

Practice #8. OS + FPGA system
Jiwon Lee, Sangjun Son

Goal

- Implement block design & execute in zedboard.
 - Running Debian Linux on zedboard.
 - Running a sample C program.

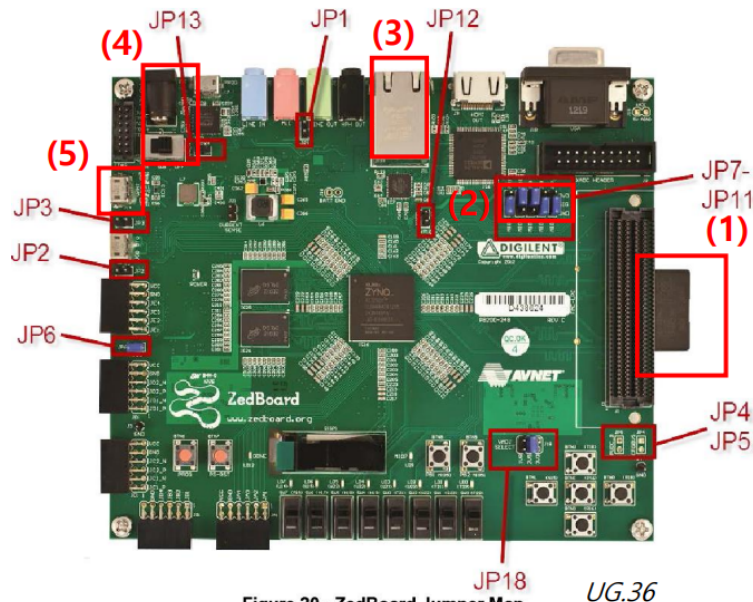


Figure 20 - ZedBoard Jumper Map

UG.36

Figure 1: 이번 실습에서 사용한 Zedboard. 이전 세션과는 달리 보드를 호스트로 사용하여 실습을 진행한다. 사각 형으로 표시된 부분을 우선으로 연결하여 보드 환경을 만들어 주었다.

1 Implementation

이번 프로젝트는 Vivado block design을 사용하여 FPGA위에 Processing System과 Bram을 적절히 연결하여 올리는 Bitstream 파일을 만들었다. 만들어진 Bitstream 파일을 SD 카드에 저장 후 Figure 1의 (1)에 해당하는 SD 카드 슬롯에 삽입하고 (2)의 SD boot mode를 01100으로 설정해주면 Zedboard 자체를 호스트로 사용할 수 있다. OS (Linux debian-zynq 4.0.0-xilinx)를 부팅하고, 그 위에서 Zedboard의 주변장치들을 사용할 수 있다 [1].

특히, block design을 통하여 BRAM과 이외의 custom IP를 사용할 수 있고, 이를 C의 mmap함수를 통하여 실제로 사용해 본다. 기 주어진 Bitstream 파일을 이용해 실제로 FPGA에서 동작 시켜보고 주어진 main.c을 이용해 Figure 2의 모듈 MyIP가 어떤 일을 하는지 확인해보았다.

Practice #8. OS + FPGA system
Jiwon Lee, Sangjun Son

1.1 About main.c

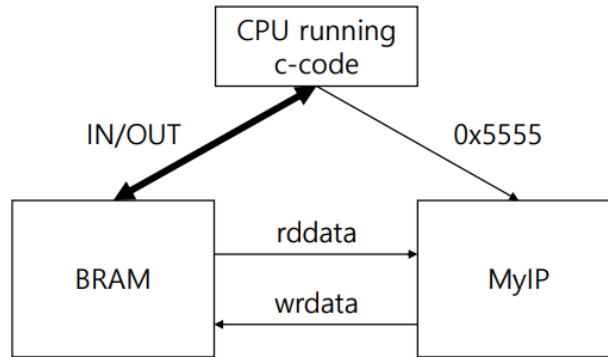


Figure 2: C program에서 사용되는 BRAM과 Custom IP의 아키텍처 [1]

CPU와 BRAM, IP (unknown)으로 구성되어 있고, IP는 BRAM의 데이터를 읽고, BRAM에 데이터를 써주는 역할을 한다는 것을 알 수 있다. 아래는 실습을 위해 주어진 main.c 소스코드이다.

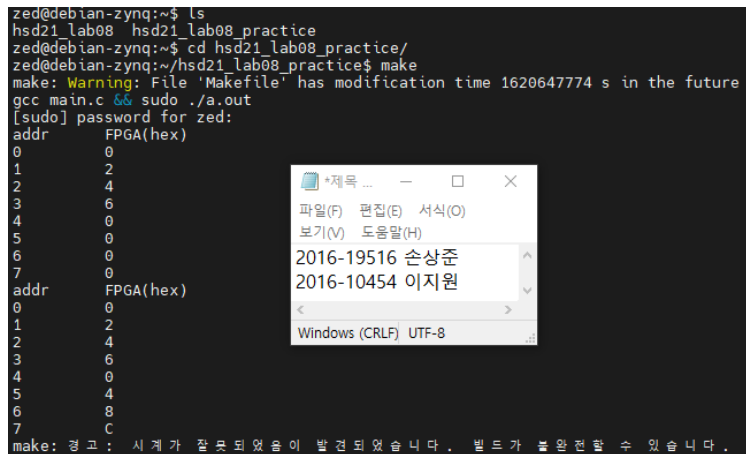
main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <fcntl.h>
5 #include <sys/mman.h>
6
7 #define SIZE 4
8
9 int main(int argc, char** argv)
10 {
11     int i;
12
13     int foo = open("/dev/mem", O_RDWR);
14     int *fpga_bram = mmap(NULL, SIZE * sizeof(int), PROT_READ|PROT_WRITE, MAP_SHARED, foo, 0x40000000);
15     int *fpga_ip = mmap(NULL, sizeof(int), PROT_READ|PROT_WRITE, MAP_SHARED, foo, 0x43C00000);
16
17     // initialize memory
18     for (i = 0; i < SIZE; i++)
19         *(fpga_bram + i) = (i * 2);
20     for (i = SIZE; i < SIZE * 2; i++)
21         *(fpga_bram + i) = 0.0f;
22
23     printf("%-10s%-10s\n", "addr", "FPGA(hex)");
24     for (i = 0; i < SIZE * 2; i++)
25         printf("%-10d%-10X\n", i, *(fpga_bram + i));
26
27     // run ip
28     *(fpga_ip) = 0x5555;
29     while (*(fpga_ip) == 0x5555);
30
31     printf("%-10s%-10s\n", "addr", "FPGA(hex)");
32     for (i = 0; i < SIZE * 2; i++)
33         printf("%-10d%-10X\n", i, *(fpga_bram + i));
34
35     return 0;
36 }
```

Practice #8. OS + FPGA system
Jiwon Lee, Sangjun Son

- file descriptor `foo`를 선언하고, `mmap`함수를 통해 BRAM, IP (unknown) 를 메모리에 대응시킨다 (13-15 라인).
- BRAM memory에 `BRAM[0]` 부터 `BRAM[3]`까지는 각각 (0, 2, 4, 6)을, `BRAM[4]` 부터 `BRAM[7]`까지는 0.0f (= (int) 0) 을 대응시킨다 (18-21 라인).
- IP (unknown) 에 0x5555를 할당하여 signal을 주고, 다시 BRAM의 메모리를 출력시킨다 (28-33 라인).

2 Result



```
zed@debian-zynq:~$ ls
hsd21 lab08 hsd21_lab08 practice
zed@debian-zynq:~$ cd hsd21_lab08 practice/
zed@debian-zynq:~/hsd21_lab08 practice$ make
make: Warning: File 'Makefile' has modification time 1620647774 s in the future
gcc main.c -o sudo ./a.out
[sudo] password for zed:
addr      FPGA(hex)
0         0
1         2
2         4
3         6
4         0
5         0
6         0
7         0
addr      FPGA(hex)
0         0
1         2
2         4
3         6
4         0
5         4
6         8
7         C
make: 경고 : 시 계 가 잘못 되었 음 이 발견 되었 습 니 다 . 빌 드 가 불 완 전 할 수 있 습 니 다 .
```

Figure 3: `main.c`를 zedboard에서 실행시켰을 때의 결과화면

처음 (MyIP가 수행되기 전) 주소값 (`addr`) 0-7에 해당하는 BRAM에 저장된 값은 BRAM memory initialization을 수행한 후의 값이고, 두 번째 주소값 0-7에 따른 data값은 IP에 특정 시그널 (0x5555)을 주어 이를 실행시킨 후의 값이다.

Figure 2에서 주소 0, 1, 2, 3에 해당하는 BRAM 값이 0, 2, 4, 6으로 잘 초기화 된 것을 확인할 수 있다. 4, 5, 6, 7은 `while(*fpga_ip == 0x5555)`가 실행되기 전까지는 0으로 초기화 되었음을 확인하였다. 실행이 된 이후에 주소 4, 5, 6, 7에 해당하는 BRAM값은 0, 4, 8, C로 값이 바뀐 것을 볼 수 있다. 0, 4, 8, C는 0, 2, 4, 6에 2를 곱한 것의 16진수 표현이므로 MyIP가 2를 곱해서 저장해준다고 추론할 수 있다. IP가 실행되고 난 후 결과값을 살펴보았을 때, 주소값 4-7만 바뀌었음을 확인할 수 있고, 다음과 같은 규칙을 가진다고 말할 수 있다.

$$\text{BRAM}[i + 4] = \text{BRAM}[i] \times 2 \quad (0 \leq i \leq 3) \quad (1)$$

`main.c`에서의 `fpga_ip`의 값이 IP input signal이다. `fpga_ip`의 값을 바꿔가며 테스트를 했을 때, `addr` 0-3값은 바뀌지 않고, `addr` 4-7값만 바뀌었다. 또한, `addr` 4-7에 0이외의 다른 값들이 있더라도, 기존 값에 위의 수식에서 나온 결과값을 덮어쓰는 것을 확인할 수 있었다.

따라서, 프로그램에서 메모리에 매핑된 IP는 multiplier라고 생각하였고, 이 IP는 memory에 있는 `BRAM[0-3]` 값을 read 하여, 0x5555 signal을 넣어주면, `BRAM[0-3]` 값을 2배 곱하여 각각 `BRAM[4-7]`에 write 해주는 역할을 수행한다.

Practice #8. OS + FPGA system
Jiwon Lee, Sangjun Son

추가적으로, BRAM[0-3]의 초기값을 굳이 $i \times 2$ 의 값으로 초기화하지 않고 다른 값을 입력시켰을 때에도 Equation (1)의 규칙이 성립함을 검증하였다. `while(*fpga_ip == 0x5555)`의 값을 다른 값으로 바꾸었을 때 제대로 동작하지 않음을 확인하였다. 이를 통해 MyIP에서 0x5555 외의 다른 값이 주어졌을 때는 기능을 하지 않는다는 사실을 추측할 수 있었다.

3 Conclusion

이번 과제에서는 Zedboard 자체를 호스트로 사용하는 것을 실습해보았다. Debian Linux OS가 zynq 버전이 있는 것이 신기하였다. 또한, block design을 vivado에서 수행하고, 이를 OS에 올려 IP들이 돌아가는 과정을 확인할 수 있었는데, 임베디드 시스템의 동작방식도 이와 비슷할 것이라고 생각하였다.

References

- [1] Computing Memory Architecture Lab. *Practice 8: OS + FPGA System*. Hardware System Design, May 2021.