

마이크로 프로세서

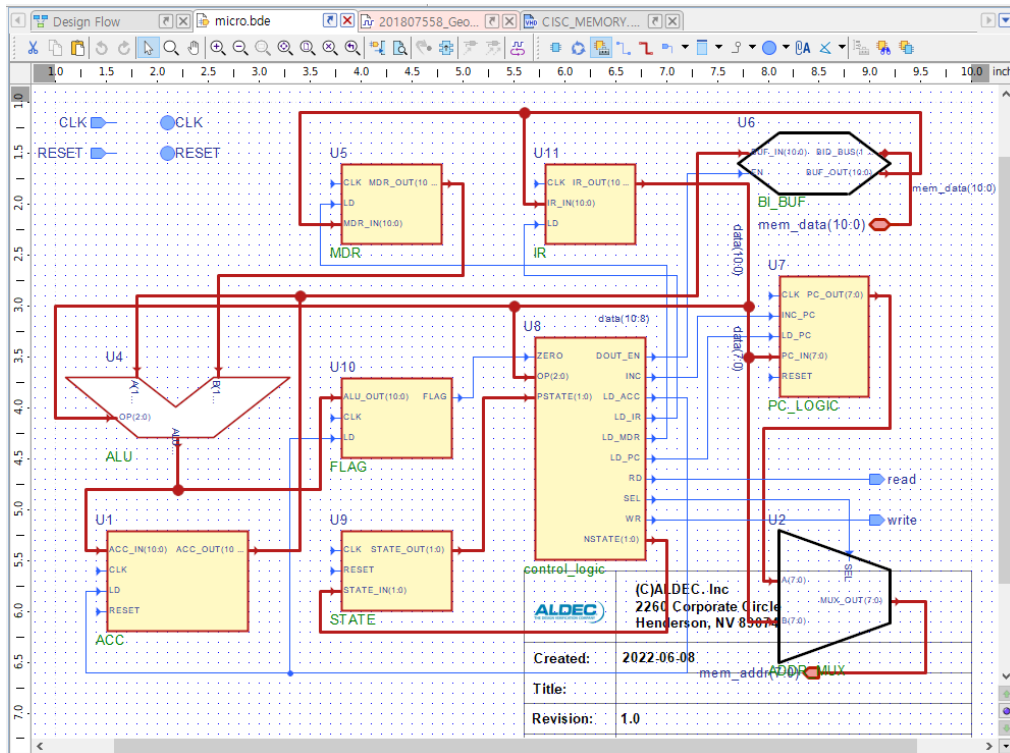
내가 설계한 **CPU** 프로그램 설명서

〈목차〉

1. 프로그램을 설계한 과정
2. CPU의 프로그램 설명
 - > 등비수열
 - > 설명시작
 - > 프로그램 응용 부록

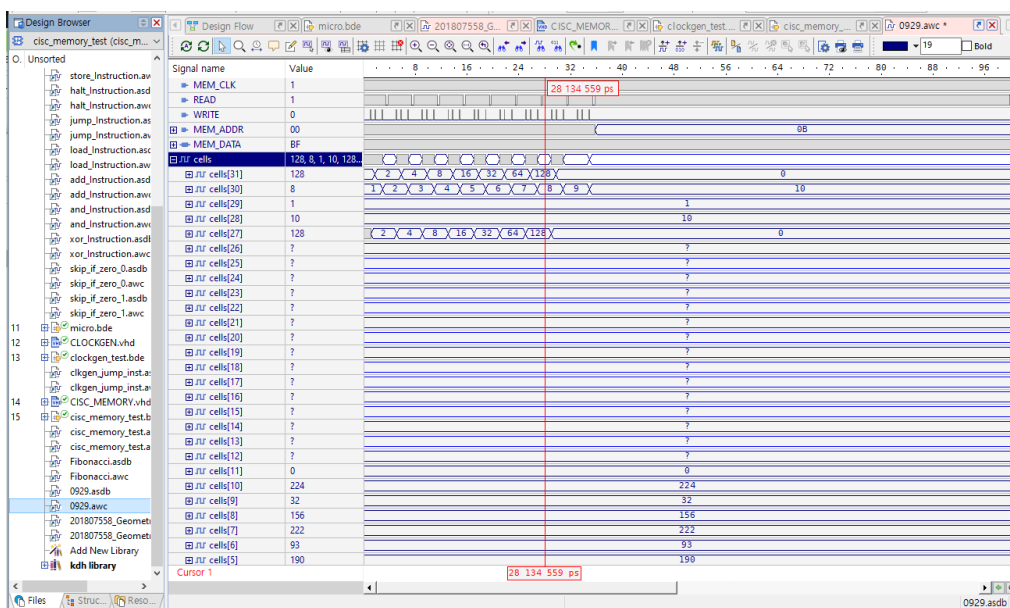
설계일자 : 2022.10.13
수업명 : 마이크로 프로세서
학번 : 201807558
이름 : 김도훈

1. 프로그램을 설계한 과정

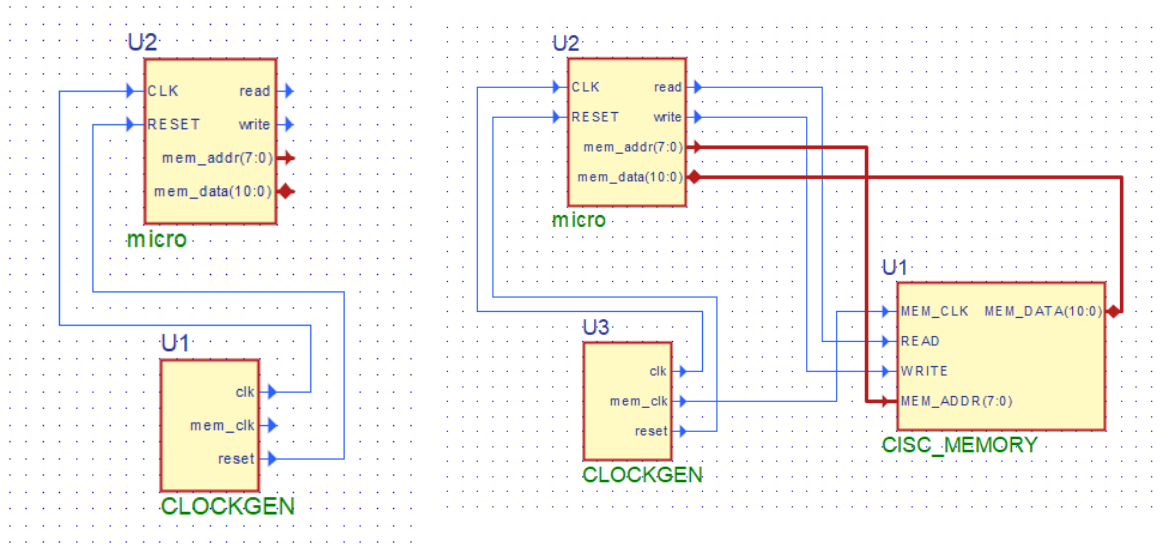


최종적으로 제가 설계한 CPU의 bde 입니다.
저는 이 CPU로 등비수열을 구현하고자 하였습니다.

그래서 제가 맨 처음 설계한 CPU는 8비트의 메모리였고
초기 구현한 등비수열 프로그램은 아래의 결과값을 보여주었습니다.



결과값이 십진수 128에서 끝나고 이를 이진수로 변환하면 10000000 이며 8비트의 메모리로 구현한 프로그램으로는 더 큰 다음 값을 얻을 수 없기 때문에 11비트까지 메모리 크기를 증가하기로 결정하였습니다.



위와 같이 CLOCKGEN 과 CISC_MEMORY 의 bde 를 전부 수정한 후 CISC_MEMORY.vhd 까지 아래와 같이 수정하고서 11비트 메모리의 번지수 255 down 0 에 맞게 번지를 추가해주었습니다.

```

25 library IEEE;
26 use IEEE.std_logic_1164.all;
27 use IEEE.std_logic_misc.all;
28 use IEEE.std_logic_arith.all;
29 --use IEEE.std_logic_components.all;
30
31 entity CISC_MEMORY is
32     port(
33         MEM_CLK : in STD_LOGIC;
34         READ : in STD_LOGIC;
35         WRITE : in STD_LOGIC;
36         MEM_ADDR : in STD_LOGIC_VECTOR(7 downto 0);
37         MEM_DATA : inout STD_LOGIC_VECTOR(10 downto 0)
38     );
39 end CISC_MEMORY;
40
41
42 architecture CISC_MEMORY of CISC_MEMORY is -- geometric sequence 등비수열
43
44     subtype MEM_WORD is std_logic_vector(10 downto 0);
45     type MEM_TABLE is array(255 downto 0) of MEM_WORD;
46     signal cells : MEM_TABLE := (
47         "000000000001", --11111111 번지: Xn = 1    255 start!
48         "000000000001", --11111110 번지: n = 1
49         "000000000001", --11111101 번지: one = 1
50         "000000010111", --11111100 번지: Last_n = 11    Result = 1024!
51         "XXXXXXXXXXXX", --11110111 번지: tmp
52         "XXXXXXXXXXXX", --
53         "XXXXXXXXXXXX", --
54         "XXXXXXXXXXXX", --
55         "XXXXXXXXXXXX", --
56         "XXXXXXXXXXXX", --
57         "XXXXXXXXXXXX", --
58         "XXXXXXXXXXXX", --
59         "XXXXXXXXXXXX", --
60         "XXXXXXXXXXXX", --
61         "XXXXXXXXXXXX", --
62         "XXXXXXXXXXXX", --
63         "XXXXXXXXXXXX", --
64         "XXXXXXXXXXXX", --
65         "XXXXXXXXXXXX", --
66         "XXXXXXXXXXXX", --
67         "XXXXXXXXXXXX", --
68         "XXXXXXXXXXXX", --
69         "XXXXXXXXXXXX", --
70         "XXXXXXXXXXXX", --
71         "XXXXXXXXXXXX", --

```

2. CPU의 프로그램 설명 - 등비수열

먼저 “등비수열”이란 3,6,12,24,48,...처럼 연속한 두 항의 비가 일정한 수열을 “등비수열”이라고 하며 여기에서 연속한 두 항의 비를 “공비”라고 합니다. 일반적으로 등비수열의 첫째 항을 a , 공비를 r 로 표기하는데 이에 따라 등비수열은 아래의 식과 같은 일반항을 가집니다.

$$a_n = ar^{n-1}$$

일반항을 쉽게 이해하기위한 등비수열의 예를 하나 보여드리겠습니다.

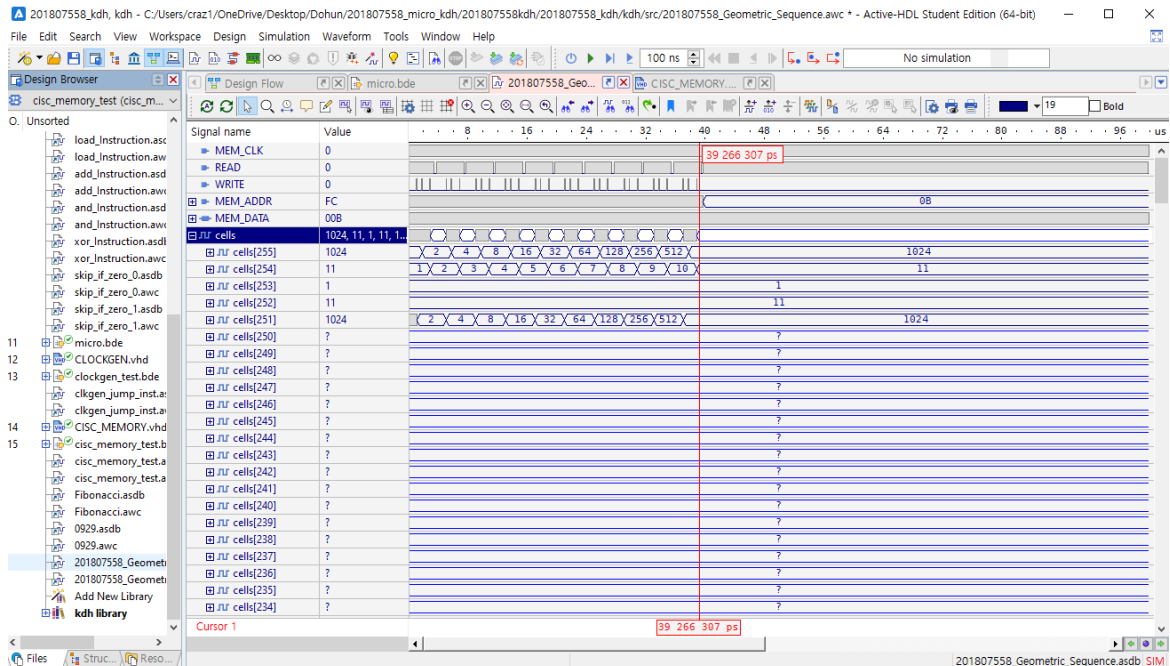
예)

수열 3,6,12,24,...의 일반항은
첫째항이 3, 공비가 $6 \div 3 = 2$ 이므로,
 $a_n = 3 \times 2^{n-1}$

등비수열이란 위와 같이 첫째항과 공비에 따라서 값이 달라지는 수열입니다.

그럼 이제 다음 장에서 본격적으로 프로그램을 설명하겠습니다.

2. CPU의 프로그램 설명 - 설명시작



위 사진이 최종적으로 만들어낸 프로그램 결과입니다.

11비트까지 증가시켜서 첫째항이 1일때 공비가 2인 등비수열이고
이 등비수열의 조건인 첫째항, 공비로 11비트 내에서 나타낼 수 있는 숫자인
1024 까지 계산값이 나오는 프로그램입니다.

```

275 "XXXXXXXXXX" --
276 "XXXXXXXXXX" --
277 "XXXXXXXXXX" --
278 "XXXXXXXXXX" --
279 "XXXXXXXXXX" --
280 "XXXXXXXXXX" --
281 "XXXXXXXXXX" --
282 "XXXXXXXXXX" --
283 "XXXXXXXXXX" --
284 "XXXXXXXXXX" --
285 "XXXXXXXXXX" --
286 "XXXXXXXXXX" --
287 "XXXXXXXXXX" --
288 "XXXXXXXXXX" --
289 "XXXXXXXXXX" --
290 "0000000000" --Halt
291 "1110000000" --Jump start
292 "0010000000" --Skip-if-zero
293 "1001111110" --Xor Last_n
294 "1101111110" --Store n
295 "0101111101" --Add one
296 "1011111110" --Load n
297 "1101111111" --Store Xn
298 "1101111101" --Load tmp
299 "1101111101" --Store tmp
300 "0101111111" --Add Xn
301 "1011111111" --Start: Load Xn
302 begin
303
304 process
305 variable address : integer range 0 to 255;
306 begin
307 wait until MEM_CLK'event and MEM_CLK='1';
308 address := conv_integer(unsigned(mem_addr));
309 if read='1' then
310 mem_data <= cells(address);
311 elsif write='1' then
312 cells(address) <= mem_data;
313 else
314 mem_data <= "ZZZZZZZZZZ";
315 end if;
316 end process;
317 end CISC_MEMORY;
318

```

프로그램의 vhd 코드 내용입니다.

변지수는 11비트의 메모리 이므로 앞의 op-code 3자리를 제외하면
8자리여서 2의 8승 값이어야 하기 때문에
그러므로 총 변지수는 0번지부터 255번지까지 총 256 입니다.

=====

왼쪽 사진부터 보면 제일 상위 “11111111” 번지를 Xn 이라 할 때
1값인 “0000000001” 를 넣었고 바로 다음 번지에 카운트로 사용할 n 값을
1부터 시작하기위해 똑같이 1로 넣어주고

그 다음 “11111101” 번지의 one 값은 카운트 n 을 1씩 증가 시켜주기 위해
똑같이 1을 넣어주었습니다.

그 다음 “11111100” 번지의 Last_n 값으로 카운트 n 이 그 값 까지만 증가하고
프로그램을 종료 할 수 있도록 값을 11(1011)로 지정하였습니다.

그리고 “11111011” 번지의 tmp 는 “XXXXXXXXXXXX” 으로 두는데
그 이유는 맨 아래의 MachineLanguage 가 연산되면서
그 값들을 tmp에 저장하기 위해서 입니다.

=====

이제 오른쪽 사진으로 넘어가서 제가 설계한 MachineLanguage 를 해석하면

[** Load(101), Store(110), Add(010), Xor(100)]

[** Skip-if-zero(001), Halt(000), Jump(111)]

1. Load Xn 으로 “11111111” 번지의 1값을 불러옵니다.
 2. Add Xn 으로 내가 맨위 “11111111” 번지에서 정한 값을 똑같이 더해주고
(위 사진의 프로그램에서는 그 값은 “1” 입니다.)
 3. Store tmp 로 그 연산한 값 (“2”) 을 tmp에 저장해줍니다.
 4. Load tmp 로 다시 값을 불러옵니다.
 5. Store Xn 으로 그 값(“2”)으로 Xn 에도 저장하여 바꿔줍니다.
 6. Load n 으로 카운트로 사용 될 n 값을 불러옵니다.
 7. Add one 으로 카운트 n 을 1 증가시킵니다.
 8. Store n 으로 증가된 카운트 n 값을 저장합니다.
 9. Xor Last_n 으로 위에서 지정한 Last_n => 11(“1011”) 값과 증가된 n 값을
Xor 연산을합니다.
 10. Skip-if-zero 는 9번에서 Xor 값 zero flag의 값이 ‘1’ 일 경우 PC의 값을
1 만큼 증가시켜서 다음 명령을 건너뛰게합니다.
이 프로그램이 계속 실행되다가 n값이 Last_n과 같아진 상태로
9번에서 Xor 연산을 하게되면 10번의 Skip-if-zero 가 다음 명령을
건너 뛰게하여 Halt 명령을 만나 프로그램이 종료되도록 합니다.
 11. Jump Start 는 카운트를 1부터 증가시키면서 Xor 연산을하고
10번 Skip-if-zero 가 실행되지 않았을때 다시 Load Xn의 번지 값인
“00000000” 으로 점프시켜줘서 프로그램을 지정한 값까지
계속 돌아가게 하는 명령어입니다.
 12. Halt 는 10번에서 설명한 내용과 같이 프로그램을 종료시킵니다.
- =====

2. CPU의 프로그램 설명 - 프로그램 응용 부록

프로그램은 바로 위의 내용이었던 목차 “설명시작” 두번째 장의 내용대로 동작합니다.

그리고 이 등비수열 프로그램은 위 결과가 끝이 아니고 첫째항과 등비를 수정할수 있는데 프로그램 사용자가 원하는대로 값을 조금만 바꾸면 첫째항과 등비가 양수인 조건 하에 등비수열을 계산하는 프로그램을 입맛에 따라 바꿀 수 있습니다!

제가 더 만든 예시를 보여드리면서 프로그램 설명서 마무리 하겠습니다.

첫째항=3, 공비=5 일 때

```

41 architecture CISC_MEMORY of CISC_MEMORY is -- geometric sequence 동비수율
42
43 subtype MEM_WORD is std_logic_vector(10 downto 0);
44 type MEM_TABLE is array(255 downto 0) of MEM_WORD;
45 signal cells : MEM_TABLE := (
46   "00000000011", --11111111 번지 : Xn = 3      start!
47   "00000000001", --11111110 번지 : n = 1
48   "00000000001", --11111101 번지 : one = 1
49   "00000000101", --11111100 번지 : Last_n = 5  Result =|
50   "XXXXXXXXXXX", --11111011 번지 : tmp
51   "XXXXXXXXXXX", --
52   "XXXXXXXXXXX", --
285  "XXXXXXXXXXX", --
286  "XXXXXXXXXXX", --
287  "00000000000", --Halt
288  "11100000000", --Jump start
289  "00100000000", --Skip-if-zero
290  "10011111100", --Xor Last_n
291  "11011111110", --Store n
292  "01011111101", --Add one
293  "10111111110", --Load n
294  "11011111111", --Store Xn
295  "10111111011", --Load tmp
296  "11011111011", --Store tmp
297  "01011111111", --Add Xn      Add 4회 공비 r = 5
298  "01011111111", --Add Xn      Add 3회 공비 r = 4 (*공비값소 시킬 때는 Add 삭제 후 비어있는 데이터값 추가)
299  "01011111111", --Add Xn      Add 2회 공비 r = 3 (*공비증가 시킬 때는 Add 추가 후 비어있는 번지 데이터값 지우기)
300  "01011111111", --Add Xn      Add 1회 공비 r = 2 (default)
301  "01011111111"); --start: Load Xn
302 begin

```

nr cells	1875, 5, 1, 5, 187...		
nr cells[255]	1875		1875
nr cells[254]	5		5
nr cells[253]	1		1
nr cells[252]	5		5
nr cells[251]	1875		1875
nr cells[250]	?		?
nr cells[249]	?		?
nr cells[248]	?		?

첫째할=7, 공비=3 일 때

```

41 architecture CISC_MEMORY of CISC_MEMORY is -- geometric sequence 등비수열
42
43 subtype MEM_WORD is std_logic_vector(10 downto 0);
44 type MEM_TABLE is array(255 downto 0) of MEM_WORD;
45 signal cells : MEM_TABLE := (
46   "00000000111", --11111111 변지 : Xn = 7 start!
47   "00000000001", --11111110 변지 : n = 1
48   "00000000001", --11111101 변지 : one = 1
49   "00000001001", --11111100 변지 : Last_n = 9 Result =
50   "XXXXXXXXXXXX", --11111011 변지 : tmp
51   "XXXXXXXXXXXX", --
287  "00000000000", --Halt
288  "11100000000", --Jump start
289  "00100000000", --Skip-if-zero
290  "10011111100", --Xor Last_n
291  "11011111110", --Store n
292  "01011111101", --Add one
293  "10111111110", --Load n
294  "11011111111", --Store Xn
295  "10111111011", --Load tmp
296  "11011111011", --Store tmp
297  "01011111111", --Add Xn Add 2회 공비 r = 3 (공비중가 시킬 때= Add 추가 후 비어있는 변지 데이터값 자유기)
298  "01011111111", --Add Xn Add 1회 공비 r = 2 (default)
299  "10111111111"); --start: Load Xn
300 begin

```

nr cells	1701, 6, 1, 6, 1701	
nr cells[255]	1701	? x 21 x 63 x 189 x 567 x 1701
nr cells[254]	6	1 x 2 x 3 x 4 x 5 x 6
nr cells[253]	1	
nr cells[252]	6	
nr cells[251]	1701	? x 21 x 63 x 189 x 567 x 1701
nr cells[250]	?	
nr cells[249]	?	