OPT lab 1

Sylvain Collange

October 9, 2017

In this lab assignment, we will write parallel algorithms. No GPU programming yet, be patient!
We will implement the algorithms on a simple PRAM simulator, available at 'http://www.irisa.fr/alf/downloads/collange/cours/opt2018/pramsim.tar.gz' or 'http://tinyurl.com/opt18tp1'.

The simulated program consists of one function step. The step function is called once every cycle for every processor. It takes the processor number and the program counter as parameters. It is essentially a large switch on the value of the program counter, where every case is a different instruction.

At each step, the program can perform an arbitrary computation on the temporary registers *x, *y, *z and/or call a PRAM instruction. Our PRAM machine supports the following instructions:

- **Read(address)** accesses shared memory at the given address and returns the value read.
- **Write(address, value)** stores the given value at the given address in shared memory.
- **Halt** stops the current processor. Once all processors are halted, the execution ends.
- **JumpIf(target, value)** moves the control to instruction at target if value is non-zero (true). Otherwise, control moves to the next instruction.

Note the size of the memory is not unlimited, but equals $4 \times N$. If you need more, you need to change the size in pramsim.h.

As an example, vectoradd.c is a PRAM algorithm that adds two vectors of N elements together.

1 Warm-up

1. Check that vector add works as expected and make sure you understand it.
2. On paper, write a PRAM algorithm for an array reduction on $2N$ elements with $N$ processors. Do not write code yet!
3. Implement your algorithm in reduction.c and test it.
4. Now, design and write an algorithm for a parallel prefix computation on $N$ elements with $N$ processors.

2 Revisiting schoolbook addition

5. With the algorithm you learned at school, how many steps do you need to add two numbers of $n$ digits together? Can you parallelize it if you have up to $n$ processors?

We will write the numbers as $a_{n-1}a_{n-2}\ldots a_1a_0$ and $b_{n-1}b_{n-2}\ldots b_1b_0$, where each $a_i$ and $b_i$ is a decimal digit. For every $i$, there may be an incoming carry $c_i$ to be added to $a_i + b_i$. The outgoing carry $c_{i+1}$ is set to one when $a_i + b_i + c_i \geq 10$.

For each $i$, we now define the carry **propagate** bit $p_i$ and the carry **generate** bit $g_i$ as follows: Bit $g_i$ is set to one when the addition of $a_i + b_i$ will always cause a carry to the next digit regardless of the incoming carry, i.e. $a_i + b_i \geq 10$. Bit $p_i$ is set when the carry to the next digit is equal to the incoming carry, i.e. when $a_i + b_i = 9$. Note that all $g_i$ and $p_i$ can be computed independently.

Then, we can propagate carries using the following boolean function:

$$c_{i+1} = f_i(c_i) = g_i \lor (p_i \land c_i).$$
Now, for each $i$, we have to compute $c_i = f_{i-1}(f_{i-2}(\ldots f_1(f_0)\ldots))$. Note that this is a parallel prefix operation using function composition as the operator.

We define the following composition law between two sets of propagate-generate bits:

$$(p, g) \circ (p', g') = (p \land p', g \lor (p \land g')).$$

This law performs the function composition of carry propagation: it can “collapse” two carry propagation stages in the chain.

6. Check that the composition law is associative.

7. Write a PRAM algorithm that, given all $p_i, g_i$ and $c_0$, computes $c_i$ for each $i$ in $O(\log(n))$ steps. Hint: use a parallel prefix algorithm with the composition law as the operator.

8. Write the parallel addition algorithm and implement it in the simulator.