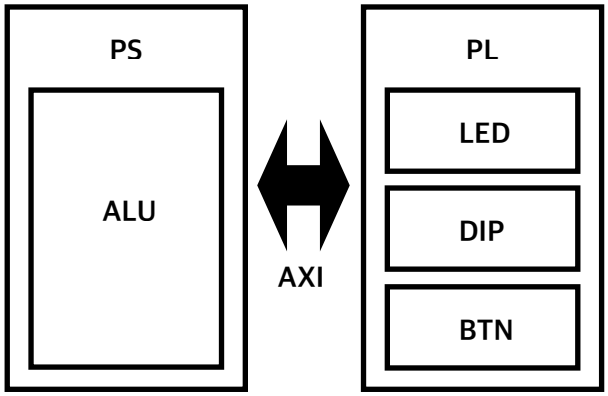
2022.05.10

전자공학과 유종상
60171851

SoC HW1 보고서



본 과제는 PL의 DIP SW, BTN을 입력을 받아 PS상의 ALU로 연산을 하고 다시 PL의 LED로 2진 출력을 하는 과제이다. PL과 PS간의 BUS는 AXI로 구성하는 것이 그 특징이다. 좌측 도표를 참고해주길 바란다.

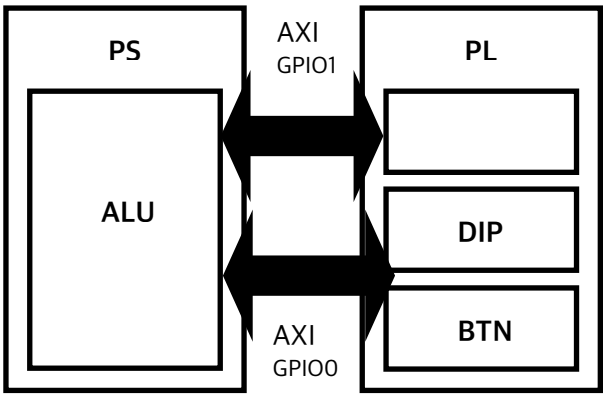
이를 구현하기 위해서 우선 Vivado에서 Block design이 선행될 필요가 있다. 이 중 우리는 Block design 요소중 AXI GPIO라는 Block에 관심을 가지고자 한다.

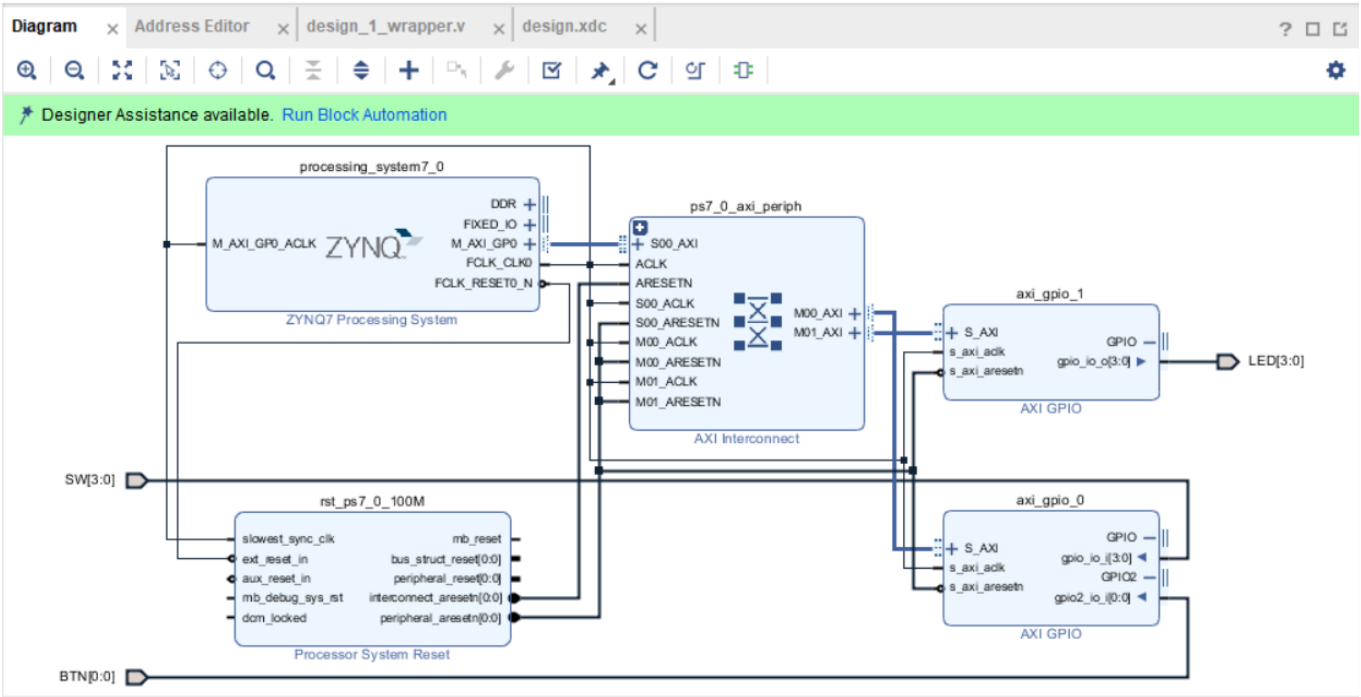
Design Constraints상에서 명칭을 변경하여 사용한다면 한 AXI GPIO 1개의 IP에서 LED, DIP-SW, BTN 모두 묶는 것이 가능하다. 하지만, 이렇게 되면 가독성이 나빠지는 문제가 발생한다.

이에 본 과제에서는 AXI GPIO를 2개 사용할 예정이다. AXI GPIO는 Dual channel 로 구성가능하다.

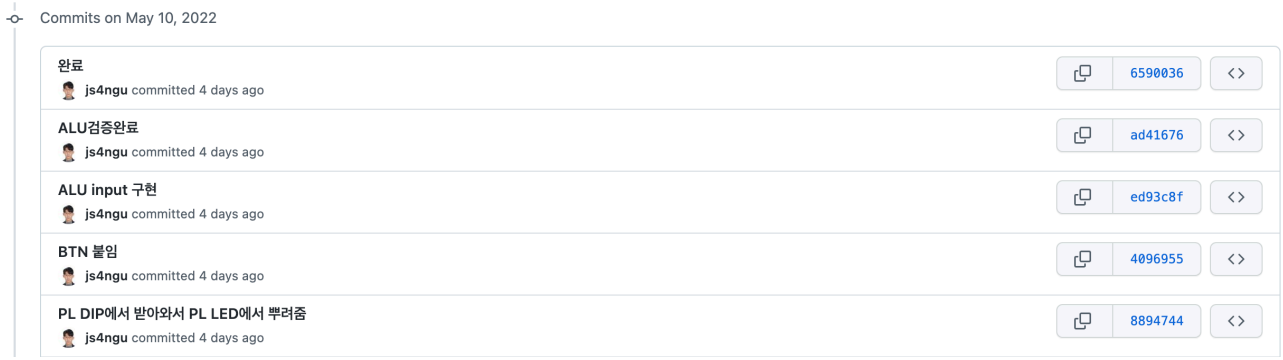
AXI-GPIO0는 DIP, BTN과 같은 input 만을 담당하는 GPIO로서 Dual 채널로 구성하여 BTN과 DIP-SW를 입력 받을 예정이다.
AXI-GPIO1의 경우 LED와 같은 Output 만을 담당하는 GPIO로서 기능하게 구성해주었다.

우측 도표를 참고해주길 바란다.
위 내용을 구현하기 위해서 Vivado Block design상에서 아래와 같이 설정해주었다.





해당 Block design대로 HW를 구성해주었다. PS의 ALU는 SDK상에서 개발하였고, 주기적으로 Git에 commit하여 Code를 관리하였다. 아래는 본 과제를 수행하면서 Commit된 log이다.



아래 항목에서 각 Commit 별로 Code 분석과 발생했던 Issue들을 소개한다

 # PL DIP에서 받아와서 PL LED에서 뿌려줌

```c
#include <stdio.h>
#include "platform.h"
#include "xgpio.h"  //GPIO PL(LED, BTN, Dip switch) 사용하기 위한 header file
#include "xgpiops.h"
#include "sleep.h"  //usleep 사용을 위한 header file

int main() {
    static XGpio input, output;
    //static XGpioPs psGpioInstance_ptr;  //PS의 GPIO를 사용하기 위한 pointer
    //static XGpioPs_Config *GpioConfigPtr;      //PS의 GPIO 설정을 위한 pointer
    unsigned int Result = 0b0110;      // ALU 결과를 저장할 register
    unsigned int DIP_SW_data = 0;
    unsigned int BTN_data = 0;
    int xStatus;

    init_platform();  //UART 및 cache 초기화

    //GPIO 0 초기화 : input 부분
    //AXI GPIO_SW Initialization
    //AXI GPIO_Button Initialization
    xStatus = XGpio_Initialize(&input, XPAR_AXI_GPIO_0_DEVICE_ID);
    if(XST_SUCCESS != xStatus)
        print("GPIO_SW INIT FAILDED \n\r");

    //GPIO 1 초기화 : output 부분
    //AXI GPIO_LED Initialization
    xStatus = XGpio_Initialize(&output,XPAR_AXI_GPIO_1_DEVICE_ID);
    if(XST_SUCCESS != xStatus)
        print("GPIO_LED INIT FAILDED \n\r");

    //AXI GPIO 입출력 설정
    XGpio_SetDataDirection(&input,1,1);  // 1 : 1st channel, 1 : 입력 -> DIP SW
    //XGpio_SetDataDirection(&input,2,1);  // 1 : 2st channel, 1 : 입력 -> BTN
    XGpio_SetDataDirection(&output,1,0); // 1 : single channel 0 : 출력

    while(1) {
        DIP_SW_data = XGpio_DiscreteRead(&input, 1);  //switch에서 입력
        //BTN_data = XGpio_DiscreteRead(&input, 2);     //BTN에서 입력
        printf("%d\n", DIP_SW_data);
        XGpio_DiscreteWrite(&output, 1 , DIP_SW_data);       //LED OUT
        usleep(1000);
    }
    cleanup_platform();
    return 0;
}
```

우선은, DIP_SW 입력을 받아 LED에 출력해주는 Code를 작성하였다. AXI GPIO를 2개 사용하기에, XGpio 포인터를 input, output 2개 선언해주었다. GPIO 입출력설정 부분(XGpio_SetDataDirection())에서 Input, output GPIO의 입출력 방향에 맞게 적절한 인자를 부여했다. DIP SW로 받는 값은 그대로 LED 로 출력하게 코드를 작성하였다.

그리고 편리한 디버깅을 위해 DIP SW로 받는 값을 UART로 뿌려주는 코드 또한 작성하였다.

 # BTN 붙임

```
#include <stdio.h>
#include "platform.h"
#include "xgpio.h"  //GPIO PL(LED, BTN, Dip switch) 사용하기 위한 header file
#include "xgpiops.h"#include "sleep.h"  //usleep 사용을 위한 header file

int main() {
    static XGpio input, output;
    //static XGpioPs psGpioInstance_ptr;  //PS의 GPIO를 사용하기 위한 pointer
    //static XGpioPs_Config *GpioConfigPtr;      //PS의 GPIO 설정을 위한 pointer
    unsigned int Result = 0b0110;      // ALU 결과를 저장할 register
    unsigned int DIP_SW_data = 0;
    unsigned int BTN_data = 0;
    int xStatus;

    init_platform();  //UART 및 cache 초기화

    //GPIO 0 초기화 : input 부분
    //AXI GPIO_SW Initialization
    //AXI GPIO_Button Initialization
    xStatus = XGpio_Initialize(&input, XPAR_AXI_GPIO_0_DEVICE_ID);
    if(XST_SUCCESS != xStatus)
        print("GPIO_SW INIT FAILDED \n\r");

    //GPIO 1 초기화 : output 부분
    //AXI GPIO_LED Initialization
    xStatus = XGpio_Initialize(&output,XPAR_AXI_GPIO_1_DEVICE_ID);
    if(XST_SUCCESS != xStatus)
        print("GPIO_LED INIT FAILDED \n\r");

    //AXI GPIO 입출력 설정
    XGpio_SetDataDirection(&input,1,1);  // 1 : 1st channel, 1 : 입력 -> DIP SW
    XGpio_SetDataDirection(&input,2,1);  // 1 : 2st channel, 1 : 입력 -> BTN
    XGpio_SetDataDirection(&output,1,0); // 1 : single channel 0 : 출력

    while(1) {
        DIP_SW_data = XGpio_DiscreteRead(&input, 1);  //switch에서 입력
        BTN_data = XGpio_DiscreteRead(&input, 2);     //BTN에서 입력
        printf("%d %d\n", DIP_SW_data, BTN_data);
        XGpio_DiscreteWrite(&output, 1 , Result);       //LED OUT
        usleep(1000);
    }
    cleanup_platform();
    return 0;
}
```

GPIO 0 2번째 채널에 붙어있는 BTN을 사용하기 위해 GPIO 입출력 방향 설정 함수(XGpio_SetDataDirection()) 에서 적절한 인자를 부여해주었다. 또한, While 문 안에서는 기존의 코드와 더불어 편리한 디버깅을 위해 DIP, BTN 으로 입력되는 값을 뿌려주는 UART 코드를 작성하였다.

이 과정에서 몇가지 혼동되는 사항이 있었다. 바로 GPIO 초기화 함수(XGpio_Initialize())의 디바이스 ID 부분이다. 해당 함수 2번째 인자에는 디바이스 ID를 기입해줘야한다. 해당 파라메타들이 정의된 파일을 확인해보니 XPAR_AXI_GPIO_0_IS_DUAL 이라는 인자가 존재하였다. 듀얼 채널 GPIO인 AXI GPIO0 이기에 해당 파라메타를 넣어줘야할 줄 알았으나, 해당 파라메타가 아니나 XPAR_AXI_GPIO_0_DEVICE_ID 를 사용해야 정상 작동했다.

이 과정에서 2가지 궁금증이 발생하였는데 해결하지는 못하였다. 해당 궁금증은 아래와 같다.
1. XPAR_AXI_GPIO_0_IS_DUAL 인자는 언제 사용하는지?
2. XPAT_AXI_GPIOx 인자는 0과 1만 존재한다. 그럼 AXI_GPIO3 는 못사용하는지?

ALU input 구현

```c
#include <stdio.h>
#include "platform.h"
#include "xgpio.h"  //GPIO PL(LED, BTN, Dip switch) 사용하기 위한 header file
#include “xgpiops.h"
#include "sleep.h"  //usleep 사용을 위한 header file

int ALU(unsigned int DIP, unsigned int BTN);

int main() {
    static XGpio input, output;
    //static XGpioPs psGpioInstance_ptr;  //PS의 GPIO를 사용하기 위한 pointer
    //static XGpioPs_Config *GpioConfigPtr;  //PS의 GPIO 설정을 위한 pointer
    unsigned int result = 0b0110;  // ALU 결과를 저장할 register
    unsigned int DIP_SW_data = 0;
    unsigned int BTN_data = 0;
    int xStatus;

    init_platform();  //UART 및 cache 초기화

    //GPIO 0 초기화 : input 부분
    //AXI GPIO_SW Initialization
    //AXI GPIO_Button Initialization
    xStatus = XGpio_Initialize(&input, XPAR_AXI_GPIO_0_DEVICE_ID);
    if(XST_SUCCESS != xStatus)
        printf("GPIO_SW INIT FAILDED \n\r");

    //GPIO 1 초기화 : output 부분
    //AXI GPIO_LED Initialization
    xStatus = XGpio_Initialize(&output,XPAR_AXI_GPIO_1_DEVICE_ID);
    if(XST_SUCCESS != xStatus)
        printf("GPIO_LED INIT FAILDED \n\r");

    //AXI GPIO 입출력 설정
    XGpio_SetDataDirection(&input,1,1);  // 1 : 1st channel, 1 : 입력 -> DIP SW
    XGpio_SetDataDirection(&input,2,1);  // 1 : 2st channel, 1 : 입력 -> BTN
    XGpio_SetDataDirection(&output,1,0); // 1 : single channel 0 : 출력

    while(1) {
        DIP_SW_data = XGpio_DiscreteRead(&input, 1);  //switch에서 입력
        BTN_data = XGpio_DiscreteRead(&input, 2); //BTN에서 입력
        //printf("%d %d\n", DIP_SW_data, BTN_data);
        result = ALU(DIP_SW_data, BTN_data);
        XGpio_DiscreteWrite(&output, 1 , result);   //LED OUT
        usleep(1000);
    }
    cleanup_platform();
    return 0;
}

int ALU(unsigned int DIP, unsigned int BTN) {
    switch(DIP){
        case 0b0000 :
            printf("%d %d\n", DIP, BTN);
        case 0b0100 :
            printf("%d %d\n", DIP, BTN);
        case 0b1000 :
            printf("%d %d\n", DIP, BTN);
        case 0b1100 :
            printf("%d %d\n", DIP, BTN);
        case 0b0001 :
            printf("%d %d\n", DIP, BTN);
        case 0b0101 :
            printf("%d %d\n", DIP, BTN);
        case 0b1001 :
            printf("%d %d\n", DIP, BTN);
        case 0b1101 :
            printf("%d %d\n", DIP, BTN);
        case 0b0010 :
            printf("%d %d\n", DIP, BTN);
        case 0b0110 :
            printf("%d %d\n", DIP, BTN);
        case 0b1010 :
            printf("%d %d\n", DIP, BTN);
        case 0b1110 :
            printf("%d %d\n", DIP, BTN);
        case 0b0011 :
            printf("%d %d\n", DIP, BTN);
        case 0b0111 :
            printf("%d %d\n", DIP, BTN);
        case 0b1011 :
            printf("%d %d\n", DIP, BTN);
        case 0b1111 :
            printf("%d %d\n", DIP, BTN);
    }
    return 0b0110;
}
```

ALU 함수를 선언하였다. 아직은 단순히 함수에 DIP SW와 BTN 변수를 받아 UART로 출력해주는 기능 만 구현하였다. LED 출력은 기본 출력인 0110으로 설정해두었다.

 # ALU 검증완료
ALU 함수 내용과 usleep 인자만 1초로 설정하는 내용 만 달라졌기에 ALU 함수 정의 부분 만 첨부하겠다.

```
int ALU(unsigned int DIP, unsigned int BTN, unsigned int RESULT) {
    unsigned int output;
    if (BTN == 1) {
       output = 0b0110;
    }
    else if(DIP == 0b0000) output = RESULT + 1;
    else if(DIP == 0b0100) output = RESULT + 2;
    else if(DIP == 0b1000) output = RESULT + 3;
    else if(DIP == 0b1100) output = RESULT + 4;
    else if(DIP == 0b0001) output = RESULT - 1;
    else if(DIP == 0b0101) output = RESULT - 2;
    else if(DIP == 0b1001) output = RESULT - 3;
    else if(DIP == 0b1101) output = RESULT - 4;
    else if(DIP == 0b0010) output = RESULT * 1;
    else if(DIP == 0b0110) output = RESULT * 2;
    else if(DIP == 0b1010) output = RESULT * 3;
    else if(DIP == 0b1110) output = RESULT * 4;
    else if(DIP == 0b0011) output = RESULT << 1;
    else if(DIP == 0b0111) output = RESULT << 2;
    else if(DIP == 0b1011) output = RESULT << 3;
    else if(DIP == 0b1111) output = RESULT << 4;
    printf("%d %d\n", DIP, output%16);
    return output % 16;
}
```

STEP	OPCODE	OPERATION	MEANING
00	00	Addition MOD 16	R = R + 1
01	00	Addition MOD 16	R = R + 2
10	00	Addition MOD 16	R = R + 3
11	00	Addition MOD 16	R = R + 4
00	01	Subtraction MOD 16	R = R – 1
01	01	Subtraction MOD 16	R = R – 2
10	01	Subtraction MOD 16	R = R – 3
11	01	Subtraction MOD 16	R = R – 4
00	10	Multiplication MOD 16	R = R x 2
01	10	Multiplication MOD 16	R = R x 3
10	10	Multiplication MOD 16	R = R x 4
11	10	Multiplication MOD 16	R = R x 5
00	11	Rotation Left	R = R << 1
01	11	Rotation Left	R = R << 2
10	11	Rotation Left	R = R << 3
11	11	Rotation Left	R = R << 4

기존 커밋대비 5가지 개선사항이 있다.

1.Switch문에서 if-else 문으로 변경
Switch문 보다는 if-else 문이 익숙하기에 변경하였다. PL의 경우 If-else문을 사용하면 MUX 합성에 손해가 있을 있지만, PS이기에 진행하였다.

2. ALU 연산 테이블 구현
과제문에서 제공하는 STEP과 OPCODE를 참조하여 if-else 분기를 진행하였고 분기후의 연산을 구현하였다.

3. 연산 결과 LED 출력
기존에는 ALU 연산결과와 LED 출력이 연동되지 않았지만 연동할 수 있게 구현하였다.

4. ALU 함수에 result 참조
ALU 함수는 기본적으로 과거의 결과를 대상으로 연산한다. 과거의 result를 참조하게 구현하였다.

5. Overflow 방지
기존의 코드는 LED의 표현 한계인 1111(2) 를 초과할 가능성이 다분하다. 출력에 대해 %16 연산을 하여 overflow를 방지하였다. 그 결과 문제 없이 작동하였다. 다음은 전체 코드이다.

```c
#include <stdio.h>
#include "platform.h"
#include "xgpio.h"  //GPIO PL(LED, BTN, Dip switch) 사용하기 위한 header file
#include “xgpiops.h"
#include "sleep.h"  //usleep 사용을 위한 header file

int ALU(unsigned int DIP, unsigned int BTN, unsigned int RESULT);

int main() {
    static XGpio input,output;
    //static XGpioPs psGpioInstance_ptr;  //PS의 GPIO를 사용하기 위한 pointer
    //static XGpioPs_Config *GpioConfigPtr;  //PS의 GPIO 설정을 위한 pointer
    unsigned int result = 6;  // ALU 결과를 저장할 register
    unsigned int DIP_SW_data = 0;
    unsigned int BTN_data = 0;
    int xStatus;

    init_platform();  //UART 및 cache 초기화

    //GPIO 0 초기화 : input 부분
    //AXI GPIO_SW Initialization
    //AXI GPIO_Button Initialization
    xStatus = XGpio_Initialize(&input, XPAR_AXI_GPIO_0_DEVICE_ID);
    if(XST_SUCCESS != xStatus)
        printf("GPIO_SW INIT FAILDED \n\r");

    //GPIO 1 초기화 :output 부분
    //AXI GPIO_LED Initialization
    xStatus = XGpio_Initialize(&output,XPAR_AXI_GPIO_1_DEVICE_ID);
    if(XST_SUCCESS != xStatus)
        printf("GPIO_LED INIT FAILDED \n\r");

    //AXI GPIO 입출력 설정
    XGpio_SetDataDirection(&input,1,1);  // 1 : 1st channel, 1 : 입력 -> DIP SW
    XGpio_SetDataDirection(&input,2,1);  // 1 : 2st channel, 1 : 입력 -> BTN
    XGpio_SetDataDirection(&output,1,0); // 1 : single channel 0 : 출력

    while(1) {
        DIP_SW_data = XGpio_DiscreteRead(&input, 1);  //switch에서 입력
        BTN_data = XGpio_DiscreteRead(&input, 2); //BTN에서 입력
        result = ALU(DIP_SW_data, BTN_data, result);
        XGpio_DiscreteWrite(&output, 1 , result);   //LED OUT
        usleep(1000000);
    }
    cleanup_platform();
    return 0;
}

int ALU(unsigned int DIP, unsigned int BTN, unsigned int RESULT) {
    unsigned int output;
    if (BTN == 1) {
       output = 0b0110;
    }
    else if(DIP == 0b0000) output = RESULT + 1;
    else if(DIP == 0b0100) output = RESULT + 2;
    else if(DIP == 0b1000) output = RESULT + 3;
    else if(DIP == 0b1100) output = RESULT + 4;
    else if(DIP == 0b0001) output = RESULT - 1;
    else if(DIP == 0b0101) output = RESULT - 2;
    else if(DIP == 0b1001) output = RESULT - 3;
    else if(DIP == 0b1101) output = RESULT - 4;
    else if(DIP == 0b0010) output = RESULT * 1;
    else if(DIP == 0b0110) output = RESULT * 2;
    else if(DIP == 0b1010) output = RESULT * 3;
    else if(DIP == 0b1110) output = RESULT * 4;
    else if(DIP == 0b0011) output = RESULT << 1;
    else if(DIP == 0b0111) output = RESULT << 2;
    else if(DIP == 0b1011) output = RESULT << 3;
    else if(DIP == 0b1111) output = RESULT << 4;
    printf("%d %d\n", DIP, output%16);
    return output % 16;
}
```