Yale University
Department of Computer Science

P.O. Box 208205
New Haven, CT 06520–8285

Slightly smaller splitter networks

James Aspnes[1]
Yale University

YALEU/DCS/TR-1438
November 2010

[1]Supported in part by NSF grant CCF-0916389.

Abstract

The classic renaming protocol of Moir and Anderson [4] uses a network of $\Theta(n^2)$ splitters to assign unique names to n processes with unbounded initial names. We show how to reduce this bound to $\Theta(n^{3/2})$ splitters.

1 Introduction

We show how to reduce the $\Theta(n^2)$ space and output namespace of renaming using a splitter network in the style of Moir and Anderson [4] to $\Theta(n^{3/2})$. The individual time complexity remains $\Theta(n)$, which is optimal for deterministic renaming given an unbounded initial namespace [3].

Our construction is based on alternating small Moir-Anderson grids with layers of small binary trees. The resulting renaming algorithm is not even remotely competitive with the tight output namespace and polylogarithmic time complexity of the best currently known randomized renaming algorithm [1], and requires more space, more time, and a larger output namespace than the best currently known deterministic algorithm [3] in the case where the initial names are sub-exponentially large. However, it uses less space than any other currently known algorithm when the initial names are unbounded, and might perhaps be useful as an initial stage before a better algorithm under such circumstances.

1.1 Splitter networks

A **splitter** [4] is a shared-memory object, implemented from two multi-writer atomic registers, with a single operation that returns a value **right**, **down**, or **stop**. Splitters satisfy the following conditions:

- In any execution of a splitter, at most one process obtains the value **stop**.
- If only one process invokes a splitter, that process obtains **stop**.
- If at least two processes invoke a splitter, at least one process obtains either **stop** or **right** and at least one process obtains either **stop** or **down**.

We can think of splitters as routing components of a network, where the **right** and **down** outputs send processes along virtual "wires" to further splitters. Figure 1 shows the splitter network used by Moir and Anderson [4]. It consists of two-dimensional triangular grid of $\binom{m}{2}$ splitters, containing splitters at all positions (i, j) where $0 \leq i, j \leq n$ and $i + j \leq m$, where each process enters the grid through the splitter at $(0, 0)$, and at each splitter (i, j) stops if it receives **stop**, proceeds to $(i + 1, j)$ if it receives **right**, and proceeds to $(i, j + 1)$ if it receives **down**. Moir and Anderson show that in any execution in which m processes follow this procedure, every process eventually receives **stop** at some splitter, before reaching one of the $2m$ output wires.

A simple explanation of this fact can be obtained by supposing that some process p reaches an output wire, and looking at the path it took to get their (see Figure 2). Each splitter on this path must handle at least two processes (or p would have stopped at that splitter). So some other process leaves on the other output wire, either **right** or **down**. If we draw a path from each of these wires that continues **right** or **down** to the end of the grid, then along each of these m disjoint paths either some splitter stops a process, or some process reaches a final output wire, each of which is at a distinct splitter. It follows that:

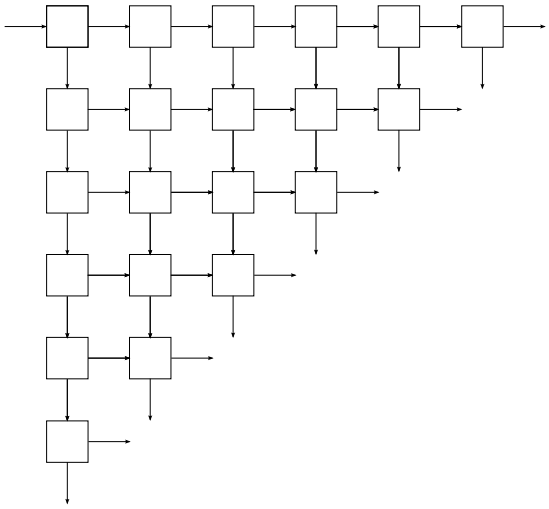

Figure 1: A 6×6 Moir-Anderson grid.

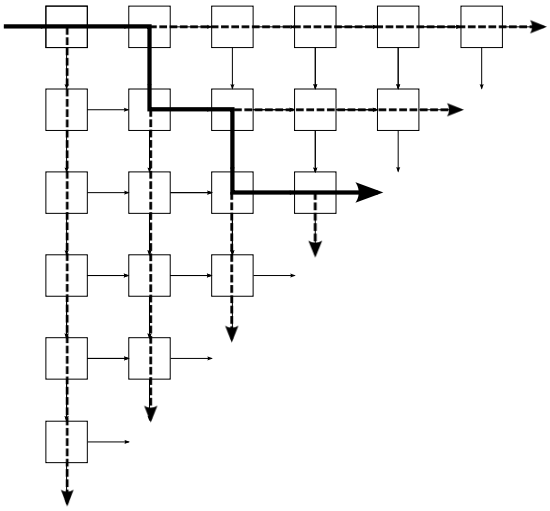

Figure 2: Path taken by a single process through a 6×6 Moir-Anderson grid (heavy path), and the 6 disjoint paths it spawns (dashed paths).

Lemma 1. *In an $m \times m$ Moir-Anderson grid, either all processes stop, or*

$$(\#\ \textit{of nonempty output wires}) + (\#\ \textit{of stopped processes}) \geq m+1, \textit{and} \tag{1}$$
$$(\#\ \textit{of nonempty output splitters}) + (\#\ \textit{of stopped processes}) \geq m. \tag{2}$$

An immediate corollary of the first bound (1) is that an $m \times m$ Moir-Anderson grid stops any set of m or fewer processes, because otherwise there are not enough to supply the $m+1$ processes in the inequality. The second bound (2) will be useful in our improved construction.

2 Blockers

Let an (m,k)**-blocker** be a splitter network with the property that if m processes enter the network at its designated input gate, at least k processes stop somewhere in the network. A single splitter is a $(1,1)$-blocker (but is only an $(m,0)$-blocker for any $m > 1$). An $m \times m$ Moir-Anderson grid is an (m,m)-blocker.

We will build an (n,n)-blocker out of $O(n^{3/2})$ splitters using a sequence of $\sqrt{n}$ stages, each of which is an $(n,\sqrt{n}\,)$-blocker. After each stage, all processes that have not stopped are fed into the single input of the next blocker. The overall structure is thus similar to the cascaded-tree splitter networks considered by [2] but we obtain much lower space complexity by using a combination of Moir-Anderson grids and binary trees in each stage instead of just a single large binary tree.

The essential idea of each $(n,\sqrt{n}\,)$-blocker is to use a Moir-Anderson grid of size $2\sqrt{n}$ and apply inequality (2) from Lemma 1 to show that at least $2\sqrt{n}$ processes either stop inside the grid or leave the grid through distinct output splitters. Since there are only n processes, fewer than $\sqrt{n}$ output splitters will get more than $\sqrt{n}$ processes. Deducting these overloaded splitters from the $2\sqrt{n}$ total gives at least $\sqrt{n}$ output splitters that either (a) get between 1 and $\sqrt{n}$ processes, or (b) correspond to a stopped process inside the grid. By attaching a $(\sqrt{n},1)$-blocker to both outputs of all $2\sqrt{n}$ splitters in the last layer, we stop at least one process for each splitter in class (a), for a total of $\sqrt{n}$ stopped processes.

We have not yet shown how to build a $(\sqrt{n},1)$-blocker. Here we can just use a binary tree with $\sqrt{n}$ splitters:

Lemma 2. *Any binary tree of m splitters is an $(m,1)$-blocker.*

Proof. By induction on m. A single splitter is a $(1,1)$-blocker. Given a binary tree of m splitters accessed by at least one process, either the root node stops a process, or it sends at least one process to each of its two subtrees. Let m_1 and m_2 be the sizes of the two subtrees; by the induction hypothesis, the subtrees are $(m_1,1)$ and $(m_2,1)$ blockers, respectively. So for no process to be blocked, we must send at least m_1+1 processes to the first subtree and m_2+1 processes to the second, for a total of $m_1+m_2+2 = m+1$ processes. It follows that the full tree is an $(m,1)$-blocker. □

Figure 3 shows an example of an $(n,\sqrt{n}\,)$-blocker constructed in this way. This uses $\binom{2\lceil\sqrt{n}\,\rceil}{2} = (2+o(1))\,n$ splitters for the Moir-Anderson grid, plus $(2\lceil\sqrt{n}\,\rceil)\lceil\sqrt{n}\,\rceil = (2+o(1))\,n$ splitters for the output blockers. The depth of the blocker is $2\lceil\sqrt{n}\,\rceil + \lceil \lg n \rceil = (2+o(1))\sqrt{n}$. Summarizing:

Lemma 3. *For any n, there is an $(n,\sqrt{n})$-blocker with $(4+o(1))n$ splitters and depth $(2+o(1))\sqrt{n}$.*

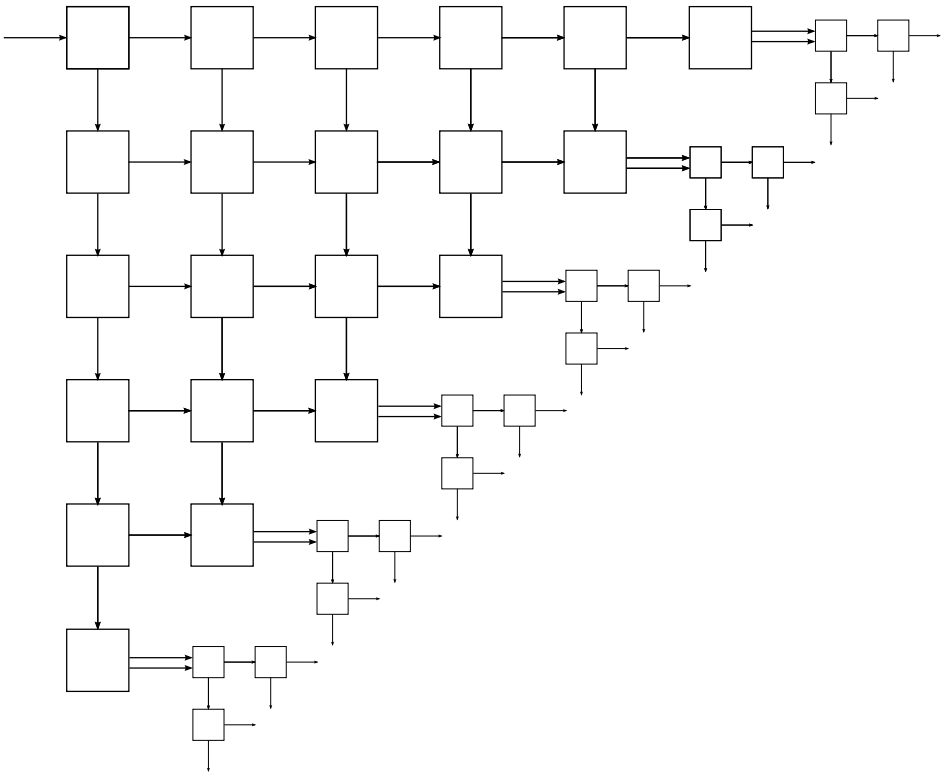

Figure 3: A $(9, \sqrt{9})$-blocker, consisting of a 6×6 Moir-Anderson grid with a $(3, 1)$-blocker on each output wire, implemented as a binary tree of splitters.

3 The full splitter network

To obtain the full splitter network, we iterate our $(n, \sqrt{n})$-blocker $\sqrt{n}$ times. Since each blocker stops at least $\sqrt{n}$ processes, every process stops at some stage. Summing the size and depth of the blockers over all $\sqrt{n}$ iterations gives:

Theorem 4. *For any n, there is a network of $(4 + o(1))\, n^{3/2}$ splitters with depth $(2 + o(1))\, n$ that solves renaming deterministically for n processes.*

Though we have assumed a known, fixed bound on the number of processes n, it is not hard to see that the algorithm could be made adaptive by stringing together geometrically increasingly large networks, with processes that fail to obtain a name in one network falling through to the next. This would give names in the range $O(k^{3/2})$ and time complexity $O(k)$, where k is the number of participating processes.

3.1 Conclusions

We have shown that it is possible to build a splitter network that assigns names to n processes in $O(n)$ individual work using $O(n^{3/2})$ space (and names). Because of the lower bound of Chlebus and Kowalski [3], improving the time complexity is not possible using a splitter network, but it may be that further improvements to the structure of the network could reduce the space complexity.

References

[1] Dan Alistarh, Hagit Attiya, Seth Gilbert, Andrei Giurgiu, and Rachid Guerraoui. Fast randomized test-and-set and renaming. In Nancy A. Lynch and Alexander A. Shvartsman, editors, *DISC*, volume 6343 of *Lecture Notes in Computer Science*, pages 94–108. Springer, 2010.

[2] Hagit Attiya, Fabian Kuhn, C. Greg Plaxton, Mirjam Wattenhofer, and Roger Wattenhofer. Efficient adaptive collect using randomization. *Distributed Computing*, 18(3):179–188, 2006.

[3] Bogdan S. Chlebus and Dariusz R. Kowalski. Asynchronous exclusive selection. In Rida A. Bazzi and Boaz Patt-Shamir, editors, *PODC*, pages 375–384. ACM, 2008.

[4] Mark Moir and James H. Anderson. Wait-free algorithms for fast, long-lived renaming. *Sci. Comput. Program.*, 25(1):1–39, 1995.