Explanation of Final Project Testbench by TAs

Advisor: Lih-Yih Chiou

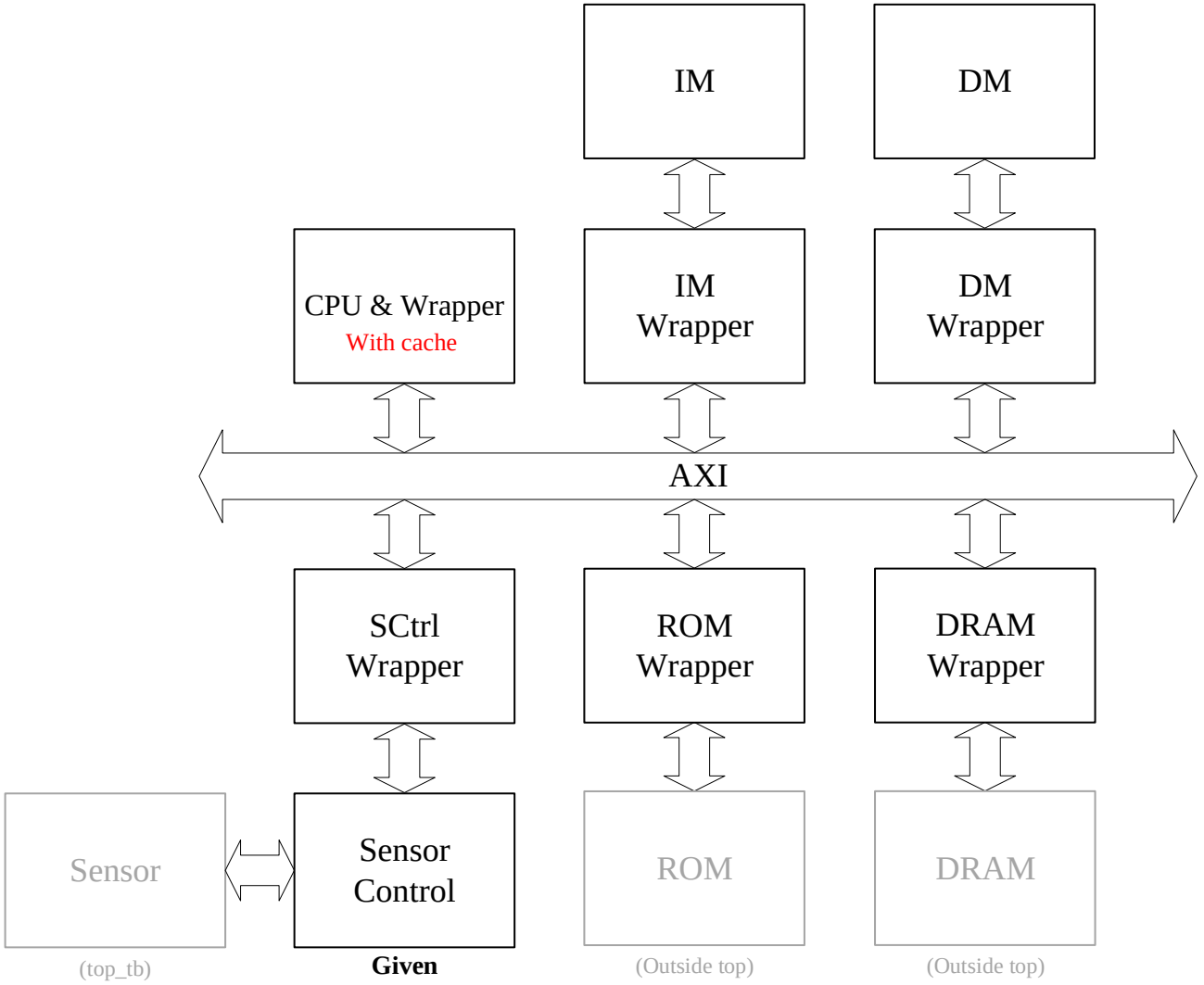
Date: 2022/12/28

Outline

- System
- Verification
- Simulation Commands

System

Basic System Architecture



Slave Configuration

Table 1-1 : Slave configuration

NAME	Number	Start address	End address
ROM	Slave 0	0x0000_0000	0x0000_3FFF
IM	Slave 1	0x0001_0000	0x0001_FFFF
DM	Slave 2	0x0002_0000	0x0002_FFFF
sensor_ctrl	Slave 3	0x1000_0000	0x1000_03FF
DRAM	Slave 4	0x2000_0000	0x207F_FFFF

- You should design all slave wrappers !!
- You can redesign memory size or add new slave devices
  - For SRAM, using the Memory Compiler
  - For ROM & DRAM, modifying the RTL code
- Remember to modify
  - The memory configuration in the link.ld for each program
  - The synthesize/APR related files if extra memory modules are used.

Sensor Controller (1/2)

- Sensor generates a new data every 1024 cycles
- Sensor controller stores data to its local memory
- When local memory is full (64 data), sensor controller will stop requesting data (sensor_en = 0) and assert interrupt (sctrl_interrupt = 1)
- CPU load data from sensor controller and store it to DM
- Write non-zero value in 0x1000_0100 or 0x1000_0200 to enable stcrl_en or stcrl_clear

Address	Mapping
0x1000_0300 – 0x1000_03FF	mem[0] – mem[63]
0x1000_0100	stcrl_en
0x1000_0200	stcrl_clear

Sensor Controller (2/2)

- Any access from address 0x1000_0000 to 0x1000_03FF should be uncacheable, which means that D-cache should pass data between CPU and CPU wrapper without writing it into cache memory.
- Add following code in L1C_data.sv and use this logic to decide whether to write valid bit, tag array, and data array in L1C_data when read miss.

```
logic cacheable;
always_comb cacheable =  (core_addr[31:16] != 16'h1000);
```

Testbench

※Attention
All kinds of testbench will be tested, including those your own APs

Testbench provided by TA

- This set of testbenches is provided to verify your Final Project system, and the following Verification and Simulation are all about this.

Testbench provided by yourself

- This is the testbench required by each group to display its Final Project and must be designed by each group.
- To verify the testbench, you should modify the Makefile with adding new command options(rtl/syn/pr).

```
81 syn0: | $(bld_dir)$
82 >.@if [ $$(echo $(CYCLE) '>' 20.0 | bc -l) -eq 1 ]; then \$
83 >.>.echo "Cycle time shouldn't exceed 20"; \$
84 >.>.exit 1; \$
85 >.fi; \$
86 >.make -C $(sim_dir)/prog0/; \$
87 >.cd $(bld_dir); \$
88 >.irun $(root_dir)/$(sim_dir)/top_tb.sv \$
89 >.-sdf_file $(root_dir)/$(syn_dir)/top_syn.sdf \$
90 >.+incdir+$(root_dir)/$(syn_dir)+$(root_dir)/$(inc_dir)+$(root_dir)/$(sim_dir) \$
91 >.+define+SYN+prog0$(FSDB_DEF) \$
92 >.-define CYCLE=$(CYCLE) \$
93 >.-define MAX=$(MAX) \$
94 >.+access+r \$
95 >.+ncmaxdelays \$
96 >.+prog_path=$(root_dir)/$(sim_dir)/prog0$
```

Remember to add “+ncmaxdelays” for syn/pr simulation

Verification (1/6)

- Prog0
  - Booting + Testing 45 instructions (Provided by TA)
- Prog1
  - Booting + Interrupt mechanism verification(Sensor controller)

Verification (2/6)

- Booting
  - The booting program is stored in ROM
  - Moves data from DRAM to IM and DM

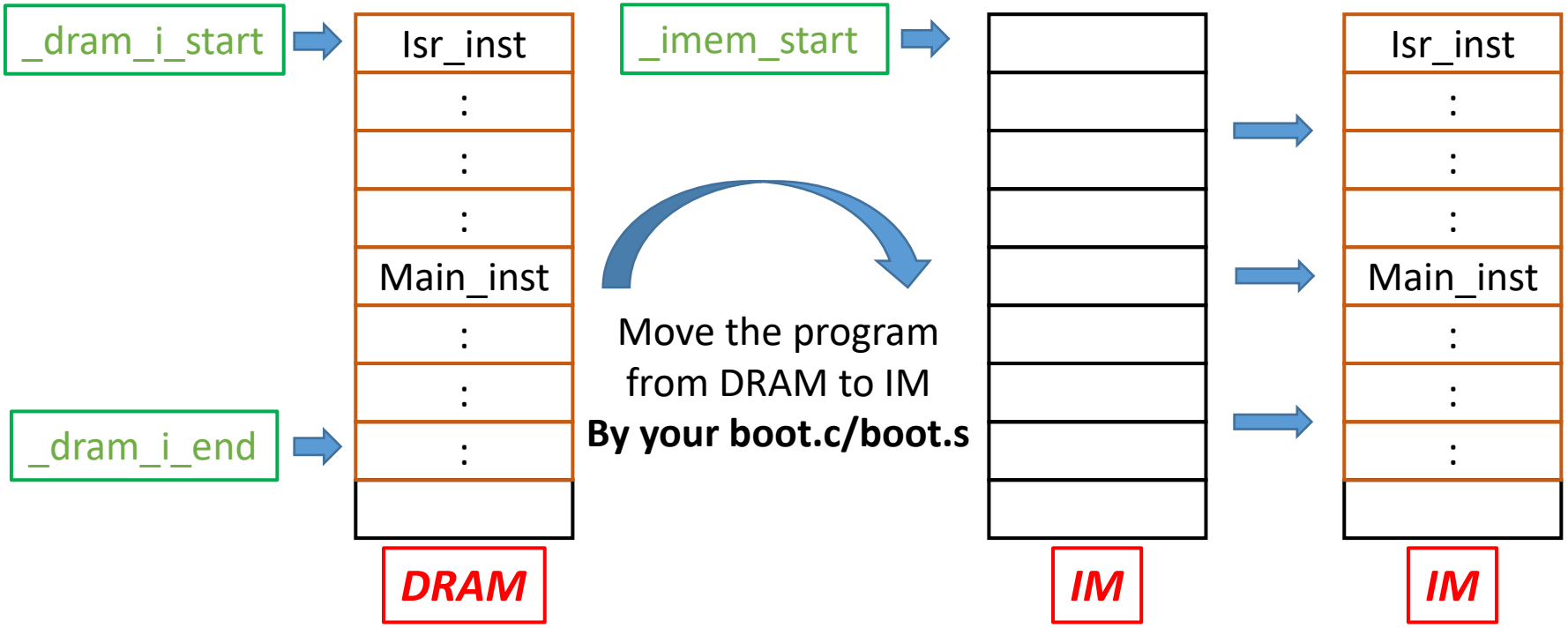
```
extern unsigned int _dram_i_start;
extern unsigned int _dram_i_end;
extern unsigned int _imem_start;

extern unsigned int __sdata_start;
extern unsigned int __sdata_end;
extern unsigned int __sdata_paddr_start;

extern unsigned int __data_start;
extern unsigned int __data_end;
extern unsigned int __data_paddr_start;
```

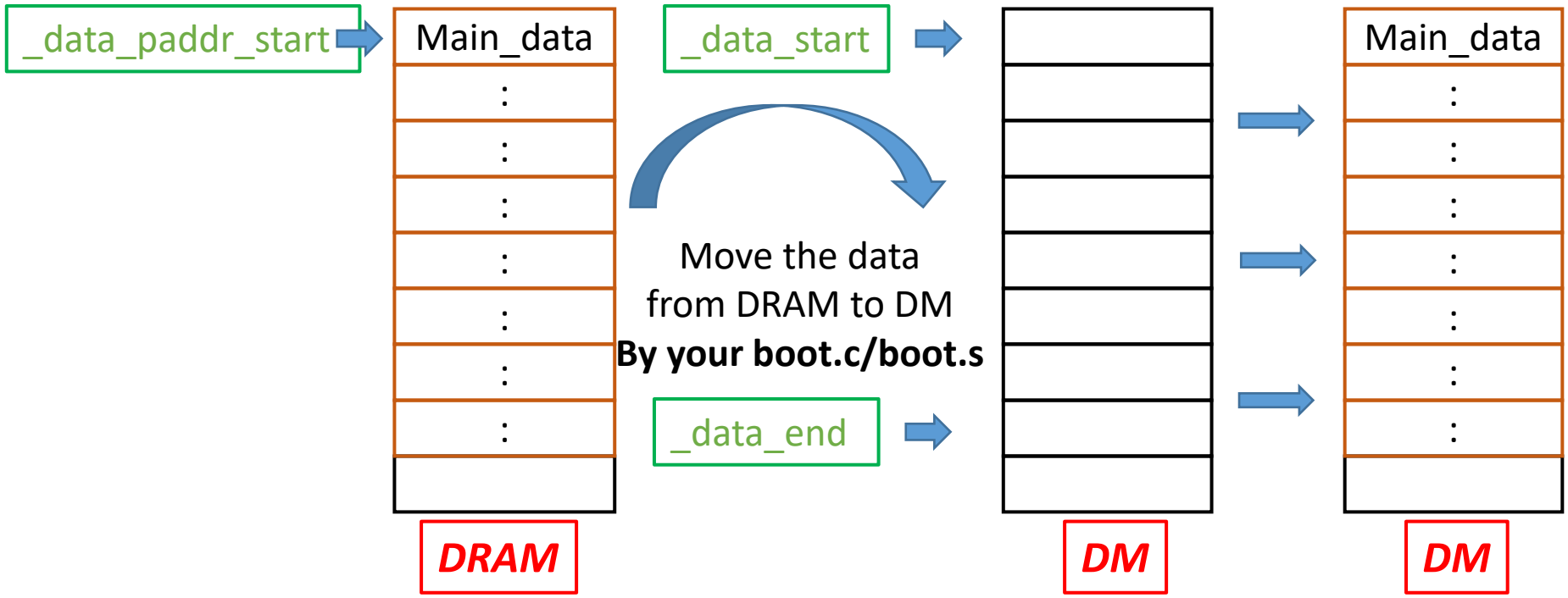
Verification (3/6)

- _dram_i_start = instruction start address in DRAM.
- _dram_i_end = instruction end address in DRAM.
- _imem_start = instruction start address in IM



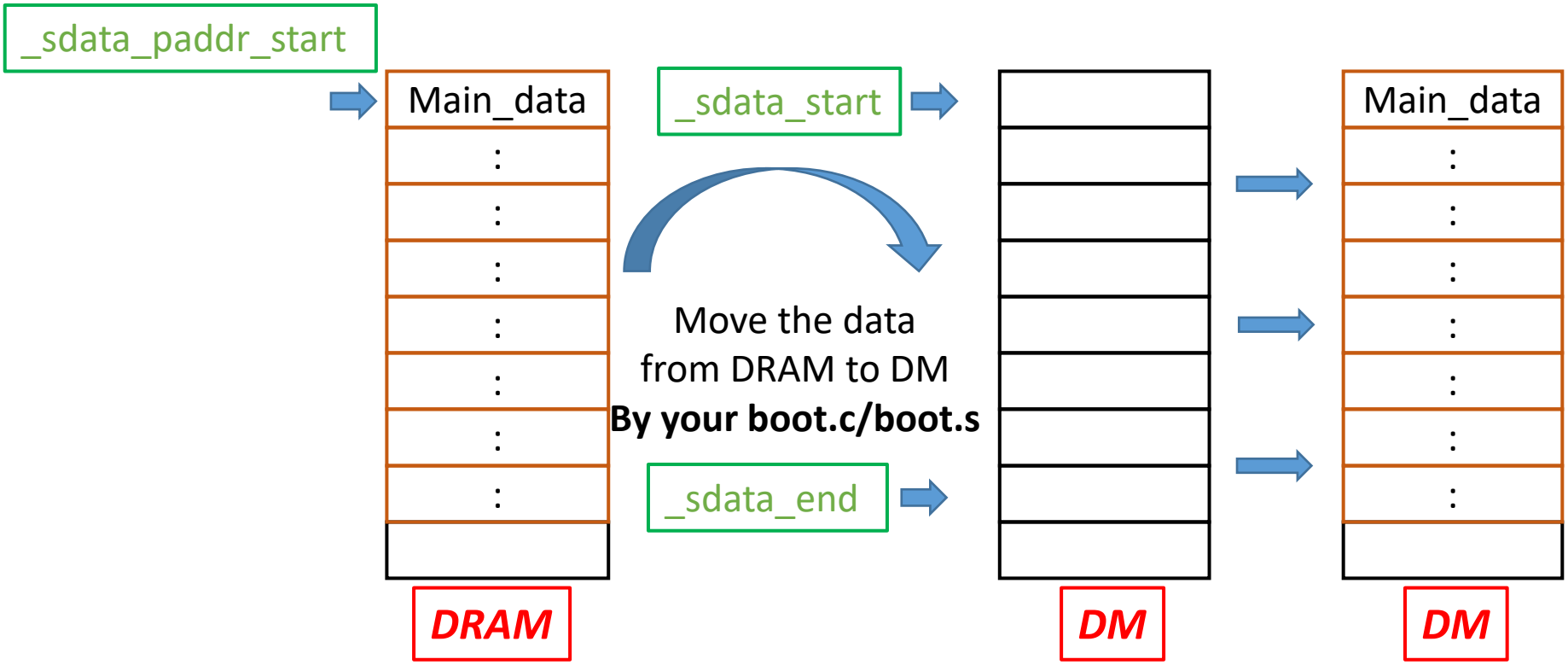
Verification (4/6)

- _data_start = Main_data start address in DM.
- _data_end = Main_data end address in DM.
- _data_paddr_start = Main_data start address in DRAM



Verification (5/6)

- _sdata_start = Main_data start address in DM.
- _sdata_end = Main_data end address in DM.
- _sdata_paddr_start = Main_data start address in DRAM



Verification (6/6)

- Interrupt
  - When sensor controller is full, it will interrupt CPU
  - ISR (isr.S)will be activated and copy data to DM. Then, reset the counter of sensor controller
  - When copy is done, ISR return to main program
  - After each group of data is copied, the copied data will be sorted.
    - This process executes 4 times.
- main.c
  - You can modify the source code if you need.
    - Copy function
    - Sort function
  - Remember to notify TA which part is changed and explain it!!!
    - Modification of the test data is not allowed.

Simulation (1/3)

Table B-1: Useful commands

Situation	Command	Example
RTL simulation for progX	make rtlX	make rtl0
Post-synthesis simulation for progX	make synX	make syn1
Dump waveform (no array)	make {rtlX,synX} FSDB=1	make rtl2 FSDB=1
Dump waveform (with array)	make {rtlX,synX} FSDB=2	make syn3 FSDB=2
Open nWave without file pollution	make nWave	
Open Superlint without file pollution	make superlint	
Open DesignVision without file pollution	make dv	
Synthesize your RTL code(You need write *synthesis.tcl* in *script* folder by yourself)	make synthesize	
Delete built files for simulation, synthesis or verification	make clean	
Check correctness of your file structure	make check	
Compress your homework to *tar* format	make tar	

Simulation (2/3)

Table B-2: Useful commands (Cont.)

Situation	Command
Open Innovus without file pollution	make innovus
RTL	make rtl_all
Post-synthesis	make syn_all
Post-layout	make pr_all

Simulation(3/3)

- You should verify by running the benchmark and remember to list the RTL-simulation time in “demo check list”.
- Hint : If your final project system has DMA , you can try to integrate your design and the testbench to improve the simulation time by your DMA .

```
DRAM[262654] = 7b8376ef, pass
DRAM[262655] = 7bb68171, pass

        ****************************
        **                        **       |__||
        **  Congratulations !!    **      / 0.0  |
        **                        **    /_____   |
        **  Simulation PASS!!     **   /^ ^ ^ \  |
        **                        **  |^ ^ ^ ^ |w|
        ****************************   \m___m__|_|

Simulation complete via $finish(1) at time 53546170 NS + 0
../sim/top_tb.sv:384     $finish;
ncsim> exit
```

※再次提醒，Demo時兩種的testbench都需要一併帶來測試。