# A Review on Score-boarding Technique Under Dynamic Scheduling

Nikita Saxena
Department of Computer Science& Engineering,
Rama University, Uttar Pradesh, Kanpur, India
saxena.nikita05@gmail.com

**ABSTRACT-**Dynamic Scheduling is a method in which hardware determines which instruction to execute.[1] It is based on data flow execution. There are basically two schemes for dynamic scheduling- a) Scoreboarding b) Tomasulo's Algorithm. It is the case study describing the working and implementation of scoreboard technique. Scoreboarding is a technique to allow instructions to execute out-of-order when there are sufficient resources and no data dependencies. It helps to overcome data hazards i.e. Write After Write (WAW) & Write After Read (WAR) hazards. Every instruction has to go through a special hardware known as scoreboard to complete its execution.[2]
**Keywords –**Hardware, Algorithm, DataHazards, Data Dependency, Scoreboarding, Dynamic Scheduling

## I. INTRODUCTION

The word scheduling means arranging the instructions in a proper manner so that they can complete their execution more efficiently. Scheduling is basically divided into two categories- a) Static scheduling in which compiler defines which instruction to execute next. In this type of scheduling the compiler does not change the order of execution of the instruction if either one instruction has nothing to do. b) Dynamic Scheduling in which hardware determines which instruction to execute next.

In dynamic scheduling, the schedule is not fixed at run time, rather it changes during execution as per the need of the instructions.

In dynamic scheduling, the instructions are executed as soon as their operands are available to them. Dynamic scheduling enables out-of-order execution to handle the problem of true dependency. But this creates Read After Write (RAW) hazards.[4].In dynamically scheduled pipeline, all instructions pass through in-order issue, but it has several advantages over static scheduling.

## II. ADVANTAGES OF DYNAMIC SCHEDULING

1. It can easily handle the dependencies which are not known at compile time. For example-Dependencies that involve memory references.
2. It simplifies the compiler.
3. It also allows the code that has been compiled for one pipelined, to execute on another pipeline.
4. Hardware speculation can be used that further leads to performance advantages that are built on dynamic scheduling.[4]

Dynamic scheduling can be implemented with the help of two approaches-

**a) Scoreboarding**
**b) Tomasulo's Algorithm**

## III. SCOREBOARDING TECHNIQUES

This technique was first used in 1964 in CDC 6600 computers to implement dynamic scheduling.[5][6] It is used to execute instructions out-of-order when the resources required by them are available and no data dependencies exist. It is a central location where information about all the instructions is kept[7].

It mainly checks two things-

a. Structural Hazards (it arises because of limited resources available in the processor).
b. True data dependency.[8]

The WAR & WAW hazards that do not arise in in-order-execution can arise in dynamic scheduling. To handle such type of data hazards, Scoreboarding technique is used. The main aim is to maintain the execution rate of one instruction per clock cycle [8] Scoreboard constructs a database for data dependencies. It keeps the information of every instruction from fetch to execute. It determines when and which instruction is to be executed from where i.e. beginning & end of the instruction (only when there is no data dependencies). It monitors all the instructions waiting to be dispatched. It also controls when an instruction has finished its execution and when it will write its result into the destination register [8]. Scoreboarding does not allow forwarding of the instructions. It is a centralized control of various operations like issue, operand reading, execution, write-back result and hazards detection.

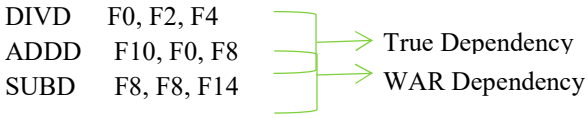
## IV. IMPLICATIONS OF SCOREBOARDING

1) Out of order completion of the instruction-WAR & WAW hazard detection & elimination.
2) Solution- Stall write to allow read operation to take place( Implemented in CDC 6600)
3) For WAW hazards- it must be detected and stalled in issue stage until the other completes its execution.
4) Needs to have multiple instructions in its execution phase.
5) It keeps track of dependencies , state or operations i.e.
   - Monitors each and every change in hardware.
   - Determines when to read operands, when to execute them , & when the result is to be written back into the registers
   - Hardware detection & resolution is centralized[8].

## V. STAGES IN SCOREBOARDING

Scoreboarding techniques involve 4 stages to complete its operation. These 4 stages are-

1. **Issue-** It is the first stage in which instructions are decoded. It checks for anavailable functional unit, structural hazards & WAW hazards. If structural or WAW hazard exist, then the instruction issue stalls and no further instruction is issued until all hazards are cleared.
2. **Read Operands**- It checks for the availability of source operands and also for no RAW hazard exist. It waits until no data hazards exist & resolves the RAW hazards dynamically.
3. **Execution**- The functional unit notifies the scoreboard when they had finished the execution. Then it starts to operate on the operands.
4. **Write Result-** After the execution of functional units , it checks for WAR hazards , if found, it instructs the functional units to stall the instruction until the hazard clears . Then after the final result is written back into the destination register.[9]




There are 3 parts in scoreboard-

1. **Instruction Status-** It defines the instruction is presently in which state out of 4 states .
2. **Functional Unit Status (FU)-** It has various parameters like-
   Busy- Defines that particular unit is executing
   Op- operation to perform
   Fi-Destination register
   Fj, Fk- Source Registers
   Qj, Qk- Functional units producing source registers Fj & Fk
   Rj, Rk- Flag indication when Fj & Fk are ready
3. **Register Result Status-** It defines which functional unit will write ratio into which register [10].

***Example-***

| Instruction | J | K |
|---|---|---|
| LD F6 | 34+ | R2 |
| LD F2 | 45+ | R3 |
| MULTI F0 | F2 | F4 |
| SUBD F8 | F6 | F2 |
| DIVD F10 | F0 | F6 |
| ADDD F6 | F8 | F2 |

LD requires 1 cycle latency, ADDD & SUBD are 2 cycle latency, MULTI -40 cycle & DIVD -10 cycle latency ( some realistic numbers)

- *Execution snapshot #1:*

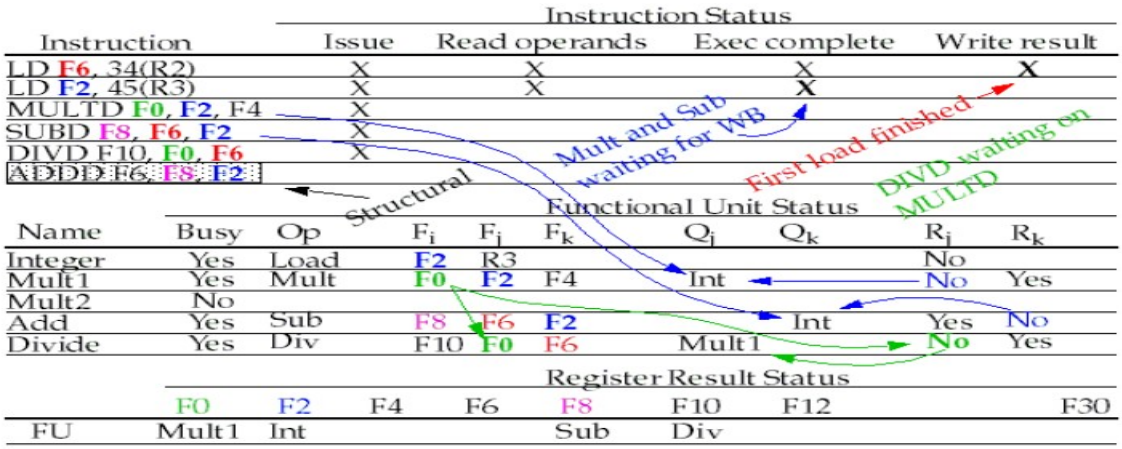



***Fig5.1: Execution Snapshot 1***

- *Execution snapshot #2:*

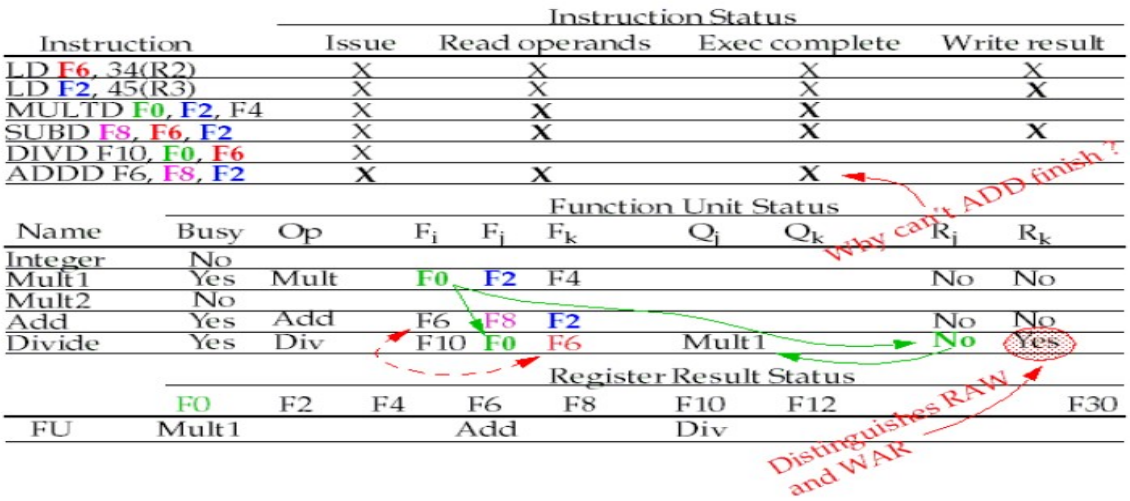



***Fig5.2: Execution Snapshot 2***

• *Execution snapshot #3:*

| | Instruction Status | | | |
|---|---|---|---|---|
| Instruction | Issue | Read operands | Exec complete | Write result |
| LD F6, 34(R2) | X | X | X | X |
| LD F2, 45(R3) | X | X | X | X |
| MULTD F0, F2, F4 | X | X | X | X |
| SUBD F8, F6, F2 | X | X | X | X |
| DIVD F10, F0, F6 | X | X | X | |
| ADDD F6, F8, F2 | X | X | X | X |

| | Function Unit Status | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Name | Busy | Op | $F_i$ | $F_j$ | $F_k$ | $Q_j$ | $Q_k$ | $R_j$ | $R_k$ |
| Integer | No | | | | | | | | |
| Mult1 | No | | | | | | | | |
| Mult2 | No | | | | | | | | |
| Add | No | | | | | | | | |
| Divide | Yes | Div | F10 | F0 | F6 | | | No | No |

| | Register Result Status | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | F0 | F2 | F4 | F6 | F8 | F10 | F12 | F30 |
| FU | | | | | | Div | | |

***Fig5.3: Execution Snapshot 3***

## VI. CONCLUSIONS

The key idea behind Scoreboarding technique is to allow instructions to execute out of order in case of no data dependencies. Scoreboard technique also prevents WAR hazards & handles RAW hazards through registers. The major limitation of scoreboard is that if we can't find independent instructions to execute, scoreboarding (or any dynamic scheduling scheme for that matter) helps very little [11]. It allows code to be compiled for one pipelined and can efficiently run on other pipelines. Astronautics ZS-1 is a FORTAN system which has been implemented on the CDC 6600 technology which is the implementation of Scoreboarding technology [12].

**REFRENCES**

[1]https://www.cs.umd.edu/class/fall2001/cmsc411/projects/dynamic/intro.html
[2]http://web.cs.iastate.edu/~prabhu/Tutorial/PIPELINE/dynamSchedTech.html
[3] https://en.wikipedia.org/wiki/Scoreboarding
[4]https://www.youtube.com/watch?v=GxesVUkmSLA

[5] https://en.wikipedia.org/wiki/Scoreboarding

[6]https://www.cs.umd.edu/class/fall2001/cmsc411/projects/dynamic/scoreboard.html

[7]https://www.cs.umd.edu/class/fall2001/cmsc411/projects/dynamic/scoreboard.html

[8]https://www.youtube.com/watch?v=GxesVUkmSLA

[9]https://kiitcseblog.files.wordpress.com/2017/03/lecture08-scoreboarding.pdf

[10]http://cgi.di.uoa.gr/~halatsis/Advanced_Comp_Arch/Patterson/Lec03-ilp_handout.pdf

[11]http://eceresearch.unm.edu/jimp/611/slides/chap4_3.html

[12]www.ece.umd.edu/courses/enee646/DynamicScheduling.