Syn 도움말

Introduction about Syn

2025. 07. 01 최초 작성

서준혁

1. 소개

Syn(Synthetic Language for Logic Simulation)은 python 기반으로 제작된 프로그래밍 언어로 가상의 비트끼리의 연산 및 논리 판단으로 구동되는 언어이다. 기본적으로 10만개의 가상의 비트가 주어지고 이 비트에 각각 0 또는 1의 값을 할당하여 여러 가지 기능들을 수행한다. AND, OR, NOR, XOR등 여러 논리 게이트를 각 비트들과 연결하여 가상의 회로를 구상하고 이를 바탕으로 알고리즘이 구동되도록 한다. 가장 단순하면서 원초적인 방법으로 코딩을 할 수 있는 방법을 모색하던 중 이와 같은 언어가 탄생하게 되었다.

2. 연혁

이 언어는 2025. 06. 30에 개발 시작하여 2025. 07. 01에 공식적으로 Syn 1.0.0으로 공개되었다.

3. 환경

인터프린터 언어인 만큼 .syn파일과 python 구동기를 통해 구동이 가능하다. 파일 확장자는 앞에서 언급하였듯 .syn이며 예로 script.syn와 같이 쓸 수 있다. 이를 syn.py를 통해 실행한다. vscode에선 .vscode/를 통해 tasks.json과 lauch.json 파일을 생성하고 이를 바탕으로 syn.py에서 실행하지 않고 자체적으로 실행되도록 할 수 있다. 다음과 같이 파일을 생성한다.

```
script.syn
syn.py
.vscode ┌ tasks.json
        └ launch.json
```

4. 문법

기본적으로 Syn은 세미클론(;)로 줄을 종료한다.
또한 문자열은 $ $로 묶어 나타낸다.
앞으로 있어 코드에 대한 출력 결과를 *Italic*으로 쓰겠다.

1) 비트 선언

이 언어의 기본 단위인 비트는 그 값으로 0 또는 1만을 갖는다. 모든 비트의 초기값은 0이다. 임의의 비트 a가 있다고 할 때 다음과 같이 선언한다.

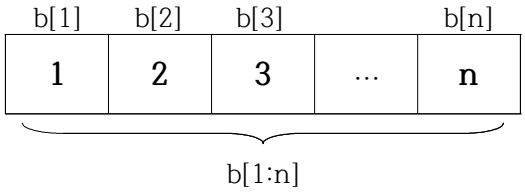
```
rel bit : a = 0;
```

이 코드를 통해 값이 0인 비트 a가 생성되었다.

비트의 묶음을 바이트라고 한다. 바이트는 다음과 같이 선언한다.

```
rel byt b[1:n];
```

이 코드로 1부터 n번 비트를 묶음으로 하는 b라는 이름의 바이트가 선언되었다. 각각의 비트는 b[k]로 정의한다. 단, k는 $1 \leq k \leq n$인 정수이다. 이를 다음 그림과 같이 나타낼 수 있다.



2) 출력

python의 print()와 같이 syn에서도 출력을 할 수 있다. 다음과 같이 출력한다.

```
out {출력할 내용};
```

예를 들어 다음과 같은 코드를 실행하면

```
out $Hello World!$;
```

Hello World!

와 같이 출력된다.

Syn에서 비트의 값을 의미하는 기호는 &이다. 만일 비트 a의 값이 1이라면 &a == 1이다. 따라서 다음과 같은 코드를 쓸 수 있다.

```
rel bit : a = 1;
out &a;
```

1

3) 연결과 게이트

Syn 언어만의 특별한 점이 두드러진다. 비트와 비트끼리를 게이트를 통해 연결하여 다른 비트와 연결시키는 함수이다. 다음과 같이 쓴다.

```
con {bit3} ~ gate:{게이트 종류}( {bit1} ,  {bit2} );
```

{게이트 종류} 게이트를 통해 연결한 두 비트 bit1, bit2를 bit3에 할당한다는 의미이다.

다음과 같은 예시를 들자.

```
rel bit : a = 1;
rel bit : b = 1;
rel bit : c = 0;

con c ~ gate:AND(a, b);

out &c;
```

1

이 출력된다. AND 게이트를 통해 1과 1이 연결되었으므로 그 결과는 1이다.
게이트의 종류는 다음과 같다.

gate의 종류	
AND	모든 값이 1일 때만 1이다.
OR	하나라도 1이면 1이다.
NOT	입력이 0이면 1, 입력이 1이면 0이다. (1개의 입력만 허용)
XOR	입력값이 서로 다르면 1이다.
NAND	AND의 반대이다.
NOR	OR의 반대이다.

4) 표준비트

이 언어에서는 앞에서 말했듯 일일이 비트를 선언할 필요가 없다. 기본적으로 10만개의 비트가 주어진다. 총 10개의 바이트(0x ~ 9x)가 주어지고 각각의 바이트마다 10000개의 비트(0000~9999)가 있다. 따라서 예시로 3x0213이라면 3번 바이트 213번 비트이다. 0x1230이면 0번 바이트 1230번 비트이다. 이를 **표준비트**라고 한다.
여기서 **고정비트**의 개념이 등장하는데 만일 0x0001C라고 하면 0번 바이트 1번 비트의 값은 코드가 실행되는 동안 영구적으로 고정된다는 의미이다.

5) Syn 명령어

rel	비트/바이트 선언
set	비트값을 강제 지정
pop	비트값을 강제로 0으로 설정
mov	mov A : B; 라고 하면 A를 B로 옮기고 그 값을 0으로 할당한다.
swc	swc A : B; 라고 하면 A와 B의 값을 서로 바꾼다.
rev	rev A; 라고 하면 A의 값을 반전시킨다.
con	연결한다.
cln	조건문이다.
goto	goto N; 이라고 하면 N번째 줄로 돌아가 다시 진행하는 것이다.
bdl	번들이다.
call	call이다.
ret	호출한 위치로 복귀한다.
out	출력 함수이다.
sft	쉬프트이다.
slp	시간제어이다.

6) 조건문

조건문은 기본적으로 다음과 같이 쓴다.

```
cln {조건}:
;    (코드1)
;       ⋮
```

7) 반복문

Syn에는 while이나 for처럼 반복문이 따로 없다. 하지만 goto를 이용해서 loop를 만들 수 있다.

```
1  out $ye ye$;
2
3  goto 1;
```

ye ye

ye ye

⋮

로 무한반복된다.

8) 시간제어

시간은 slp으로 지연이 가능하다.

```
slp {시간};
```

이라 하면 {시간}초 만큼 대기한 후 다음줄이 출력된다.

```
out $Hello$;
slp 5;
out $World!$;
```

Hello

(5초 대기)

World!

로 지연되며 차례대로 출력된다.

9) 번들과 call

번들(Bundle)은 꾸러미를 의미하며 마치 꾸러미속의 코드를 불러와 실행한다는 의미에서 붙여진 이름이다. bdl {이름};으로 시작하여 ret;으로 끝난다. 불러올 때는 call {이름};으로 한다.

```
bdl HELLO;
out $Hello$;
ret;

call HELLO;
```

Hello

10) 쉬프트

쉬프트란, 바이트에서 비트의 값들을 순차적으로 얼마나 밀 것인지를 결정한다.
만일 a[0]~a[4]인 바이트 a[0:4]가 있다고 할 때 그 값이 다음과 같다.

1	0	1	1	0

이때 오른쪽으로 쉬프트(index가 커지는 방향)하는 것을 양수로, 그 반대를 음수로 하면 다음과 같은 코드를 실행하자.

```
sft a[0:4] : 1;
```

실행 결과는 우측과 같다.

0	1	0	1	1

이때 밀고 남은 빈 비트는 0이 되고, 끝에 도달한 비트의 값은 소멸된다.

```
sft a[0:4] : -2;
```

의 결과는 다음과 같다.

1	1	0	0	0

5. 주석

'\'다음에 쓰는 글자는 주석처리 된다.

6. 사용자들에게

이 언어는 최대한 단순한 원리를 갖지만 강력한 힘을 갖고 있다. 마음만 먹는다면 기초적인 ALU까지 가상으로 만들 수 있다. 이를 통해 하드웨어 프로그래밍의 능력을 기르고 그 사고를 길러보자.

Syn Lang v1.0.0. released in 2025

developed by 서준혁
release date 2025. 07. 01