

report.md

HSD Lab 08

2014-17831 김재원

MyIP의 기능

- 보이지 않은 코드가 어떻게 짜여있을지 그 가능성이 너무 다양해서 정확히는 모르겠지만, main.c 코드를 수정해가며 도출해낼 수 있었던 결론들은 다음과 같다:
- 0x43C00000 의 위치에 0x5555 값이 들어왔을 경우
 - 0x40000000 위치에서 시작하는 BRAM의 0, 1, 2, 3번째 주소에 있는 값을 읽어 와서
 - 해당 값들을 각각 2배 한 값들을 4, 5, 6, 7번째 주소에 써준다.
 - 이 때, 0x43C00000 위치의 값은 0x5555 가 들어올 시에만 0x0 으로 변경을 해줄 것으로 예상이 된다. 이 과정이 언제 일어나는지 정확히는 알 수 없지만, 앞의 과정들이 모두 종료된 후에 일어날 가능성이 높다.
- 0x5555 값이 들어오지 않으면 별다른 동작 없이 종료할 것으로 예상된다.

MyIP 기능 추론 과정

Step 0

- main.c 를 수정하지 않고 실행했을 때의 결과는 다음과 같았다.

```

return 0;
"main.c" 38L, 903C 저장 했습니다
zed@debian-zynq:~/hsd21_lab08_practice$ make
gcc main.c && sudo ./a.out
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
zed@debian-zynq:~/hsd21_lab08_practice$
  
```

Step 1: fpga_ip 의 state

- fpga_ip 에 0x5555 가 들어간 후 어떤 일들이 일어나는지 확인하고자 다음과 같이 코드를 변경하였다.
 - fpga_ip[0] 에 0x5555 가 들어간 후 앞서 mmap() 을 했던 fpga_ip 의 영역에 어떤 값들이 저장되어 있는 상태가 되는지,
 - 그리고 while 문 안에서 fpga_ip 의 값이 어떻게 변경되는지 알아보고자 했다.

```

*(fpga_ip) = 0x5555;
// 추가
printf("%-10s%-10s\n", "addr", "IP(hex)");
for (i = 0; i < SIZE; i++)
    printf("%-10d%-10X\n", i, *(fpga_ip + i));

int j = 1;
while (*fpga_ip == 0x5555) {
    printf("===\n");
    printf("loop %d\n", j++);
    for(i = 0; i < SIZE; i++)
        printf("%-10d%-10X\n", i, *(fpga_ip + i));
}

```

- 결과 및 결론: while 문은 너무 빨리 돌아서 안의 동작을 살펴보기가 어렵고, fpga_ip의 첫번째 값은 0x5555를 대입해도 즉시 0x0 값이 출력된다.

addr	FPGA(hex)
0	0
1	2
2	4
3	6
4	0
5	0
6	0
7	0

addr	IP(hex)
0	0
1	0
2	0
3	0

addr	FPGA(hex)
0	0
1	2
2	4
3	6
4	0
5	4
6	8
7	C

Step 2: while 문의 역할

- (fpga_ip의 값들을 출력하는 코드를 삭제하고) while (*fpga_ip == 0x5555); 부분을 주석처리하였다.
- 결과 및 결론: while문이 없어도 동일하게 동작한다. 맥락상 MyIP의 동작이 완료된 후에 fpga_ip[0]의 값이 0이 되도록 구현되어 있어 해당 while문으로 MyIP의 동작이 완료된 후에 변경된 BRAM 속 값들을 출력하도록 한 것 같지만, MyIP의 동작이 충분히 빨라 이를 정확히 알기는 어려웠다.

```

gcc main.c && sudo ./a.out
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

```

Step 3: 0x5555 값의 역할

- 0x5555의 역할이 무엇인지 알기 어려웠다. BRAM에 입력된 값들이 0x5555 값과 어떤 식으로든 interact하여 결과가 도출되었을거라 생각하여 이 값을 다음과 같은 방식으로 변경해보았다.
 - 앞서 fpga_ip[0]에 저장된 0x5555가 0x0로 즉시 변경되어 출력되었기에 0x0 값부터 시작해서 다양한 값들을 이용해보았다.
 - 우선 hexadecimal to decimal/binary 변환기를 이용해보니 0x5555가 10101010..01의 값이었다.
 - 따라서 binary 형태가 유사하거나, 특이하거나, 5라는 값을 공유하는 0x5000, 0x0005, 0x0555, 0x1111, 0xFFFF, 0x5557 (LSB의 바로 왼쪽 값만 1로 변경된 값)를 모두 대입해보았다.

```
*(fpga_ip) = 0x0000; // 0x1000, ...
while(*fpga_ip == 0x0000); // 0x1000, ...
```

- 결과 및 결론: 무한루프가 생성되었다. 0x5555가 아닌 값들 중 자기 자신과 다른 값 (i.e., 0x0)으로 변경되는 값을 찾지는 못했다. 편의상 0x5555를 입력해야만 0x0로 변경이 된다고 가정하였다.

Step 4: 0x5555 이외의 입력

- 앞서 Step 3에서 무한루프가 생성되어 0x5555가 아닌 값이 입력되었을 때에도 MyIP가 BRAM에 대해 동일한 동작을 하는지 확인하고자 하였다.
 - 즉, 0x5555가 아닌 값이 입력될 경우 fpga_ip[0]의 값만 변경이 되지 않는 것인지, 혹은 BRAM의 값도 변경시켜주지 않는 것인지 살펴 보았다.

```
// run ip
*(fpga_ip) = 0x1000;
printf("===\n");
for(i = 0; i < SIZE; i++)
    printf("%-10d%-10X\n", i, *(fpga_ip+i));
printf("===\n");
for(i = 0; i < SIZE * 2; i++)
    printf("%-10d%-10X\n", i, *(fpga_bram+i));
printf("===\n");
while (*fpga_ip == 0x1000) {
    printf("IP:\n");
    for(i = 0; i < SIZE; i++)
        printf("%-10d%-10X\n", i, *(fpga_ip+i));
    printf("BRAM:\n");
    for(i = 0; i < SIZE * 2; i++)
        printf("%-10d%-10X\n", i, *(fpga_bram+i));
}
printf("===\n");
```

- 결과 및 결론: while문 안에 fpga_ip와 fpga_bram의 값들을 출력하도록 하였는데, 두 개 모두 initialize할 당시의 값과 동일한 값들이 유지되었다. 0x1000 값으로만 확인을 했지만, 편의상 0x5555가 아닌 값을 대입할 경우 BRAM의 작동도 하지 않는다고 일반화된 결론을 내렸다.

addr	FPGA(hex)
0	0
1	2
2	4
3	6
4	0
5	0
6	0
7	0
===	
0	1000
1	0
2	0
3	0
===	
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	0
6	0
7	0
===	
0	1000
1	0
2	0
3	0
===	
0	0
1	2
2	4
3	6
4	0
5	0
6	0
7	0
===	
IP:	
0	1000
1	0
2	0
3	0
BRAM:	
0	0
1	2
2	4
3	6
4	0
5	0
6	0
7	0
IP:	
0	1000
1	0
2	0
3	0
BRAM:	
0	0
1	2
2	4
3	6
4	0
5	0
6	0
7	0
IP:	
0	1000
1	0
2	0
3	0
BRAM:	
0	0
1	2
2	4
3	6
4	0
5	0
6	0
7	0
3	0
BRAM:	
0	0
1	2
2	4
3	6
4	0
5	0
6	0

```

7      0
IP:
0      1000
1      0
2      0
3      0
BRAM:
0      0
1      2
2      4
3      6
4      0
5      0
6      0
7      0
IP:
0      1000
1      0
2      0
3      0
BRAM:
0      0
1      2
2      4
3      6
4      0
5      0
6      0
7      0

...

(ctrl+c)

```

Step 5: BRAM에 저장되는 값

- 0x5555 라는 숫자의 의미에 대해 명확히 아는 것은 포기하고 BRAM에 overwrite되는 값들의 의미를 파악하고자 하였다. 언뜻 보서는 앞의 네 개 값을 두 배 곱한 값들이 뒤의 네 개 주소에 대입되는 듯이 보였기에, 이를 확인하고자 fpga_bram 를 다른 값들($*(fpga_bram + i) = (i * 3);$) / ($*(fpga_bram + i) = (i * 2) + 1;$)로 initialize 해보았다.
- 결과 및 결론: 다음은 fpga_bram 을 ($*(fpga_bram + i) = (i * 3);$)로 initialize 했을 때의 결과이다. addr 47에 저장되는 값은 addr 03에 있는 값들을 두 배 곱한 값들일 것이라는 앞선 가정이 대강 맞을 것이라고 결론지을 수 있었다.

```

addr    FPGA(hex)
0       0
1       3
2       6
3       9
4       0
5       0
6       0
7       0
addr    FPGA(hex)
0       0
1       3
2       6
3       9
4       0
5       6
6       C
7       12

```

Step 6: 다른 값으로 initialize한 경우

- 앞선 과정에서 addr 4~7의 값들이 변하는 것을 확인할 수 있었는데, 이것이 addr 에 저장된 값이 0이 아닐 때도 마찬가지로 작동하는지 확인하고자 다음과 같이 코드를 수정해보았다.

```

// initialize memory
for (i = 0; i < SIZE * 2; i++)

```

```
*(fpga_bram + i) = (i * 3);
// for (i = SIZE; i < SIZE * 2; i++)
// *(fpga_bram + i) = 0.0f;
```

- 결과 및 결론: addr 47번째 값이 바뀌는 것은 addr 03번째 값으로 부터 받아와서 두 배 한 후 overwrite해주는 형태로 변경이 되는 것이라고 대강 결론지을 수 있었다.

addr	FPGA(hex)
0	0
1	3
2	6
3	9
4	C
5	F
6	12
7	15

addr	FPGA(hex)
0	0
1	3
2	6
3	9
4	0
5	6
6	C
7	12

Step 7-1: SIZE 가 4가 아닌 경우

- addr 4~7번째 값이 변하는 것이 SIZE 값이 4로 선언되어 있어서 그런 것인지, 아니면 해당 주소들에 대해서만 값이 변경이 되도록 설정되어 있는 것인지 확인하고자 하였다.
 - 즉, SIZE 가 BRAM으로의 write을 하는 데에 있어 parameter 역할을 하는 것인지, 아니면 단순 constant인지 확인하고자 하였다.
 - 따라서, SIZE 를 6으로 변경해보았다.
- 결과 및 결론: SIZE 가 6이어도 4번째 값부터 변경이 되며, 이에 따라 자연스레 addr 의 8~ 번째 값들은 5~ 번째 값들의 두 배를 한 값이 저장되는 것이 아니라, 기존에 initialize를 한 값이 그대로 저장되어 있다.

addr	FPGA(hex)
0	0
1	3
2	6
3	9
4	C
5	F
6	12
7	15
8	18
9	1B
10	1E
11	21

addr	FPGA(hex)
0	0
1	3
2	6
3	9
4	0
5	6
6	C
7	12
8	18
9	1B
10	1E
11	21

Step 7-2: SIZE 가 4가 아닌 경우

- addr 가 4를 주기로 다르게 동작할 수도 있기에 13번째 이후의 값들을 확인할 수 있도록 SIZE 를 12로 변경해보았다.

- 결과 및 결론: addr 의 4~7번째 값을 제외한 나머지 값은 변경되지 않는다고 대강 가정할 수 있었다.

addr	FPGA(hex)
0	0
1	3
2	6
3	9
4	C
5	F
6	12
7	15
8	18
9	1B
10	1E
11	21
12	24
13	27
14	2A
15	2D
16	30
17	33
18	36
19	39
20	3C
21	3F
22	42
23	45
addr	FPGA(hex)
0	0
1	3
2	6
3	9
4	0
5	6
6	C
7	12
8	18
9	1B
10	1E
11	21
12	24
13	27
14	2A
15	2D
16	30
17	33
18	36
19	39
20	3C
21	3F
22	42
23	45

Step 8: *fpga_ip 가 변하는 시점

- 0x5555 가 입력이 되어야 BRAM에 값이 변경이 되는 것이 맞는지, 그리고 0x5555 가 0x0 으로 변하는 시점을 확인하고자 하였다.
 - 우선 0x1111 을 입력 후 fpga_ip 와 fpga_bram 의 값들을 확인하고, 3초가 지난 후에 0x5555 를 입력해주면 제대로 동작하는지 확인하고자 했다.
 - 또한, while(*fpga_bram == 5555); 로 기다리는 것이 아니라 while(*(fpga_bram + 4) == 0) 의 반복문 안에서 fpga_ip[0] 를 출력하도록 하여 fpga_ip[0] 에 저장된 값이 바뀌는 시점이 언제인지 파악하고자 하였다. 이를 위해 fpga_bram 의 03번째 값은 0이 아닌 값들로, 47번째 값은 0으로 initialize를 해주었다.

```
// initialize memory
for (i = 0; i < SIZE; i++)
    *(fpga_bram + i) = (i * 2) + 1; // 변경
for (i = SIZE; i < SIZE * 2; i++)
    *(fpga_bram + i) = 0.0f; // 변경

...
```

```

*(fpga_ip) = 0x1111;
printf("IP (IP=0x1111):\n");
for (i = 0; i < SIZE; i++)
    printf("%-10d%-10X\n", i, *(fpga_ip + i));
sleep(3);
printf("BRAM (IP=0x1111):\n");
for (i=0; i < SIZE*2 ; i++)
    printf("%-10d%-10X\n", i, *(fpga_bram+i));

*(fpga_ip) = 0x5555;
while(*(fpga_bram+4) == 0) {
    printf("IP[0]: %X\n", *fpga_ip);
}
printf("IP (IP=0x5555):\n");
for(i = 0; i < SIZE; i++)
    printf("%-10d%-10X\n", i, *(fpga_ip + i));
while (*fpga_ip == 0x5555);

...

```

- 결과 및 결론: 0x5555 가 입력되지 않으면 fpga_ip 와 fpga_bram 의 값은 변하지 않았고, MyIP의 동작이 너무 빨라 while 문 안의 출력이 되지는 않아서 0x5555 가 0x0 로 변하는 시점을 정확히 알기는 어려웠다. 하지만 해당 값이 변경된 후에 fpga_bram 의 값들이 출력되도록 구현되어 있는 것으로 보아, 맥락상 fpga_bram 의 값들이 먼저 변경되고 그 후 fpga_ip[0] 가 변경될 것을 짐작해볼 수 있다.

addr	FPGA(hex)
0	1
1	3
2	5
3	7
4	0
5	0
6	0
7	0
IP (IP=0x1111):	
0	1111
1	0
2	0
3	0
BRAM (IP=0x1111):	
0	1
1	3
2	5
3	7
4	0
5	0
6	0
7	0
IP (IP=0x5555):	
0	0
1	0
2	0
3	0
addr FPGA(hex)	
0	1
1	3
2	5
3	7
4	2
5	6
6	A
7	E

Step 9: 0x5555 가 아닌 입력에 대한 MyIP의 동작

- fpga_ip 에 0x5555 가 아닌 값이 입력되었을 때 MyIP가 0x5555 의 입력을 기다리는지, 아니면 별다른 동작을 하지 않는지 확인하고자 fpga_ip 에 0x5555 이 아닌 값을 입력하고 while 문을 삭제하여 다음과 같이 수정하였다.

```

*(fpga_ip) = 0x1111;
printf("IP (IP=0x1111):\n");

```



```
for (i = 0; i < SIZE; i++)
    printf("%-10d%-10X\n", i, *(fpga_ip + i));
```

- 결과 및 결론: while 문이 있을 때와 달리 fpga_bram 의 값들이 변경되지 않은 채로 즉각 프로그램이 실행 후 종료되어 MyIP가 0x5555 의 입력을 기다리는 것은 아니고, 0x5555 이 입력되었을 경우에만 특별히 다른 동작 (i.e., addr 의 47번째 주소에 03번째 값들을 2배 한 값들을 저장)을 한다는 것을 확인할 수 있었다.

addr	FPGA(hex)
0	1
1	3
2	5
3	7
4	0
5	0
6	0
7	0
IP (IP=0x1111):	
0	1111
1	0
2	0
3	0
addr	FPGA(hex)
0	1
1	3
2	5
3	7
4	0
5	0
6	0
7	0