TB_RISCV_* file 설명

TB_RISCV_inst는 RISC-V의 I type과 R type의 integer computational instruction을 test해 보는 testbench입니다.

TB_RISCV_forloop과 TB_RISCV_sort는 각각 for loop과 sort를 하는 testbench입니다.

TB_RISCV_inst의 경우 하나의 instruction씩 수행시킬 때마다 각 instruction을 올바르게 수행했는지 확인을 하고, 가장 기본적인 integer computational instruction을 다루고 있게 때문에 가장 먼저 TB_RISCV_inst로 시작을 하기를 권장합니다.

Testbench에서 clock과 reset을 만들어 내는 riscv_clkrst1, core의 역할을 하는 riscv_top1, instruction memory인 i_mem1, data memory인 d_mem1, register file인 reg_file1을 생성하여 각 module을 연결합니다.

이제 testbench의 전체 구성을 순서대로 설명하겠습니다.

Clock generator 생성

riscv_clkrst1에서 만들어 지는 RSTn signal로 모든 module이 초기화되도록 해야 하며, core, memory, 그리고 register file은 모두 CLK에 의해서 동기화 되도록 해야 합니다.

riscv_top1은 여러분이 설계해야 하는 core module입니다.

CLK 신호에 의해 동기화되도록 CLK 신호를 입력으로 받습니다.

Core 생성

core에서 memory를 접근하기 위해서는 I_MEM_CSN, I_MEM_DI, I_MEM_ADDR, D_MEM_CSN, D_MEM_DI, D_MEM_DOUT, D_MEM_ADDR, D_MEM_WEN, D_MEM_BE 신호를 만들어서 memory에 addressing하고 data를 받도록 구성되어 있습니다. I_MEM_CSN과 D_MEM_CSN은 RSTn 신호가 1이면 0을, 0이면 1의 값을 갖도록 core에서 wire를 assign하여야 memory를 정상동작 시킬 수 있습니다. I_MEM_ADDR와 D_MEM_ADDR는 각각 instruction memory와 data memory를 access할 때 사용하는 address로 특정 address를 접근하기 위해서 core내부에서 만들어 내야하는 값입니다. I_MEM_DI와 D_MEM_DI는 각각 instruction memory의 output과 data memory의 output 값으로 각각의 memory에서 읽은 값을 가지고 있습니다. D_MEM_DOUT은 data memory에 저장할 값을 담고 있고, D_MEM_WEN은 write enable negative으로 data를 memory에 써야 하는 instruction을 실행할 때에는 WEN이 0이 되도록 core에서 값을 만들어 내야 합니다. D_MEM_BE는 byte enable 값으로 data를 읽거나, 쓸 때 byte 단위를 조정하기 위해 사용합니다. (SB, SH, SW, LB, LH, LW, LBU, LHU instruction 실행 시 각각의 instruction의 semantic에 맞도록 조정. ex) SB → b0001 SH → b0011 SW → b1111 LB → b0001 LH → b0011 LW → b1111) memory module과 연결되어 있는 각

각의 wire들은 testbench file에서 모두 연결이 되어 있기 때문에 core 구현 시 각각의 output에 memory를 접근하기 위해 필요한 signal 값을 잘 내보내도록 설계하면 memory를 동작 시킬 수 있도록 구성되어 있습니다.

RF_WE는 register file write enable로 register에 값을 저장해야 될 때 RF_WE 값이 1이 되도록 core에서 생성해야 됩니다. RF_RA1, RF_RA2, RF_WA는 각각 source register 1, source register 2, destination register의 number를 만들어 내야합니다. RF_WD는 register file에 적을 data이고, RF_RD1, RF_RD2는 각각 source register로부터 읽을 값을 줍니다. Testbench file에서 register file과 core의 wire가 연결되어 있기 때문에 register number와 원하는 operation을 필요에 따라서 생성하면 register에 값을 읽거나 쓸 수 있습니다.

HALT 신호는 특정 조건이 만족되었을 때 프로그램을 종료하기 위해서 사용됩니다. 종료 조건은 0x00008067 instruction을 받았을 때 RF_RD1 register의 값이 0x0000000c이므로 두 조건을 모두 만족하면 HALT를 1로 바꿀 수 있도록 해야 합니다.

NUM_INST은 core에서 실행시켰던 instruction의 개수입니다. 각각의 Testcase는 현재까지 실행시킨 instruction의 개수를 기반으로 그 때에 OUTPUT_PORT에 나와야 되는 output value를 비교하도록 만들어져 있습니다. 따라서 정상작동을 확인하기 위해서는 NUM_INST와 OUTPUT_PORT에 값이 정확하게 나오도록 해야합니다.

Instruction memory 생성

Instruction memory는 hex file을 불러서 초기화 됩니다. ROMDATA에 여러분 컴퓨터에서 hex file이 위치한 장소를 적어 놓으셔야 합니다. path에는 한글이 포함되면 안됩니다. AWIDTH가 10, SIZE가 1024로 초기화 됩니다. 해당 option으로 초기화를 시키면 memory module의 ram에 총 2^10개의 indexing할 수 있는 entry가 만들어지고, 각각의 entry는 4byte로 만들어 집니다. 각각의 entry에 instruction이 하나씩 들어가도록 구성되어 있습니다.

Instruction memory 또한 CLK에 의해 동기화되고, I_MEM_CSN input을 받습니다.

Instruction의 경우 byte단위로 불러들이지 않고, write operation이 없기 때문에 BE는 0, WEN는 1, DI는 z로 고정되어서 만들어 지게 됩니다. DOUT은 memory에서 나오는 output이고, ADDR에는 I_MEM_ADDR가 들어가게 됩니다. 이때 I_MEM_ADDR[11:2]로 accessing을 하도록 wire가 연결되어 있기 때문에 I_MEM_ADDR의 상위 10bit로 각각의 entry를 access하도록 design되어 있다는 것을 알 수 있습니다.

Data memory 생성

Data memory는 AWIDTH가 12, SIZE가 4096으로 초기화 됩니다. 해당 option으로 초기화를 시키면 memory module의 ram에 총 2^12개의 indexing할 수 있는 entry가 만들어지고, 각각의 entry

는 4byte로 만들어 집니다. 각각의 entry에 data가 하나씩 들어가도록 구성되어 있습니다.

Data memory 또한 CLK에 의해 동기화되고, D_MEM_CSN input을 받습니다.

Data의 경우 instruction에 따라 byte 단위로 불러 들이거나 쓸 수 있어야 하기 때문에 core에서 나오는 D_MEM_BE를 통해서 조절을 하도록 연결되어 있고, write operation을 enable할 수 있도록 D_MEM_WEN을 받아서 D_MEM_DI를 ram에 쓰도록 했습니다. DOUT은 memory에서 나오는 data output이고, ADDR에는 D_MEM_ADDR가 들어가게 됩니다. 이때 D_MEM_ADDR는 wire를 slicing 하지 않고 그대로 넣는 점에 주의하여 core에서 D_MEM_ADDR를 만들어 내야 합니다.

Data memory와 instruction memory는 주어진 address와 BE, WEN과 같은 interface를 이용해 사용하면 clock에 동기화되어 1 cycle 뒤에 값을 내보내거나 쓸 수 있도록 이미 구성되어 있습니다.

Register file 생성

Register file은 각각의 DWIDTH, MDEPTH, AWIDTH가 32, 32, 5로 초기화 되어 32개의 general purpose register가 있으며 이가 각각 32bit data를 가질 수 있도록 구성되어 있습니다.

RSTn으로 초기화되며, CLK에 동기화되어 있습니다. RF_RA1, RF_RA2가 각각 접근하고자 하는 source register 1, 2의 number이며, 이 값은 RF_RD1, RF_RD2로 나오게 됩니다. Write를 하기 위해 write enable인 WE가 있으며, WD에 있는 data를 register file에 반영합니다.

각각의 register의 initial value는 REG_FILE 에서 초기화가 되며, Register들의 초기값에 대해서는 RSTn signal만 주면 초기화가 되기 때문에 따로 신경 쓰지 않아도 됩니다.

Testcase의 채점원리

모든 instruction의 실행 마다 output value를 확인하는 것이 아닙니다.

NUM_INST를 확인하여 지정된 수만큼의 instruction이 실행이 되었으면, OUTPUT_PORT를 test case value와 비교하여 채점을 하게 됩니다.

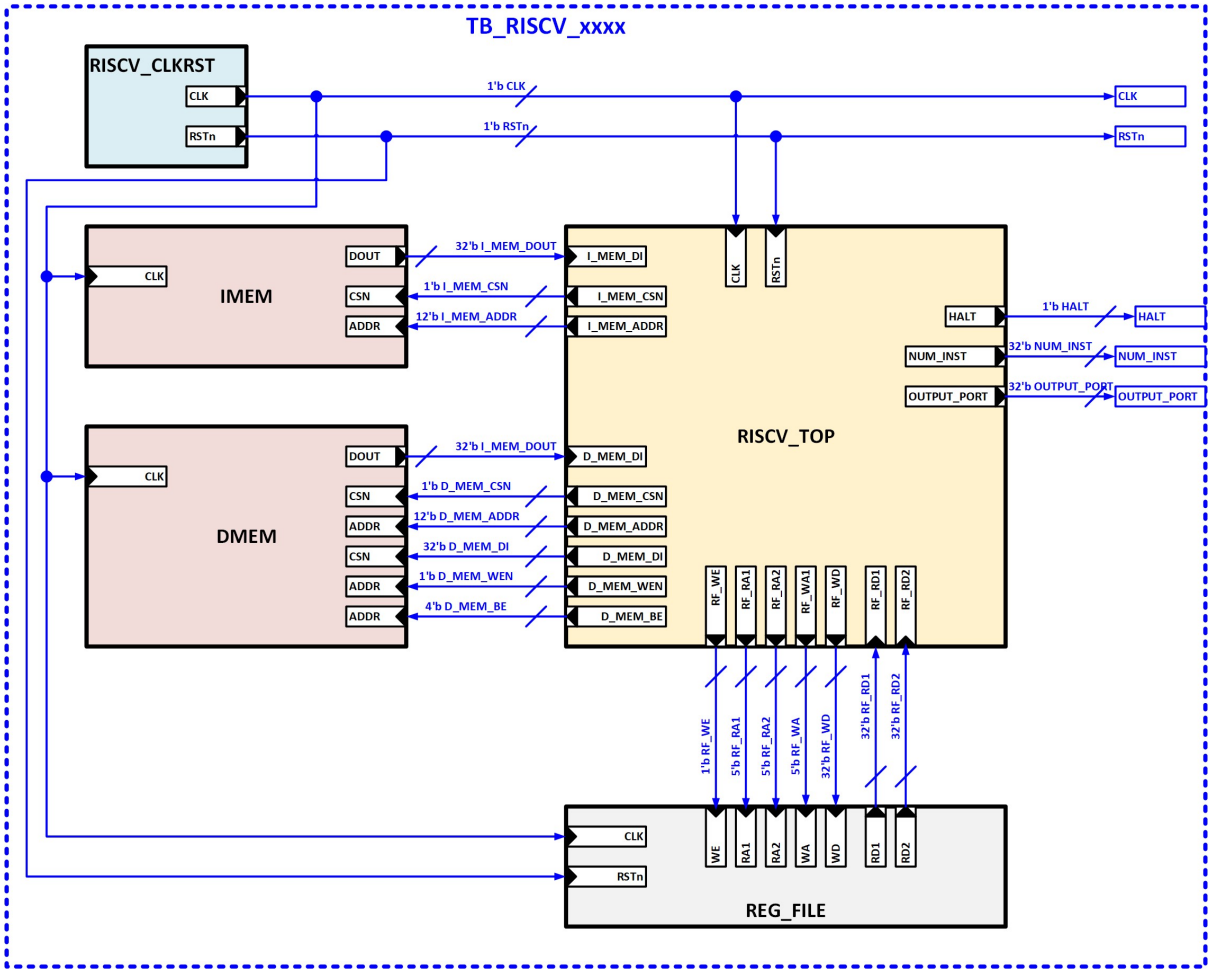

Figure 1 TB_RISCV의 구조