DataLab. Internship Program (2023 Summer)

Week 1

Tutor: 배은태

학습 목표

5주 간의 연구실 인턴 교육 프로그램을 통해 FPGA(Field Programmable Gate Array) 기반의 하드웨어 설계의 기초부터 Bluespec System Verilog를 이용한 해시 알고리즘 가속기의 구현과 RISC-V Custom Instruction를 이용한 제어까지 학습하고 실습할 예정입니다.

하드웨어를 설계하기 위해서 직접 회로도(schematic)를 그리는 대신 하드웨어의 동작을 기술하는 일종의 프로그래밍 언어인 하드웨어 기술 언어(Hardware Description Language, HDL)를 이용할 수도 있습니다. 대표적인 HDL에는 Verilog HDL과 VHDL, System Verilog 등이 있으며, 이보다 고수준의 문법을 제공하는 Bluespec System Verilog(BSV)나 Chisel 등의 High-level HDL(HLHDL)도 존재합니다. 또한 고급 언어인 C/C++를 이용하여 하드웨어 커널을 기술하는 High-level Synthesis(HLS)도 널리 사용되고 있습니다.

HDL을 이용한 하드웨어 회로 설계는 프로그래밍 언어를 사용한다는 점에서 소프트웨어 개발과 닮은 점도 있지만 논리 합성(synthesis)이나 구현(implementation) 등 생소한 하드웨어 용어부터 순차적으로 실행되는 소프트웨어 코드와는 사뭇 다른 하드웨어 회로의 동작까지 많은 부분이 다소 낯설게 다가올 수 있습니다. 따라서, 1-2주차에는 기본적인 Verilog HDL 코딩과 시뮬레이션, Xilinx FPGA 사용법에 대해 익히고, 이후에 BSV를 이용하여 좀 더 복잡한 회로를 구성하고자 합니다.

1주차에는 논리회로에 대한 기초적인 지식과 Verilog HDL의 기본 문법을 익히고 시뮬레이션을 통해 회로의 동작을 검증해보는 것을 목표로 합니다.

실습 환경 준비

소프트웨어와 마찬가지로 하드웨어 역시 의도에 맞게 설계되었는지 동작을 검증하는 단계가 필요합니다. 회로를 직접 합성하기 앞서(단순한 회로도 합성하고 구현해서 보드에 올리려면 상당한 시간이 필요합니다), 시뮬레이터를 이용하면 회로의 입출력 신호를 모니터링할 수 있습니다. 시뮬레이션을 하려면 회로가 원하는 값을 출력하는지 확인할 수 있도록 입력될 값을 직접 정의해줘야 합니다. 이를 테스트벤치(testbench) 또는 테스트 벡터라고 합니다. 또한 Verilog HDL은 시뮬레이션을 위한 각종 시스템 태스크($display)와 시뮬레이션에서만 사용 가능한 문법들(initial 등)을 제공하고 있습니다. 이들 문법들은 논리 합성, 즉 하드웨어 회로로 구현은 불가능하지만, 회로의 기능(functionality)을 검증하기 위한 유용한 도구로 사용할 수 있습니다.

Vivado는 Xilinx(AMD에 인수)에서 제공하는 EDA tool입니다. HDL 코드에 대한 시뮬레이션은 물론이고, Xilinx FPGA를 대상으로 논리 합성(synthesis), 구현(implementation), 비트스트림 생성

(bitstream generation) 등 다양한 기능을 제공합니다. Vivado는 무료 버전인 Standard Edition과 유료 버전인 Enterprise Edition이 있으며, 더 많은 제품군(FPGA)을 지원하느냐의 차이입니다. 실습에서 사용되는 교육용 보드는 Standard Edition으로도 충분합니다. EDA 툴은 하드웨어에 맞게 선택하시면 됩니다. 예를 들어, Xilinx FPGA 대신 Altera(Intel에 인수) FPGA를 사용할 경우에는 Vivado 대신에 Quartus를 선택하면 됩니다. (**Vivado는 용량이 매우 크고, 설치 시간도 크고 아름답기 때문에 미리 설치해두시기를 권장합니다.**)

https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools.html

물론 Vivado에도 시뮬레이터(xsim)가 존재하지만, 가볍고 편리하게 이용할 수 있는 유용한 오픈소스 툴이 있어서 소개하고자 합니다. 이들 툴은 설치 방법도 크게 복잡하지 않고, CLI 환경에서 쉽게 실행할 수 있습니다.

우선 Icarus Verilog입니다. Icarus Verilog(iverilog)는 입력된 소스코드(HDL 코드)를 실행 가능한 시뮬레이션 파일로 컴파일합니다. 이렇게 생성된 실행 파일은 vvp라는 런타임 엔진을 이용하여 실행시킬 수 있습니다. 시뮬레이션 파일을 실행하면 테스트벤치의 입력에 맞게 시뮬레이션을 진행하며, 그 결과를 터미널을 통해 확인할 수 있습니다. 또한 시뮬레이션 과정에 생성된 입력과 출력 신호를 파일 형태로(.vcd 파일) 덤프하여 파형을 확인할 수도 있습니다. 시뮬레이션 결과로 출력된 vcd 파일을 시각화 해주는 오픈소스 툴도 존재하며, gtkwave가 잘 알려져 있습니다.

https://steveicarus.github.io/iverilog/usage/installation.html

Icarus Verilog는 소스코드를 직접 컴파일하여 설치할 수도 있지만, 그냥 패키지 관리자로 설치할 수도 있습니다.

```
$ sudo apt-get install build-essential libboost-dev iverilog
$ sudo apt-get install gtkwave
```

iverilog의 사용법은 gcc 같은 CLI 환경에서 동작하는 컴파일러와 매우 비슷하기 때문에 gcc 사용에 익숙한 분들이라면 어렵지 않게 사용할 수 있을 것입니다. 예를 들어, ha.v라는 소스 파일과 이에 대한 테스트벤치인 tb_ha.v를 이용하여 시뮬레이션 파일을 생성하려면 다음과 같이 입력하면 됩니다(Verilog HDL의 소스 파일 확장자는 .v입니다).

```
$ iverilog tb_ha.v ha.v
$ vvp a.out
```

출력 파일을 지정하지 않으면 기본적으로 a.out이라는 파일이 생성되지만, (gcc와 마찬가지로) -o옵션을 이용하여 출력 파일명을 지정할 수도 있습니다.

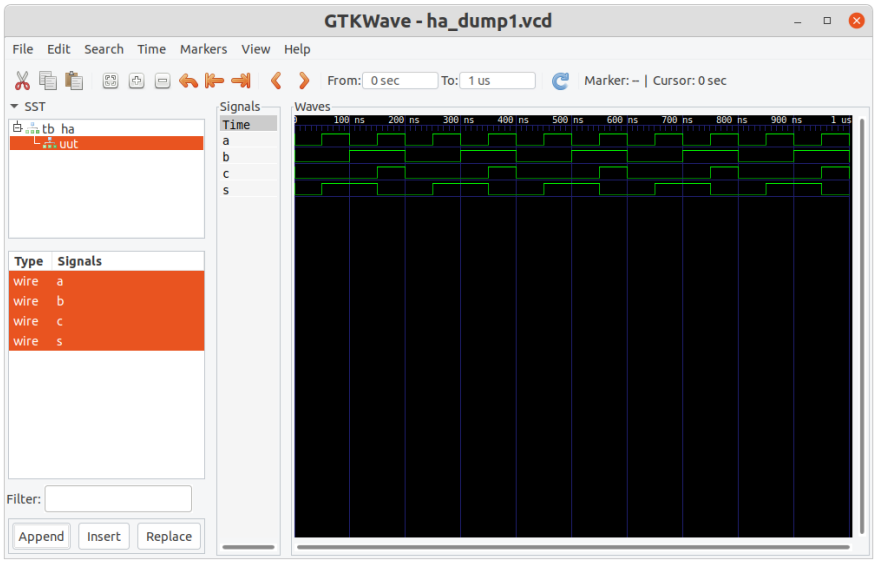
```
$ iverilog tb_ha.v ha.v -o ha
```

시뮬레이터를 실행하면 파형을 ha_dump1.vcd라는 파일로 덤프한다고 가정하면, 다음과 같이 gtkwave를 이용하여 확인할 수 있습니다.

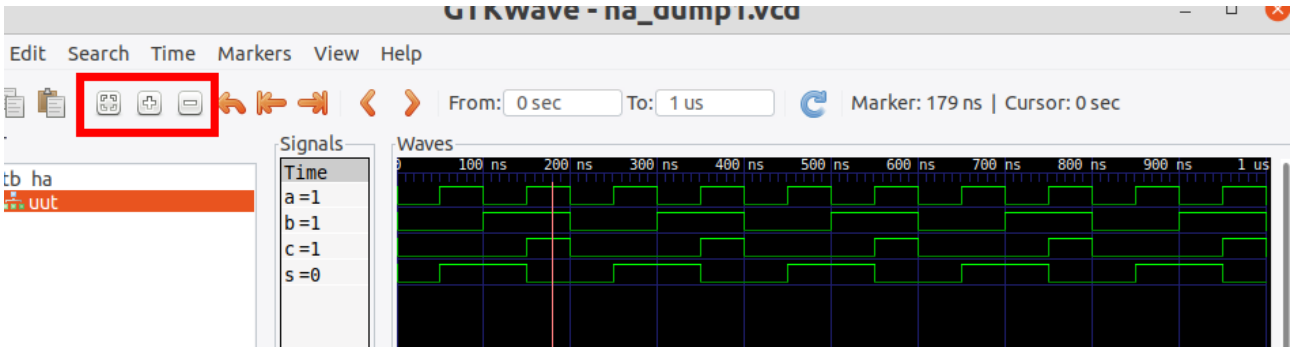
```
$ gtkwave ha_dump1.vcd &
```

실제 실행 예시:

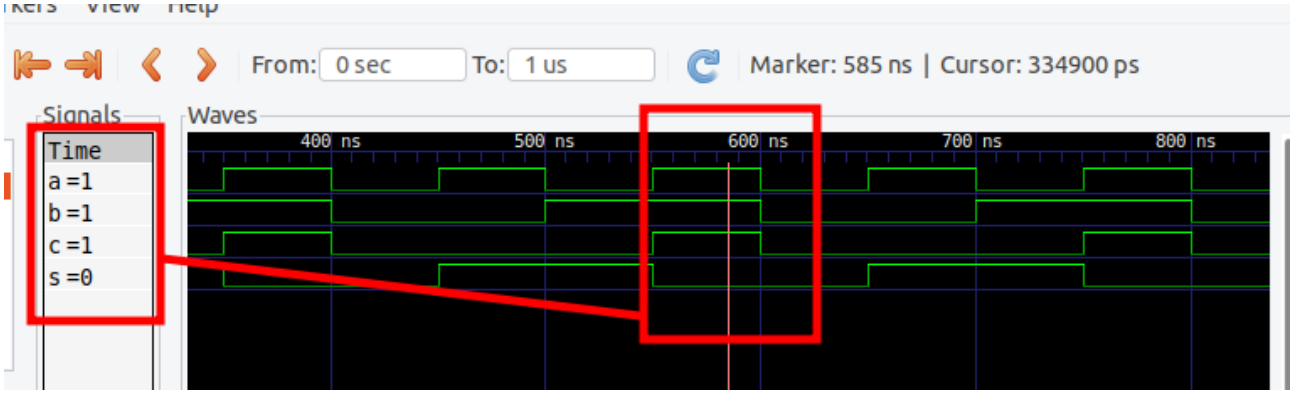
다음 그림은 입출력 신호 a, b, c, s를 선택하여 시각화 하고자 하는 신호 리스트에 append한 모습입니다.



파형의 모습은 좌측 상단의 버튼(빨간색 상자로 강조)을 이용하여 확대/축소할 수 있습니다. + 왼쪽에 있는 버튼은 현재 창 크기에 맞게 파형 폭을 조정합니다.



그리고 파형에서 특정 타이밍을 클릭하여 해당 타이밍의 신호 값을 확인할 수 있습니다. 다음 그림은 입력 a와 b가 모두 1일 때의 신호를 나타내고 있습니다. 반가산기(half adder)의 출력 c(carry)와 s(sum)은 각각 1과 0을 출력하고 있는 모습을 확인할 수 있습니다.



연습문제

1. C는 1비트 입력 A와 B의 논리 연산에 대한 출력을 나타냅니다. 진리표와 이에 대응하는 Verilog 코드(assign문)를 완성하세요. (논리 연산기호 •는 AND, +는 OR, ⊕는 XOR을 나타냅니다. 그리고 NOT은 A'나 $\bar{A}$ 등으로 나타냅니다.)

C=A • B

A	B	C
0	0	
1	0	
0	1	
1	1	

assign c =

C=A+B

A	B	C

0	0	
1	0	
0	1	
1	1	

assign c =

C=A⊕B

A	B	C
0	0	
1	0	
0	1	
1	1	

assign c=

C=A'

A	C
0	
1	

assign c=

2. 조합회로(combinational circuit)와 순차회로(sequential circuit)의 차이점을 간단히 서술하세요.

3. Behavioral modeling과 data-flow modeling, structural modeling에 대해 간단히 서술하세요(설명과 함께 교재나 웹에서 예시 코드를 가져와도 좋습니다).

4. wire와 reg의 차이점에 대해 간단히 서술하세요.

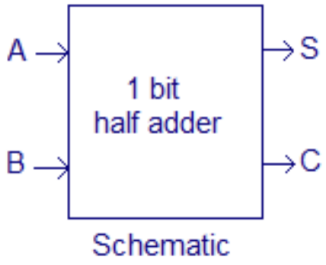
5. initial과 always의 차이점에 대해 간단히 서술하세요.

6. 실습: 다음 그림은 반가산기(half adder)의 진리표와 회로도입니다. 그림을 참고하여 반가산기의 design ha.v와 그 테스트벤치인 tb_ha.v를 작성하고, iverilog로 시뮬레이션 해봅시다. 그리고

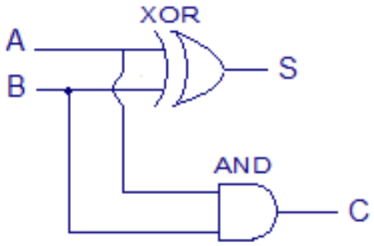
gtkwave를 통해 파형을 직접 확인해봅시다.

Inputs		Outputs	
A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Truth table



Schematic



Realization

참고: 파형을 파일로 덤프(dump)하려면 코드에 해당 구문을 추가하면 됩니다.

```
initial begin
  $dumpfile("ha_dump1.vcd");
  $dumpvars(0, tb_ha);
end
```

시스템 태스크 $dumpfile은 지정된 이름(ha_dump1.vcd)으로 파형 파일을 출력(dump)하라는 뜻이고, $dumpvars를 통해 모니터링하고자 하는 모듈을 지정합니다. 테스트벤치 모듈 이름이 tb_ha라면 tb_ha를 전달하면 됩니다.

7. 심화: 반가산기 모듈 2개를 이용하여 전가산기를 설계해봅시다. 6번 문제와 마찬가지로 gtkwave를 통해 입출력 파형을 확인해봅시다. (Hint: structural modeling)