];
    }

    for (i=0; i<N; i++) ②
    {
        X_DFT[i] = add_cal(init1_int,init2_int);
        for (k=0; k<N; k++)
        {
            temp_mult[k] = multiple(input[k],W[(k*i)%64]);
            X_DFT[i] = add_cal(temp_mult[k],X_DFT[i]);
        }
    }

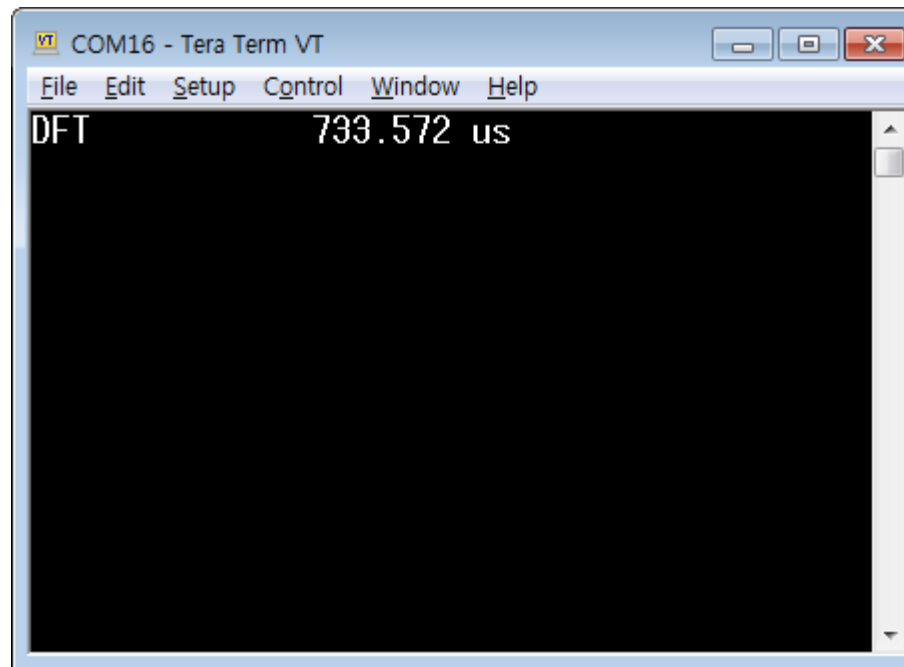
    for (n=0; n<N; n++) ③
    {
        out_re[n] = X_DFT[n].re;
        out_im[n] = X_DFT[n].im;
    }
}
```

Running C Applications

- ❑ Repeat the previous steps
 - Follow pp. 31~34 of the following lab workbook:
[Lab_SD2019_0w.pdf](#)

Running C Applications

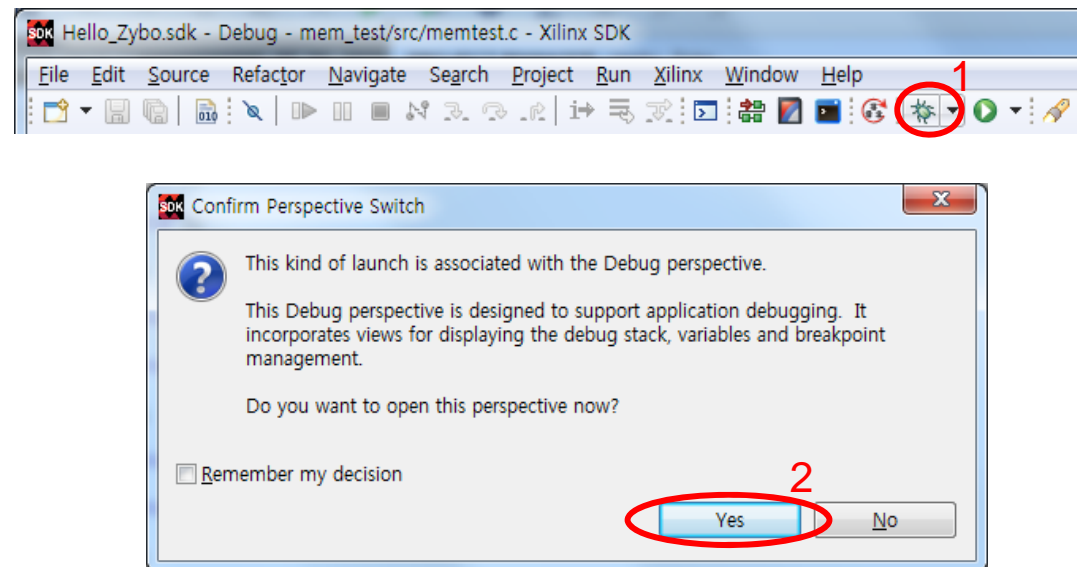
- ❑ Run the application Check the output of the application on '*Tera Term*'
 - ✓ You should see the execution time as shown below.



Debugging C Applications

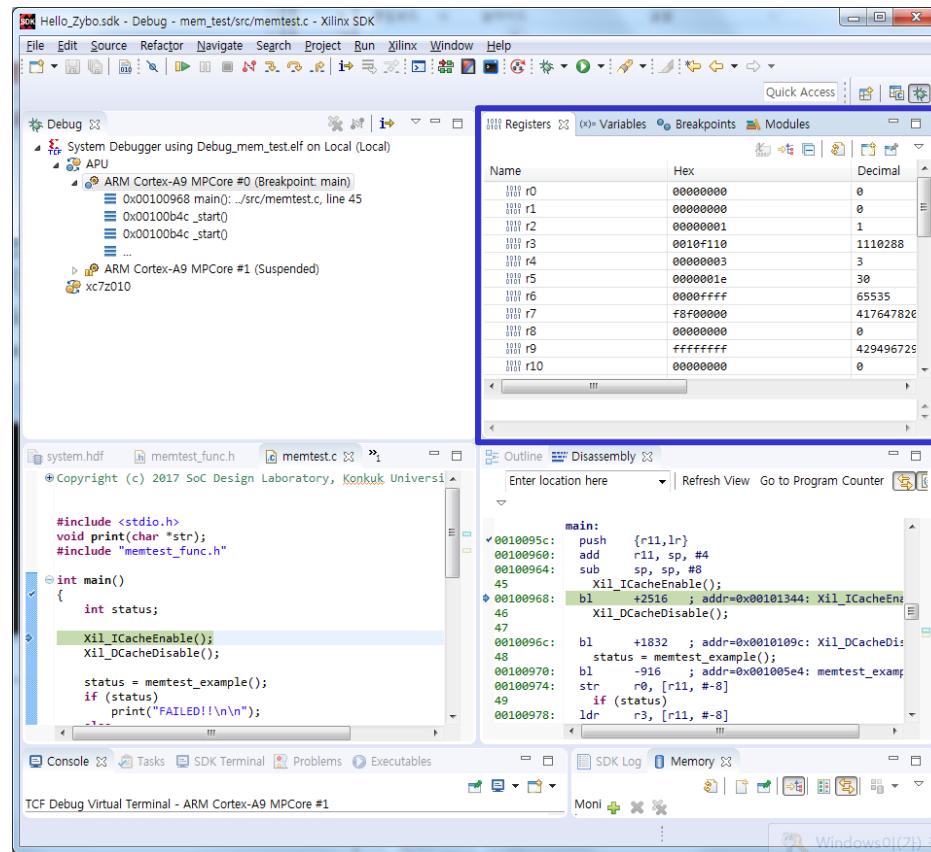
❑ Debug the application

- Click the '**Debug System Debugger**' icon and then click '**Yes**'



Debugging C Applications

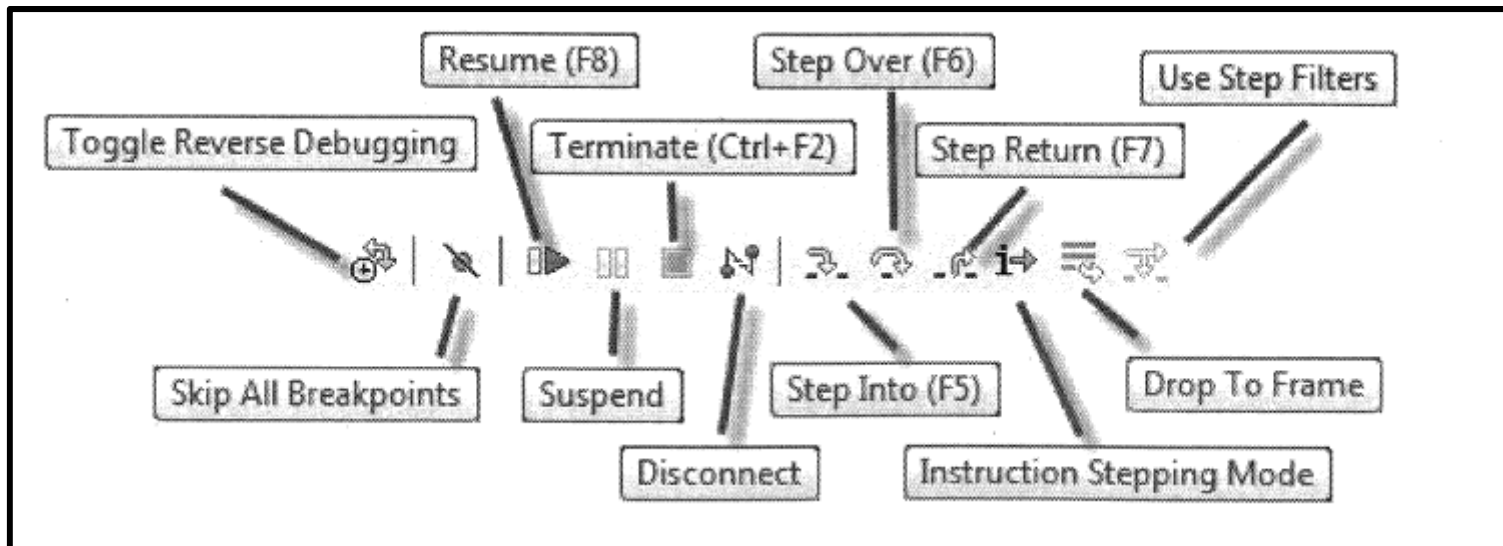
- ❑ Debug the application (cont'd)
 - Choose the **'Registers'** or **'Variable'** tab



Debugging C Applications

- ❑ Debug the application (cont'd)

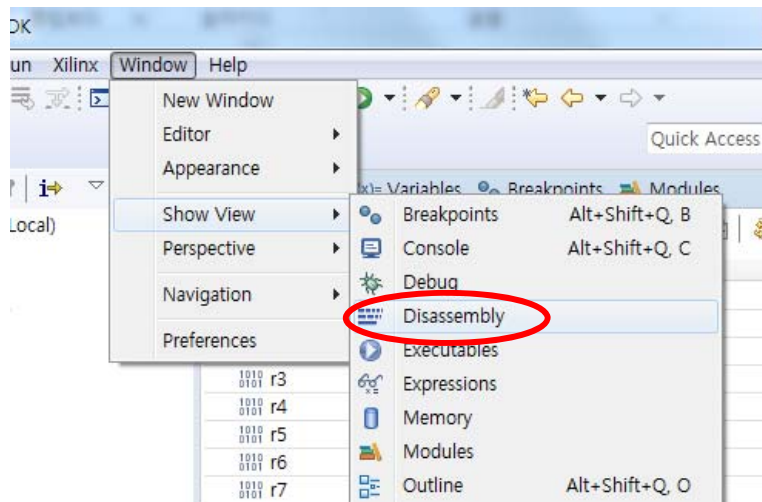
Debug Tool Bar



Debugging C Applications

❑ Review the disassembly

- Open the '**Window**' > '**Show View**' menu and then click '**Disassembly**'

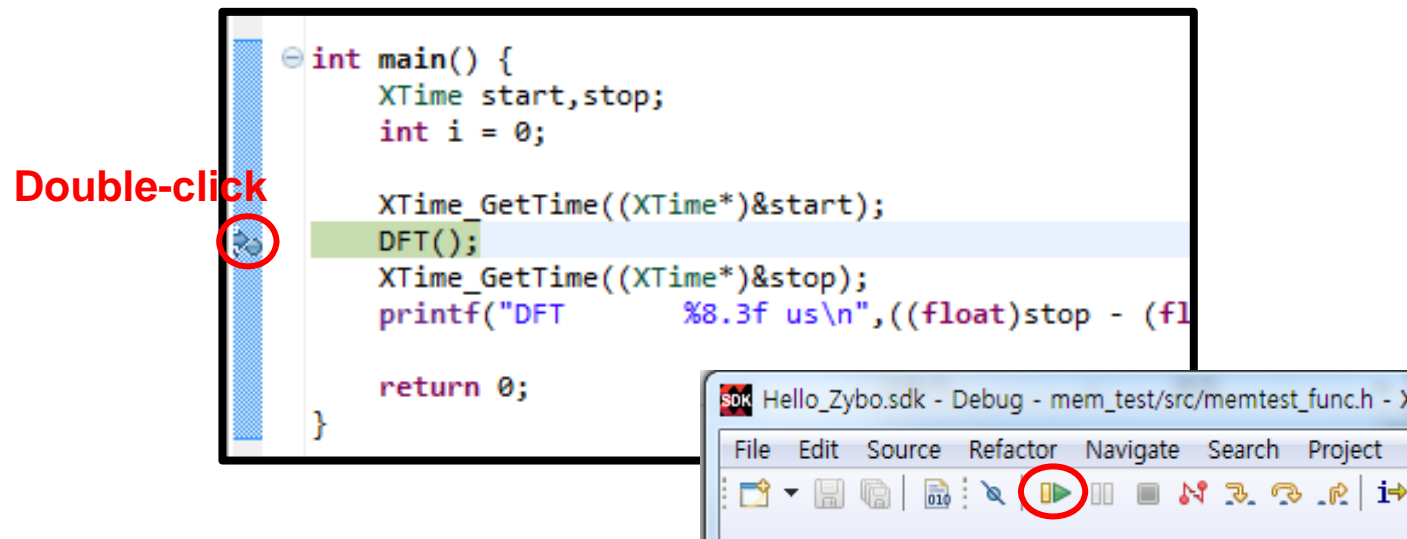
A screenshot of the Xilinx IDE's 'Disassembly' window. The window shows assembly code for a program. The code is as follows:

```
main:
0010095c: push    {r11,lr}
00100960: add     r11, sp, #4
00100964: sub     sp, sp, #8
45      Xil_ICacheEnable();
00100968: bl      +2516 ; addr=0x00101344: Xil_ICacheEna
46      Xil_DCacheDisable();
47
0010096c: bl      +1832 ; addr=0x0010109c: Xil_DCacheDis
48      status = memtest_example();
00100970: bl      -916 ; addr=0x001005e4: memtest_examp
00100974: str     r0, [r11, #-8]
49      if (status)
00100978: ldr     r3, [r11, #-8]
0010097c: cmp     r3, #0
00100980: beq     +12 ; addr=0x00100994: main + 0x0000
50      print("FAILED!!\n\n");
51      else
00100984: movw    r0, #57924
00100988: movt    r0, #16
0010098c: bl      +172 ; addr=0x00100a40: print
00100990: b       +8 ; addr=0x001009a0: main + 0x0000
52      print("PASSED!!\n\n");
53
00100994: movw    r0, #57936
00100998: movt    r0, #16
0010099c: bl      +156 ; addr=0x00100a40: print
54      status = memtest_0();
001009a0: bl      -616 ; addr=0x00100740: memtest_0
```

Debugging C Applications

❑ Review the disassembly (cont'd)

- Double-click on the left side of a code line to add a breakpoint
- Click the '**Resume**' icon to continue debugging



Debugging C Applications

□ Review the disassembly (cont'd)

The image shows a debugger window with two panes. The left pane displays the C source code for 'main.c', and the right pane displays the corresponding disassembly. A blue arrow points from the 'DFT();' line in the C code to the disassembly instruction 'b1 -784 ; addr=0x00100960: DFT'.

C Source Code (main.c):

```
main.c Stage6_Assem... Disassembly »3
{
    input[n].re=in_real[n];
    input[n].im=in_imag[n];
}
for (i = 0; i<N; i++)
{
    X_DFT[i] = add_cal(init1_int,init2_int);
    for (k = 0; k<N; k++)
    {
        temp_mult[k] = multiple(input[k],W[(k*i)%64]);
        X_DFT[i] = add_cal(temp_mult[k],X_DFT[i]);
    }
}
for (n = 0; n<N; n++)
{
    out_re[n] = X_DFT[n].re;
    out_im[n] = X_DFT[n].im;
}
}

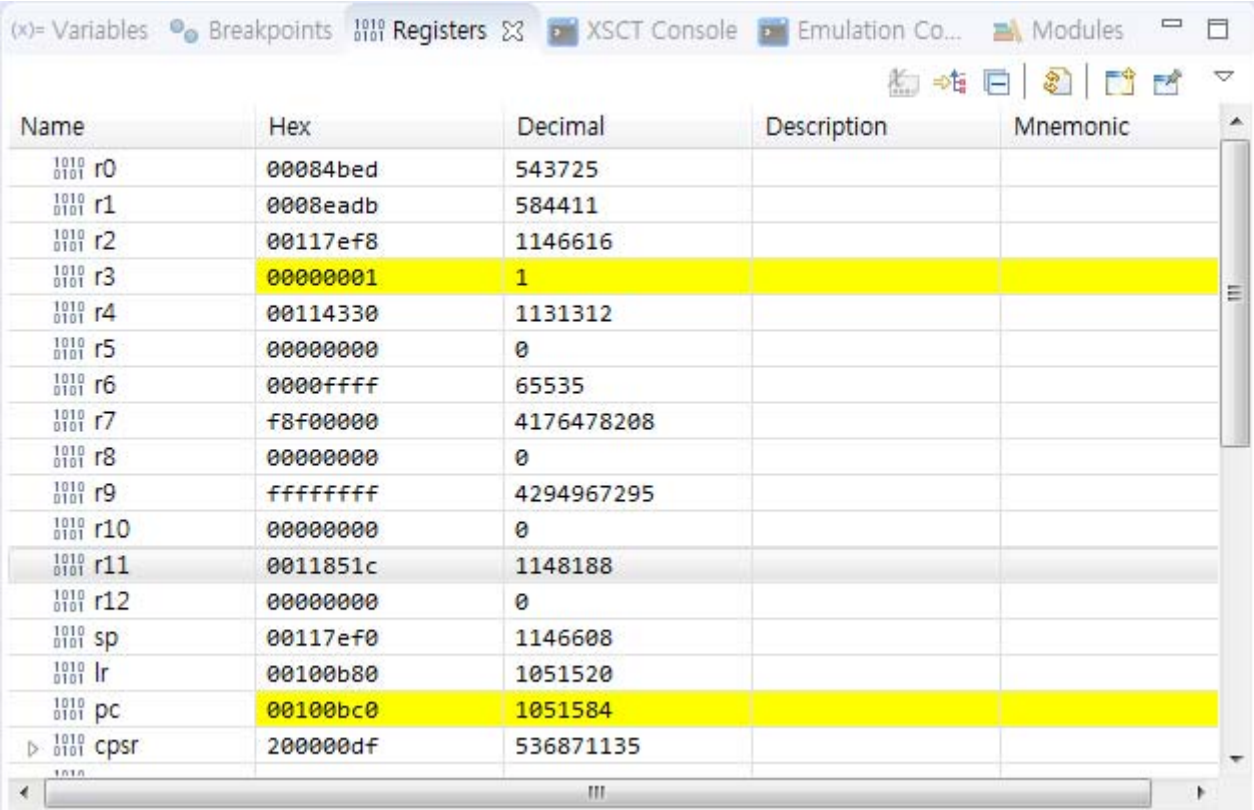
int main() {
    XTime start,stop;
    XTime_GetTime((XTime*)&start);
    DFT();
    XTime_GetTime((XTime*)&stop);
    printf("DFT      %8.3f us\r\n",((float)stop - (float)sta
    return 0;
}
```

Disassembly:

```
Outline Disassembly
Enter location here Refresh View Go to Program Counter
00100c5c: sub    r3, r11, #20
00100c60: mov    r0, r3
00100c64: bl     +476      ; addr=0x00100e48: XTime_GetTime
86      DFT();
00100c68: bl     -784      ; addr=0x00100960: DFT
87      XTime_GetTime((XTime*)&stop);
00100c6c: sub    r3, r11, #28
00100c70: mov    r0, r3
00100c74: bl     +460      ; addr=0x00100e48: XTime_GetTime
88      printf("DFT      %8.3f us\r\n",((float)stop - (float)start);
89
00100c78: ldrd   r2, r3, [r11, #-28]
00100c7c: mov    r0, r2
00100c80: mov    r1, r3
00100c84: bl     +1012     ; addr=0x00101080: __aeabi_ul2f
00100c88: vmov   s16, r0
00100c8c: ldrd   r2, r3, [r11, #-20]
00100c90: mov    r0, r2
00100c94: mov    r1, r3
00100c98: bl     +992      ; addr=0x00101080: __aeabi_ul2f
00100c9c: vmov   s15, r0
00100ca0: vsub.f32 s14, s16, s15
00100ca4: vldr   s13, [pc, #+48]
00100ca8: vdiv.f32 s15, s14, s13
00100cac: vldr   s14, [pc, #+44]
00100cb0: vmul.f32 s15, s15, s14
00100cb4: vcvt.f64.f32 d16, s15
00100cb8: vmov   r2, r3, d16
00100cbc: movw   r0, #56112
00100cc0: movt   r0, #16
00100cc4: bl     +1656     ; addr=0x00101344: printf
90      return 0;
```

Debugging C Applications

- ❑ Check the content of a register



Name	Hex	Decimal	Description	Mnemonic
r0	00084bed	543725		
r1	0008eadb	584411		
r2	00117ef8	1146616		
r3	00000001	1		
r4	00114330	1131312		
r5	00000000	0		
r6	0000ffff	65535		
r7	f8f00000	4176478208		
r8	00000000	0		
r9	ffffffff	4294967295		
r10	00000000	0		
r11	0011851c	1148188		
r12	00000000	0		
sp	00117ef0	1146608		
lr	00100b80	1051520		
pc	00100bc0	1051584		
cpsr	200000df	536871135		

Debugging C Applications

- ❑ Check the content of a memory location
 - Location for the loop variable (i)

SDK Log Memory

Monitors

0x00118508

0x00118508 <Hex Integ 0x00118508 : 0x118508 New Renderings...

Address	0 - 3	4 - 7	8 - B	C - F
00118500	9682	64	1	64
00118510	3	30	1148220	1051756
00118520	0	1148208	1173952210	0
00118530	-1648853999	541615911	0	1052104
00118540	0	0	0	0
00118550	0	0	0	0
00118560	0	0	0	0
00118570	0	0	0	0
00118580	0	0	0	0
00118590	0	0	0	0
001185A0	0	0	0	0
001185B0	0	0	0	0
001185C0	0	0	0	0
001185D0	0	0	0	0
001185E0	0	0	0	0
001185F0	0	0	0	0

Writable Smart Insert 64 : 1

S/W Optimization (FFT)

Programming in C

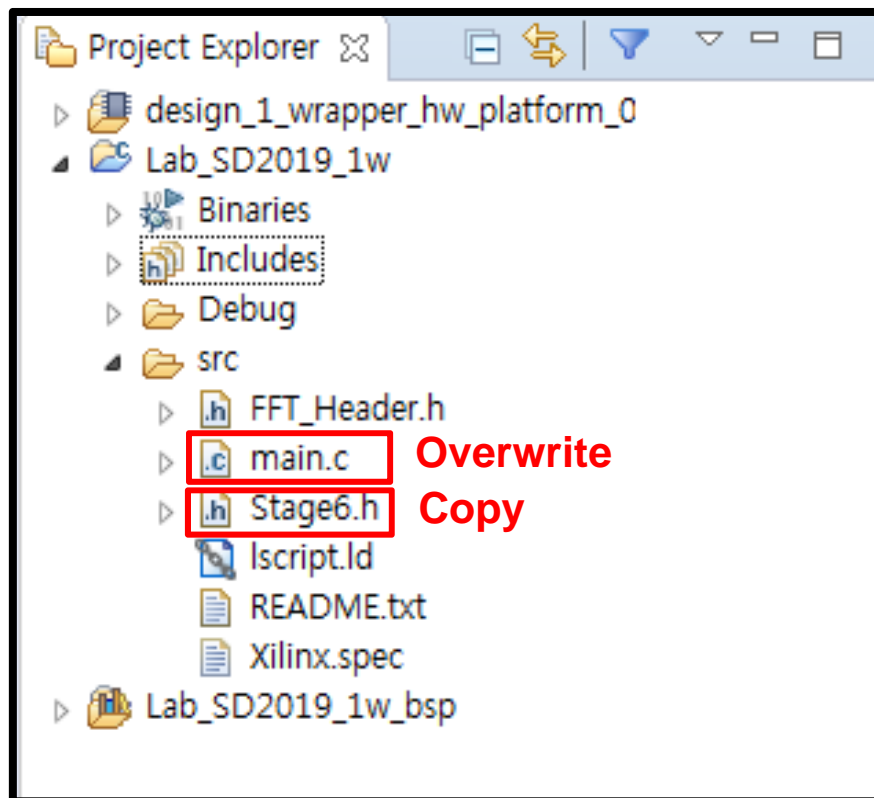
- ❑ Copy the following files and paste them into the '**src**' folder.



main.c.2



Stage6.h



Programming in C

❑ Review the function '*main()*'

- ① Calls '*FFT()*'
- ② Compares the output of '*FFT()*' with that of '*DFT()*'

```
int main() {
    XTime start, stop;
    int i = 0;

    float error_total, error_real, error_imag;
    float sig_total;
    float SNR;

    XTime_GetTime((XTime*)&start);
    DFT();
    XTime_GetTime((XTime*)&stop);
    printf("DFT      %8.3f us\n", ((float)stop - (float)start)/COUNTS_PER_SECOND*1000000);

    XTime_GetTime((XTime*)&start);
    FFT();
    XTime_GetTime((XTime*)&stop);
    printf("FFT      %8.3f us\n", ((float)stop - (float)start)/COUNTS_PER_SECOND*1000000);

    error_total = 0;
    sig_total = 0;
    for(i = 0; i < N; i++){
        error_real = (X_DFT[i].re) - (X_FFT[i].re);
        error_imag = (X_DFT[i].im) - (X_FFT[i].im);

        error_total += error_real*error_real + error_imag*error_imag;

        sig_total += (X_DFT[i].re)*(X_DFT[i].re) + (X_DFT[i].im)*(X_DFT[i].im);
    }
    SNR = 10*log10(sig_total/error_total);
    xil_printf("FFT model SNR : %d dB\n", (int)SNR);

    return 0;
}
```

① {

② {

Programming in C

□ Review the function 'FFT()'

- ① Butterfly: Stage 1 ~ Stage 5
- ② Butterfly: Stage 6 (*incomplete*)
- ③ Reordering

```
#ifndef STAGE6_H_
#define STAGE6_H_

void Stage6(complex *output, complex *input)
{
    int n;

    //////////////////////////////////////
    //
    //      Fill Your Code Here.
    //
    //////////////////////////////////////

}

#endif /* STAGE6_H_ */
```

```
void FFT()
{
    complex input[N], temp[N];

    int out_re[N];
    int out_im[N];

    int data;
    int n, k;

    for (data=0; data<N; data++)
    {
        input[data].re=in_real[data];
        input[data].im=in_imag[data];
    }

    for (n=0; n<32; n++) //stage1
    {
        temp[n]=add_cal(input[n], input[n+32]);
        temp[n+32]=multiple(sub_cal(input[n], input[n+32]), W[n]);
    }

    for (n=0; n<16; n++) //stage2
    {
        for (k=0; k<2; k++)
        {
            input[n+(32*k)]=add_cal(temp[n+(32*k)], temp[n+((32*k)+16)]);
            input[n+((32*k)+16)]=multiple(sub_cal(temp[n+(32*k)], temp[n+((32*k)+16)]), W[2*n]);
        }
    }

    for (n=0; n<8; n++) //stage-3
    {
        for (k=0; k<4; k++)
        {
            temp[n+(16*k)] = add_cal(input[n+(16*k)], input[n+((16*k)+8)]);
            temp[n+((16*k)+8)] = multiple(sub_cal(input[n+(16*k)], input[n+((16*k)+8)]), W[4*n]);
        }
    }

    for (n=0; n<4; n++) //stage4
    {
        for (k=0; k<8; k++)
        {
            input[n+(8*k)] = add_cal(temp[n+(8*k)], temp[n+((8*k)+4)]);
            input[n+((8*k)+4)] = multiple(sub_cal(temp[n+(8*k)], temp[n+((8*k)+4)]), W[8*n]);
        }
    }

    for (n=0; n<2; n++) //stage5
    {
        for (k=0; k<16; k++)
        {
            temp[n+(4*k)]=add_cal(input[n+(4*k)], input[n+(4*k+2)]);
            temp[n+(4*k+2)]=multiple(sub_cal(input[n+(4*k)], input[n+(4*k+2)]), W[16*n]);
        }
    }

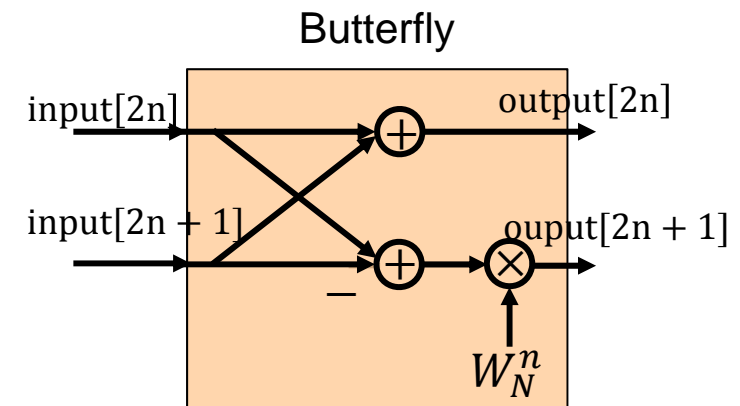
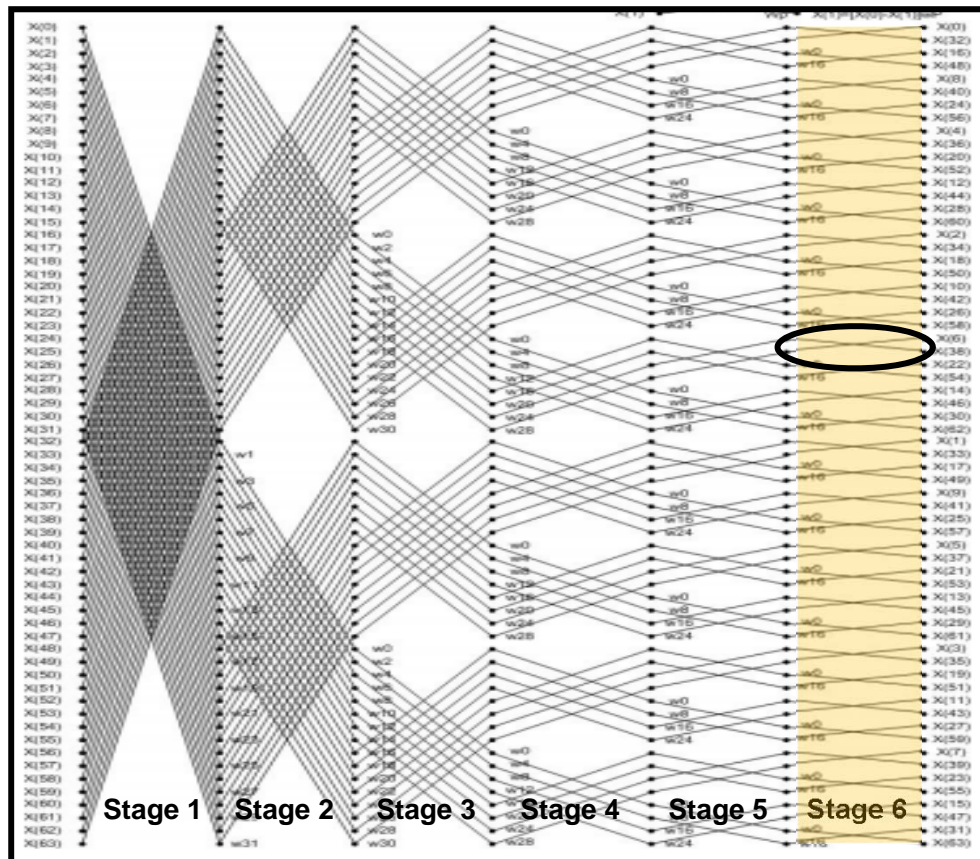
    // Stage 6
    Stage6(input, temp);

    for (n=0; n<N; n++)
    {
        X_FFT[n]=input[Re_ordering(n)];
    }

    for (n=0; n<N; n++)
    {
        out_im[n]=(X_FFT[n].im)>>10;
        out_re[n]=(X_FFT[n].re)>>10;
    }
}
```

Programming in C

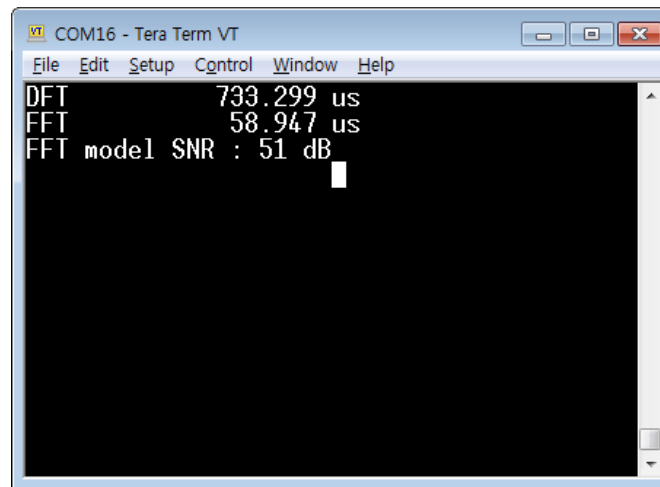
- Add lines to the *'Fill Your Code Here'* section in *'Stage6.h'*



Programming in C

❑ Run the application

- Check the output of the application on '*Tera Term*'
 - ✓ Check the performance gain.
 - ✓ Check the accuracy given by SNR



The screenshot shows a Tera Term window titled 'COM16 - Tera Term VT'. The window has a menu bar with 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. The main display area shows the following text:

```
DFT      733.299 us
FFT      58.947 us
FFT model SNR : 51 dB
```

A cursor is visible on the line 'FFT model SNR : 51 dB'.

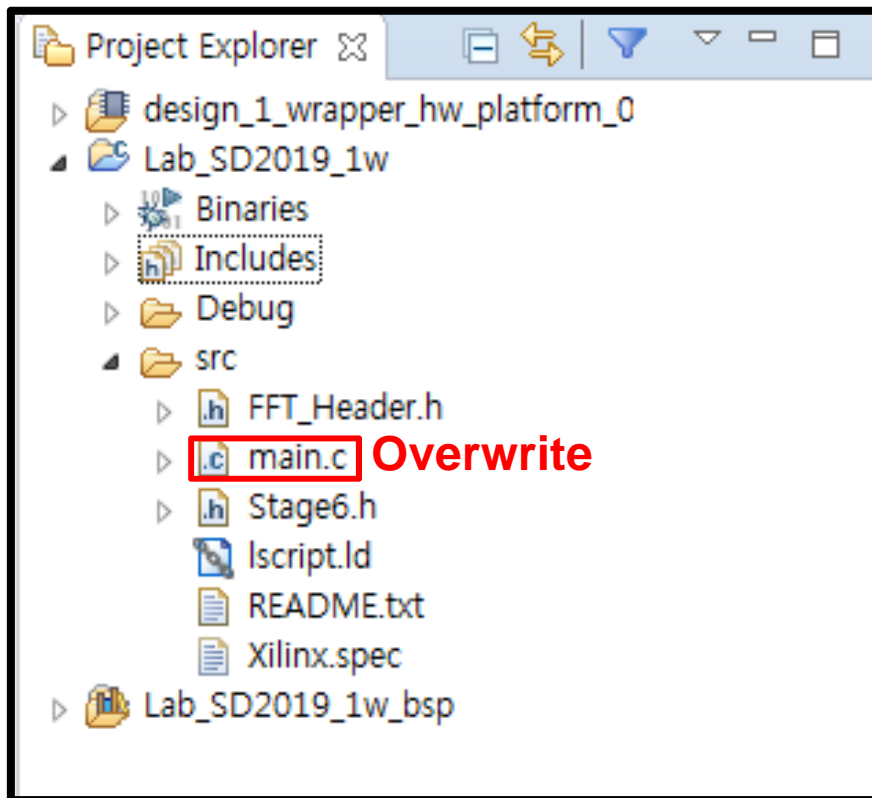
- Figure out the reason for the performance gain.

Programming in Assembly

- ❑ Copy the following file and paste it into the '**src**' folder.



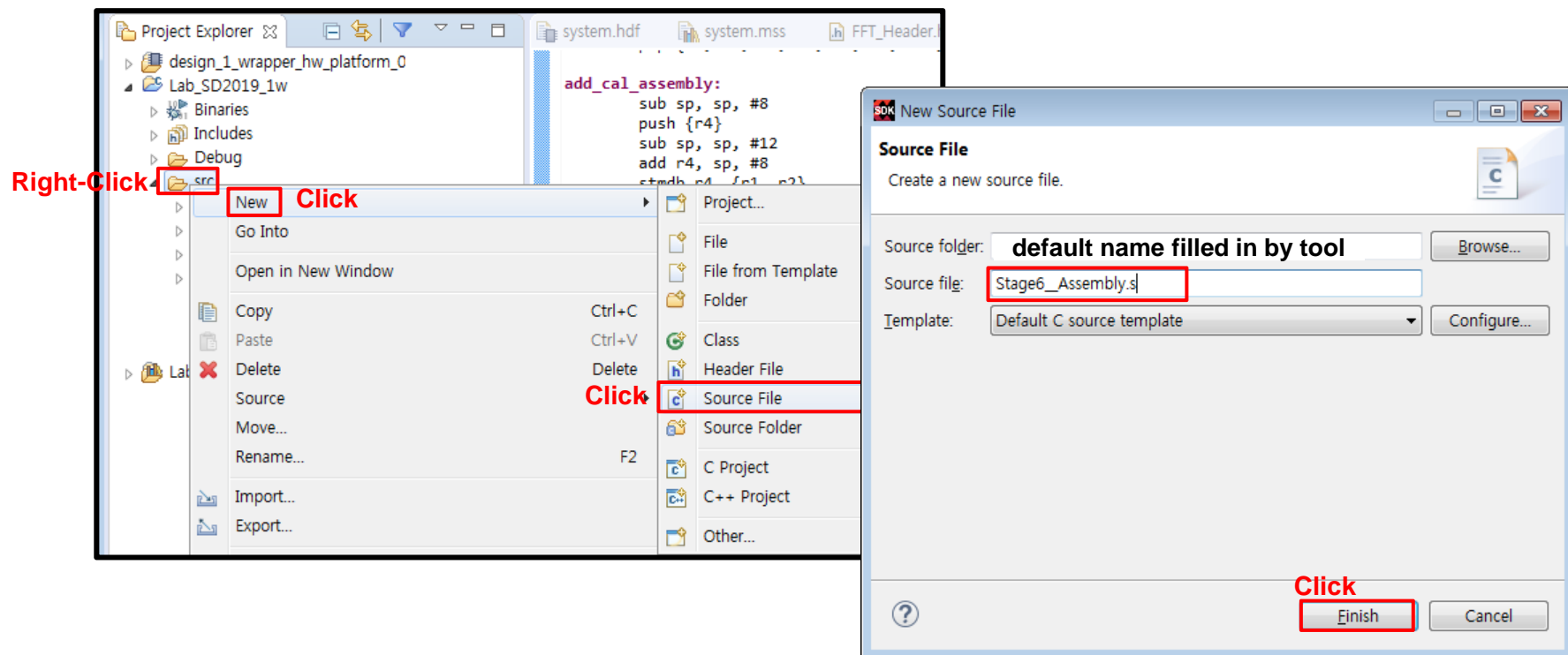
main.c.3



Programming in Assembly

❑ Add assembly source files

- Click '*src > New > Source File*'.
- Type '*Stage6_Assembly.s*' (using file extension '*.s*')



Programming in Assembly

□ Review the function 'main()'

- ① Calls '*FFT_Assembly()*'
- ② Compares the output of '*FFT()*' with that of '*DFT()*'

```
int main() {
    XTime start, stop;
    int i = 0;

    float error_total, error_real, error_imag;
    float sig_total;
    float SNR;

    XTime_GetTime((XTime*)&start);
    DFT();
    XTime_GetTime((XTime*)&stop);
    printf("DFT          %8.3f us\n", ((float)stop - (float)start)/COUNTS_PER_SECOND*1000000);

    XTime_GetTime((XTime*)&start);
    FFT();
    XTime_GetTime((XTime*)&stop);
    printf("FFT          %8.3f us\n", ((float)stop - (float)start)/COUNTS_PER_SECOND*1000000);

    XTime_GetTime((XTime*)&start);
    FFT_Assembly();
    XTime_GetTime((XTime*)&stop);
    printf("FFT Assembly %8.3f us\n", ((float)stop - (float)start)/COUNTS_PER_SECOND*1000000);

    error_total = 0;
    sig_total = 0;
    for(i = 0; i < N; i++){
        error_real = (X_DFT[i].re) - (X_FFT[i].re);
        error_imag = (X_DFT[i].im) - (X_FFT[i].im);

        error_total += error_real*error_real + error_imag*error_imag;

        sig_total += (X_DFT[i].re)*(X_DFT[i].re) + (X_DFT[i].im)*(X_DFT[i].im);
    }
    SNR = 10*log10(sig_total/error_total);
    xil_printf("FFT model SNR : %d dB\n", (int)SNR);

    error_total = 0;
    sig_total = 0;
    for(i = 0; i < N; i++){
        error_real = (X_DFT[i].re) - (X_FFT_Assembly[i].re);
        error_imag = (X_DFT[i].im) - (X_FFT_Assembly[i].im);

        error_total += error_real*error_real + error_imag*error_imag;

        sig_total += (X_DFT[i].re)*(X_DFT[i].re) + (X_DFT[i].im)*(X_DFT[i].im);
    }
    SNR = 10*log10(sig_total/error_total);
    xil_printf("FFT Assembly model SNR : %d dB\n", (int)SNR);

    return 0;
}
```

Programming in Assembly

□ Review the function 'FFT_Assembly()'

- ① Butterfly: Stage 1 ~ Stage 5
- ② Butterfly: Stage 6 (*incomplete*)
- ③ Reordering

```
#ifndef STAGE6_H_
#define STAGE6_H_

void Stage6(complex *output, complex *input)
{
    int n;

    //////////////////////////////////////
    //
    //      Fill Your Code Here.
    //
    //////////////////////////////////////

}

#endif /* STAGE6_H_ */
```

```
void FFT()
{
    complex input[N], temp[N];

    int out_re[N];
    int out_im[N];

    int data;
    int n, k;

    for (data=0; data<N; data++)
    {
        input[data].re=in_real[data];
        input[data].im=in_imag[data];
    }

    for (n=0; n<32; n++) //stage1
    {
        temp[n]=add_cal(input[n], input[n+32]);
        temp[n+32]=multiple(sub_cal(input[n], input[n+32]), W[n]);
    }

    for (n=0; n<16; n++) //stage2
    {
        for (k=0; k<2; k++)
        {
            input[n+(32*k)]=add_cal(temp[n+(32*k)], temp[n+((32*k)+16)]);
            input[n+((32*k)+16)]=multiple(sub_cal(temp[n+(32*k)], temp[n+((32*k)+16)]), W[2*n]);
        }
    }

    for (n=0; n<8; n++) //stage-3
    {
        for (k=0; k<4; k++)
        {
            temp[n+(16*k)] = add_cal(input[n+(16*k)], input[n+((16*k)+8)]);
            temp[n+((16*k)+8)] = multiple(sub_cal(input[n+(16*k)], input[n+((16*k)+8)]), W[4*n]);
        }
    }

    for (n=0; n<4; n++) //stage4
    {
        for (k=0; k<8; k++)
        {
            input[n+(8*k)] = add_cal(temp[n+(8*k)], temp[n+((8*k)+4)]);
            input[n+((8*k)+4)] = multiple(sub_cal(temp[n+(8*k)], temp[n+((8*k)+4)]), W[8*n]);
        }
    }

    for (n=0; n<2; n++) //stage5
    {
        for (k=0; k<16; k++)
        {
            temp[n+(4*k)]=add_cal(input[n+(4*k)], input[n+(4*k+2)]);
            temp[n+(4*k+2)]=multiple(sub_cal(input[n+(4*k)], input[n+(4*k+2)]), W[16*n]);
        }
    }

    // Stage 6
    Stage6(input, temp);

    for (n=0; n<N; n++)
    {
        X_FFT[n]=input[Re_ordering(n)];
    }

    for (n=0; n<N; n++)
    {
        out_im[n]=(X_FFT[n].im)>>10;
        out_re[n]=(X_FFT[n].re)>>10;
    }
}
```

Programming in Assembly

- ❑ Use the following assembly program



Stage6_Assembly.s

```
.text
.syntax unified

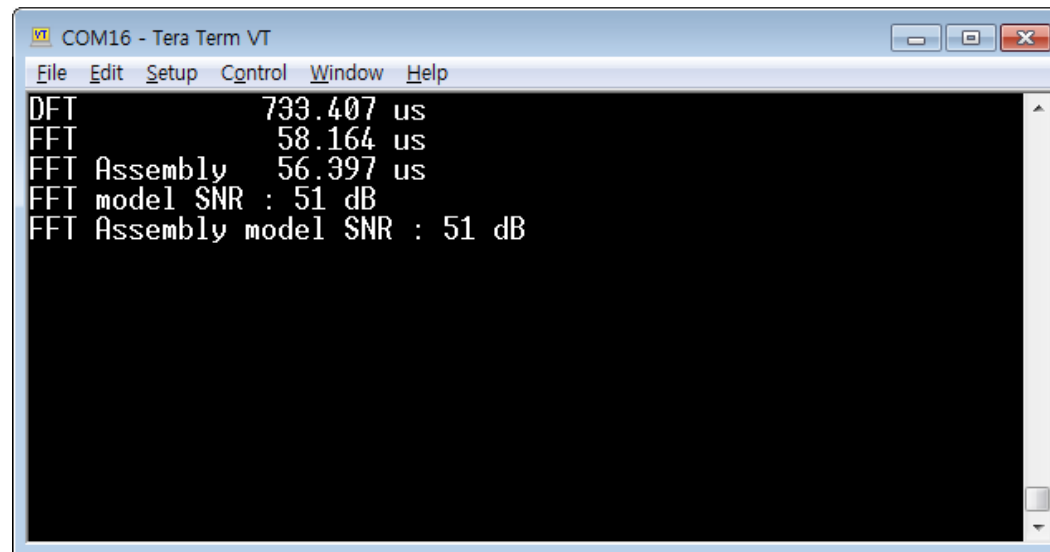
.align 4
.global Stage6_Assembly
.arm

Stage6_Assembly:
    push {r4, r5, r6, r7, r8, r9, r10, r11, lr}
    sub sp, sp, #20
    mov r9, r0
    mov r10, r1
    mov r4, #0
    add r7, r1, #8
    add r5, sp, #8
    add r6, r4, r10
    add r8, r7, r4
    ldr r3, [r8, #4]
    str r3, [sp]
    ldr r3, [r7, r4]
    mov r0, r5
    ldm r6, {r1, r2}
    bl add_cal_assembly
    add r3, r4, r9
    ldm r5, {r0, r1}
    stm r3, {r0, r1}
    add r11, r3, #8
    ldr r3, [r8, #4]
    str r3, [sp]
    ldr r3, [r7, r4]
    mov r0, r5
    ldm r6, {r1, r2}
    bl sub_cal_assembly
    ldm r5, {r0, r1}
    stm r11, {r0, r1}
    add r4, r4, #16
    cmp r4, #512
    bne Stage6_Assembly+28
    add sp, sp, #20
    pop {r4, r5, r6, r7, r8, r9, r10, r11, pc}
```

Programming in Assembly

❑ Run the application

- Check the output of the application on '*Tera Term*'
 - ✓ Check the performance gain.
 - ✓ Check the accuracy given by SNR



A screenshot of a Tera Term window titled 'COM16 - Tera Term VT'. The window has a menu bar with 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. The main text area displays the following output:

```
DFT          733.407 us
FFT          58.164 us
FFT Assembly 56.397 us
FFT model SNR : 51 dB
FFT Assembly model SNR : 51 dB
```

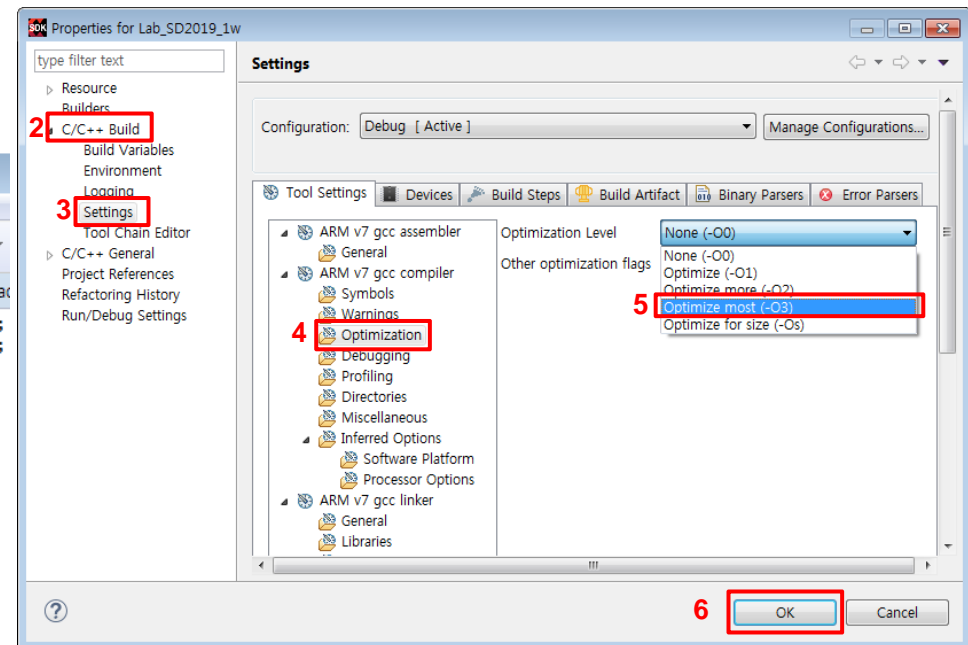
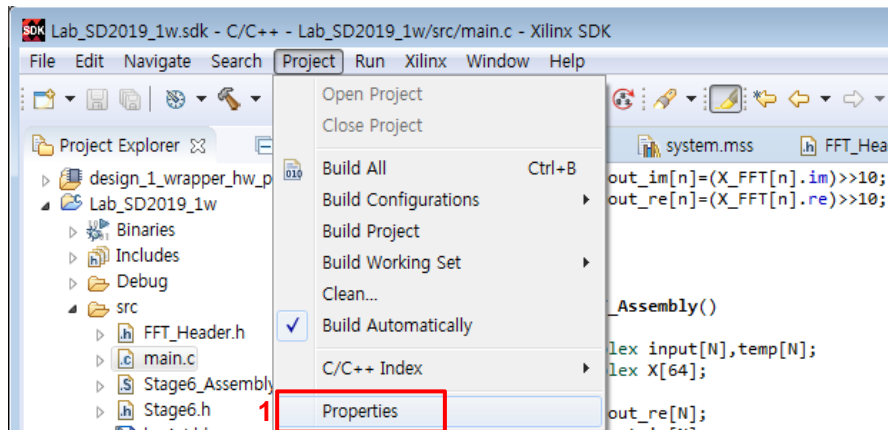
Programming in Assembly

☐ Debug the application

- Repeat the steps depicted in pp. 20~27 to reason for the performance gain.

Setting Optimization Level

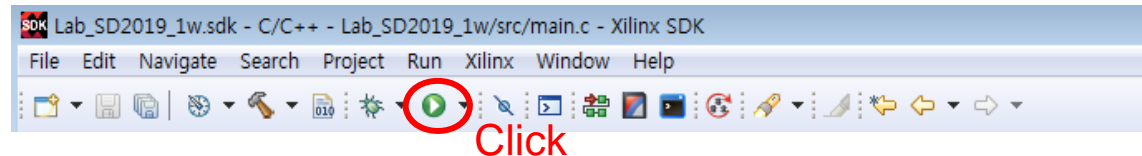
- ❑ Set the compiler optimization level
 - Select '**Project**' menu and click '**Properties**'
 - Select '**Settings**' tab and click '**ARM v7 gcc compiler > Optimization**'
 - Select '**Optimization most (-O3)**' in the dropdown menu of '**Optimization Level**'
 - Click '**OK**'



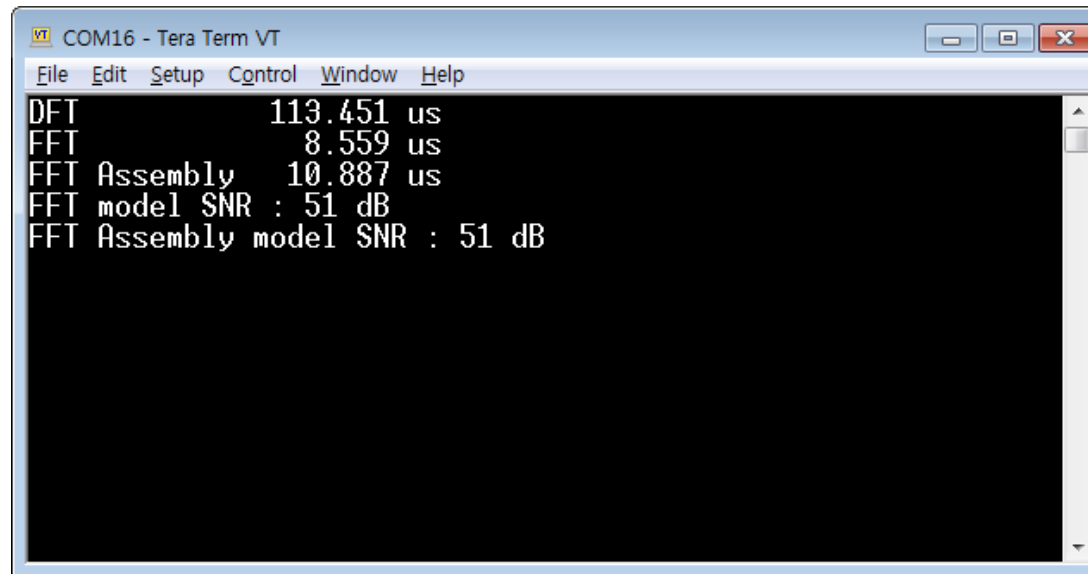
Setting Optimization Level

□ Run the application

- Click the '**Run As...**' icon to run the application again



- Check the output of the application on '**Tera Term**'
 - ✓ Check how much it accelerates the application



Setting Optimization Level

☐ Debug the application

- Repeat the steps depicted in pp. 20~27 to figure out the impact of the compiler optimization level on the assembly codes

Demo

- ❑ Compare the (normalized) execution times of all the **three** FFT implementations as follows
 - Optimization level: O0
 - ✓ D-cache disabled/enabled
 - Optimization level: O3
 - ✓ D-cache disabled/enabled
- ❑ Figure out the reasons for the measured speedup