CS567 - Special/Advanced Topics in Algorithms

Assignment 5

Group members :

Yi Ren (002269013) Wentao Lu (002276355) Simin Li (002259850)

1. Some background

With the rapid development of CPU manufacturing technology, it has become more and more difficult to improve the single-core performance, that's why CPU manufacturers turned to develop multicore processors. As a consequence, people began to put more effort into parallel computation research, which means a problem can be solved by more than one processor simultaneously. In this way, reconfiguration was raised to satisfy the need for this kind of problem. Compared with traditional models, reconfigurable models can make better use of hardware resources, which means the processors could be used to run the tasks when available. As a result, reconfigurable models make it possible to solve the problem more efficiently.

2. The Definition of RMBM

The Reconfigurable Multiple Bus Machine (RMBM) [2], is one of the models for reconfiguration. Compared with R-mesh or even RN, RMBM use multiple buses to communicate with one processor. Suppose we have N processors and M buses in this model, we index the processors from 0 to N-1 and buses from 0 to M-1 (see figure 1).

We define the switch set to be the intersection of a processor and a bus because there could be more than one switches in the intersection. In this way, the RMBM model will have M*N switch sets in total.

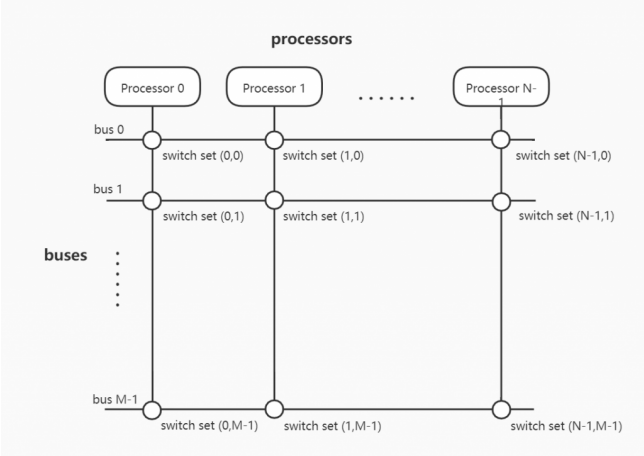

Figure 1. The architecture of RMBM

For a single switch, variant kinds of switches could be included [1].

● Connect switch: connect a port to the bus, like C(i,j,0) in figure 2, it connect the write port of processor i to bus j.

● Segment switch: disconnect or segment the bus, which means these switches is used to divide the bus into segments, like S(i,j,0) and S(i,j,1) in figure 2, which segment bus j to three parts.

● Fuse switch: connect the fuse line to buses. For example, f(i,j) fuse processor i to bus j, if processor i open some other fuse switch such as f(i,j+1) and f(i,j-1), then bus j-1, j and j+1 will be fused together.

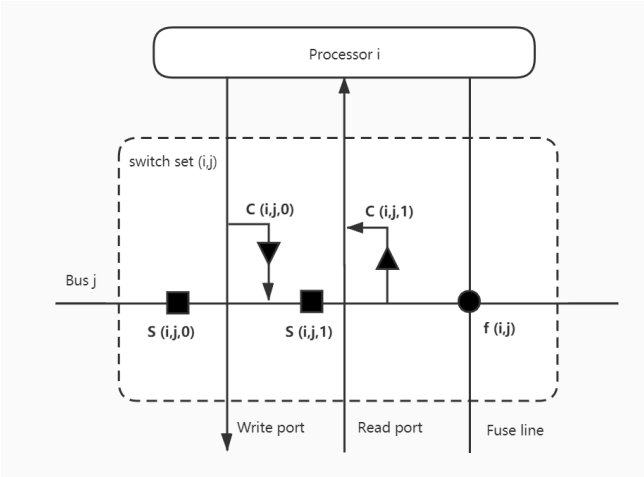

Figure 2. The structure of RMBM switch

Furthermore, we can specify RMBM by different memory accessing mode, that is concurrence and exclusion. So we have 4 kinds of RMBM:

● Exclusive-read exclusive-write (EREW) RMBM

● Concurrent-read exclusive-write (CREW) RMBM

● Concurrent-read concurrent-write (CRCW) RMBM

● Exclusive-read concurrent-write (ERCW) RMBM

While concurrent read is comprehensible, the concurrent write will need some strategy to resolve the conflict. As a consequence, 4 strategies were raised [1]:

● Common: when different processors write on the same bus simultaneously, the value should also be the same.

● Collision: when different processors write on the same bus simultaneously, a collision value should be written instead of the values from processors.

● Priority: when different processors write on the same bus simultaneously, the processor with a lower index has the priority to write.

● Combining: when different processors write on the same bus simultaneously, an operation should be done to all the values provided by processors. The operation could be one of the follows:

sum, product, logical conjunction, logical disjunction, logical exclusive disjunction, maximum and minimum.

3. Some Variants

Generally speaking, the RMBM model has two basic functions, segment, and fuse, which is mentioned above. Segment means the processor can divide a bus into separate segments while fuse means different buses could be connected. Base on segment and fuse, RMBM has four variants [3].

● Basic RMBM (B-RMBM), which is non-reconfigurable, just like a PRAM. Connect switch is the only choice to constitute its switch set. In this way, B-RMBM is not able to fuse or segment buses.

● Segmenting RMBM (S-RMBM), Compared to B-RMBM, it has the segment switch other than connecting switch. S-RMBM can segment but not fuse.

● Fusing RMBM (F-RMBM), the F-RMBM has both fuse switch and connect switch. F-RMBM can fuse buses, however, it cannot segment buses.

● Extended RMBM (E-RMBM), it has all the three switches mentioned, in this way, E-RMBM can fuse and segment buses.

4. Directed Variants

DRMBM, which denotes the Directed Reconfigurable Multiple Bus Machine, is another variant of RMBM [2]. The main difference between DRMBM and RMBM is the direction. In an RMBM model, a signal can be transmitted to all the buses fused, while in the DRMBM model, the signal can only be transmitted in a certain direction.

Actually, when we look into the structure of the DRMBM model, we will notice that every processor is connected by two fuse lines with different directions, as is shown in figure 3. One of them from top to bottom, the other keeps an inverse direction.

Take figure 3 as an example, if we connect all the buses to fuse line 1， and a signal from processor i is placed on bus j, then that signal will be only transmitted to bus k($k > j$) that connected to fuse line. However, RMBM can be emulated by DRMBM when fuse line 1 and fuse line 2 are synchronized, which means the two switches of bus j and fuse line need to keep the identical state so that the signal could be transmitted in both directions.

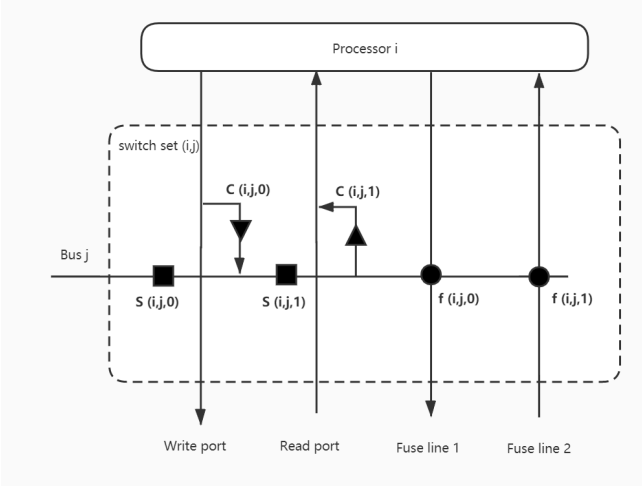

Figure 3. The structure of DRMBM switch

Similar to RMBM model, DRMBM also has some variants:

- Basic DRMBM (B-DRMBM)
- Segmenting DRMBM (S-DRMBM)
- Fusing DRMBM (F-DRMBM)
- Extended DRMBM (E-DRMBM)

5. Problems can be solved

As we know, many problems can be solved by parallel computation models with linear or even constant time. However, some of them can exploit the power of reconfiguration to a great extend, such as Permutation Routing, Counting Bits, Prefix Sums of Bits, Neighbor Localization and Chain Sorting [1]. One interesting thing is many problems can be solved by different kinds of reconfigurable models, with respective resources and complexity.

For RMBM models, some of them resemble R-Mesh models. For instance, the F-RMBM is similar to HVR-RMBM, as F-RMBM comprises vertical fuse lines and horizontal buses [1]. Also, the S-RMBM, which can be seen as a set of segmentable buses, can be used to simulate R-Mesh.

As mentioned above, some R-Mesh algorithms can be performed on S-RMBM in constant time, we will take Neighbor Localization and Chain Sorting as an example [1]. The object for Neighbor Localization is connecting active processors with a single bus, every active processor will hold a 'pointer' which point at the next active processor, as is illustrated in Figure 4. If a processor is active, the inner East port and West port will be disconnected, otherwise, they will be fused.

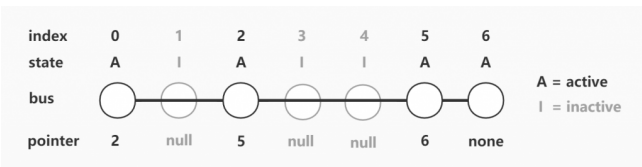

Figure 4. An example of Neighbor Localization

Neighbor Localization is a key point to solve Chain Sorting problem, suppose we have n integers from a_0 to a_{n-1}, we can use Chain Sorting to put these integers in a certain order. We denote the given integer by b bits, then the Chain Sorting problem could be solved with a few steps as following [1].

● Step 1: Denote a 2^b * n R-Mesh, each processor(0,j) in the first row holds the input values respectively, as Figure 5.

● Step 2: For every column i from 0 to n-1, broadcast the value from processor(i,0).

● Step 3: For every row i from 0 to 2^b-1, we note processor(i, j) active if and only if a_j == i.

● Step 4: For every row i from 0 to 2^b-1, we construct a list L_i with Neighbor Localizatin, to indicate the order of same input values.

● Step 5: For every list L_i in Step 3, we denote p(j) as the pointer which points to the next integer in the list, as illustrated in Figure 6 stage 1.

● Step 6: For a certain list L_i, send the first and the last integer to processor(i,0). If no data is sent to processor(i,0), we will note it as inactive.

● Step 7: Connect all the lists in ascending order, which means the last integer of L_i will point to the first integer of L_k, when processor(k,0) is the next active node for processor(i,0). As illustrated in Figure 6 stage 2 and 3.

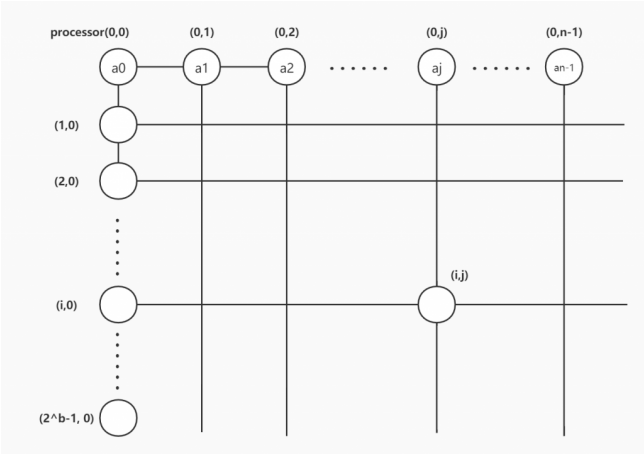

Figure 5. An example of R-Mesh

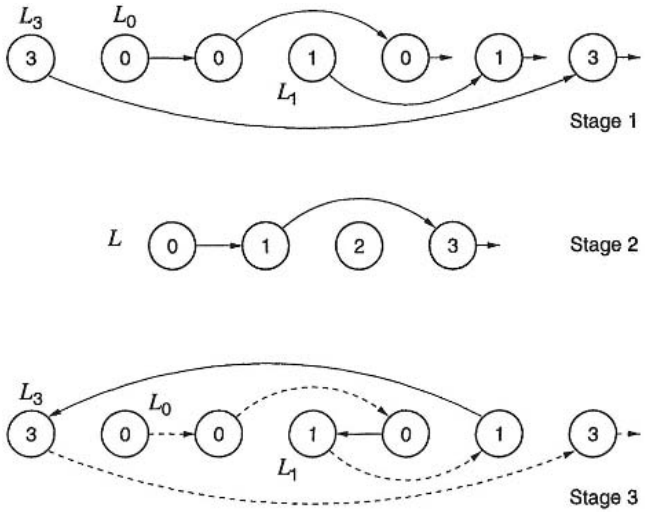

Figure 6. Three main stages of Chain Sorting [1]

As mentioned, we can perform this algorithm on an S-RMBM model, which has n processors and 2^b buses. Considering the algorithm runs on R-Mesh in constant time and S-RMBM can perform Neighbor Localization in constant time, it can run on RMBM in sub-linear time.

On the one hand, RMBM models use fewer resources (processors) than R-Mesh, which seems to be economic. On the other hand, it is slower than its R-Mesh counterparts, since a processor in RMBM models is permitted to set only one switch at a time, which means time needs to be spent on the switch setting.

6. Relation between RMBM and shared memory models

The shared memory model is one of the conventional models for parallel computation, all the processors share a common memory, which means any processor has access to any memory unit. Parallel Random Access Machine (PRAM) is one of the shared memory models, it is widely used for its concise and comprehensible architecture. Broadcast with Selective Reduction (BSR) [4] is also a shared memory model, it is based on the CRCW PRAM model with a unique combining collision resolution. Obviously, the main difference between RMBM and shared memory models is reconfiguration features, such as segment and fuse. PRAM models are non-configurable whereas RMBM models can change the connectivity between processors and buses with reconfiguration.

When we talk about the relation between reconfigurable models and shared memory models, one important thing is the computational power comparison. Here's a definition of computational power [1], for a problem of size N and for some constant $c > 0$, a polynomially bounded instance of a model has $O(N^c)$ elements (processors, wires, gates, etc.). Model M_1 is as powerful as model M_2 if, for every problem that an instance of M_2 can solve, there is an instance of M_1 that can solve the problem as fast as M_2. Model M_1 is more powerful (or model $M_2$is less powerful) if, in addition, an instance of M_1 can solve at least one problem faster than M_2.

Generally speaking, RMBM models are more powerful than PRAM models, although some of the PRAM models are as powerful as the weak RMBMs [1].

● B-RMBM, as the weakest RMBM, is as powerful as PRAM. As we know, the B-RMBM doesn't have the segment or fuse feature, therefore it is non-reconfigurable like the PRAM models.

● CRCW PRAM is as powerful as CREW S-RMBM. Actually, each step of a PRIORITY CRCW PRAM with P processors and S shared memory cells can be simulated in O(1) time on a CREW S-RMBM [P + S, S].

● F-RMBM and E-RMBM are more powerful than PRAM models. In fact, all of the fusing models are more powerful than PRAM models.

The computational power comparison is illustrated in Figure 7.

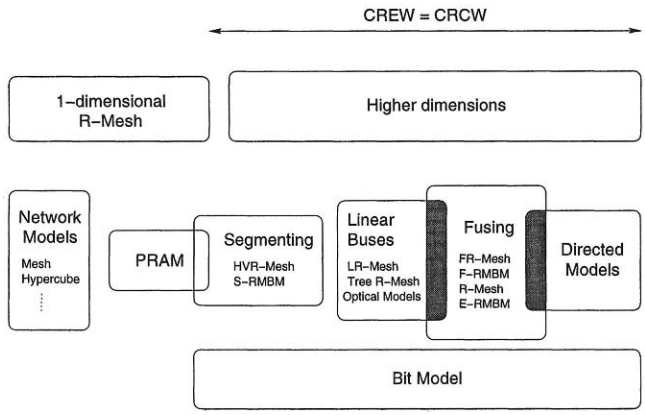

Figure 7. Power of conventional and reconfigurable models [1]

References:

[1] Ramachandran Vaidyanathan and Jerry L. Trahan, Dynamic.Reconfiguration.Architectures.and.Algorithms.Feb.2004, pp.23-57, pp.78-79, pp.91-92, pp.98-100, pp.327-329, pp.332-339.

[2] Stefan D. Bruda, The Graph Accessibility Problem and the Universality of the Collision CRCW Conflict Resolution Rule∗, pp.2-3.

[3] Mingxian Jin* and Johnnie W. Baker, On the Power of the Multiple Associative Computing (MASC) Model Related to That of Reconfigurable Bus-Based Models, pp.3-4.

[4] CATHERINE MAXCY CHOW, Broadcasting with Selective Reduction: An Alternative Implementation and New Algorithrns, pp.1-2.