Andy, Yu-Guang Chen

# EE6094 CAD for VLSI Design
# Programming Assignment 2 (Due: 23:59:59, 2022/04/14)

- 20220324 We modify the output format.

## Introduction

In the front-end design phase, behavior synthesis converts high-level description of a design to the corresponding netlist. To meet performance requirement or reduce area/power overhead, the behavior synthesis process will try to either reduce computation cycles or minimize computational unit needed. This process is called **Scheduling**. The goal of scheduling is to determine when each operation should be executed so that the above goals can be achieved. There are various scheduling algorithms. In this Programming Assignment, you are asked to implement "**Force-Directed**" method to solve a performance-constrained resources minimization problem.

## Background

In integrated circuit design flow, high-level synthesis is a step in the standard design cycle. It is a process to convert the algorithm-level or behavior-level description of a circuit design specification into a circuit structure description under certain constraints. In particular, scheduling and binding mainly determine the performance and resource size of the generated RTL.

Scheduling plays a central role in the behavioral synthesis process, which automatically compiles high-level specifications into optimized hardware implementations. Scheduling, which exploits the parallelism in the behavior-level design and determines the time at which different computations and communications are performed, is commonly recognized as one of the most important problems in behavioral synthesis. However, finding an optimal schedule is intractable in general.

## Problem definition

You are asked to implement a "**Force-Directed**" latency-constrained scheduler to minimize the resource required. Assume there are only two types of operations: addition and multiplication. An addition takes 1 time unit, and a multiplication takes 3 time units. Moreover, only two type of computing resources are available, adder and multiplier. To simplify the problem, we assume the adder can only perform addition function and the multiplier can only perform multiplication function. You are given an input file which contains the sequencing graph information of a circuit and the maximum allowed latency. Your program needs to find a scheduling result which leads to minimal resource (hardware) requirement while satisficing timing requirement. Then a corresponding output file should be generated with the information of resource requirement and scheduling result.

## Input file format

The first three lines list the maximum latency, total number of nodes, and total number of edges.

The next $n$ ($n$ = total number of nodes) lines list the node information. Each of these $n$ lines contains a unique node number followed by a symbol. There are 4 kinds of symbol: **i** represents an input node, **o** represents an output node, + represents an addition node, and * represents a multiplication node. Note that the order of node number is **NOT** guaranteed. After that, there are $m$ ($m$=number of edges) lines listing the edge information. Each of these $m$ lines contains a node number **u** followed by another node number **v** which represents a directed edge from **u** to **v**.

**Output file format**

The first line gives the number of adders of your solution. The second line gives the number of multipliers of your solution. ~~The third line gives the sum of the number of adders and multipliers.~~ The ($i$+2)th line lists the node(s) executed during the $i$th time unit in the increasing order w.r.t the node number. The output format is strict. The output file of your program should be named as *name***.out** where *name* is the input file name.

**Algorithm**

You must use **Force-Directed** algorithm to find best solution (minimum resource). You should be able to explain the algorithm you applied. When the condition is tight, always take the node with minimum node number. For more details, please reference papers [1] [2].

**Requirement**

You have to write this program in C or C++. You also need to write a **makefile** which can compile and execute your program directly. I will verify your program on our workstation. You should also write a report which should at least include a description of the data structure and algorithm you used and how to execute your program. Name you file as PA2_studID.cpp. Upload your code and report to ee-class. Some test input files will be announced on the ee-class soon.

**Makefile**

Your makefile should at least contain these 3 commands, which is (1) **make all**, (2) **make run**, and (3) **make clean**. The descriptions of each command is shown below.

(1) **make all**: This command will automatically compile your source codes and generate the corresponding objects and executable file.

(2) **make run**: This command will execute your executable file and run your program.

(3) **make clean**: This command will automatically remove all the objects and executable file generated by **make all**.

**Score**

Your assignment will be ranked and scored according to (1) the quality of your solution, (2) the runtime of your program, (3) the readability of your source code, (4) the report you wrote, and (5) the demo session.

**Example**

Below is a sample input file and a sample output file with comments added. Fig. 1 is the corresponding DFG and Fig. 2 is the result of scheduling.

Sample input file:

```
5        // maximum latency (the beginning time of end NOP)
13       // total number of nodes
17       // total number of edges
1 i      // node 1~3 are inputs
2 i
3 i
4 +      // node 4, 5 are Add operators
5 +
6 *      // node 6 is a Multiply operator
7 +
8 +
9 +
10 +
11 o     // node 11~13 are outputs
12 o
13 o
1 5      // an edge from node 1 to node 5
2 5      // an edge from node 2 to node 5
2 4
2 6
3 4
3 7
4 6
```

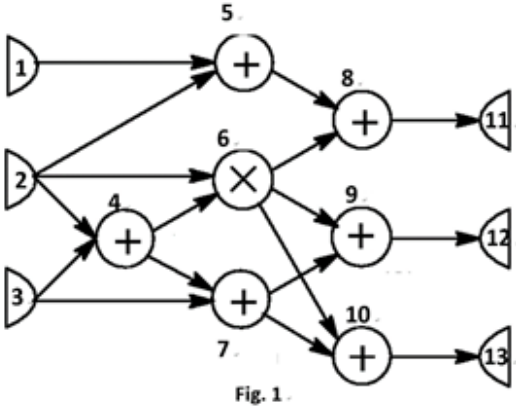

Fig. 1

4 7
5 8
6 8
6 9
6 10
7 9
7 10
8 11
9 12
10 13

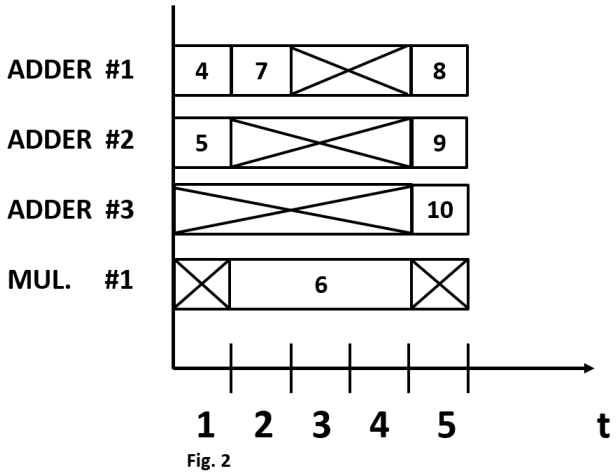

Fig. 2

Sample output:

3 // number of adders
1 // number of multipliers
4 5 // execute the operation of node 4 and 5 at t1
6 7 // execute the operation of node 4 and 5 at t2
6
6
8 9 10

**References**

[1] P. G. Pauline and J. P. Knight, "Force-Directed Scheduling in Automatic Data Path Synthesis," 24th ACM/IEEE Design Automation Conference, 1987, pp. 195-202, doi: 10.1145/37888.37918.

[2] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASICs," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 8, no. 6, pp. 661-679, June 1989, doi: 10.1109/43.31522.