Short Webcast

by

Gandhi Puvvada
### 10-element Arrays

#### Unsigned Sorted Array

<table>
<thead>
<tr>
<th>Value</th>
<th>Index</th>
<th>Array M</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0011</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>0111</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>1001</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>1010</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>1100</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>1110</td>
</tr>
</tbody>
</table>

#### Signed Sorted Array

<table>
<thead>
<tr>
<th>Array N</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1011</td>
<td>-7</td>
</tr>
<tr>
<td>1010</td>
<td>-6</td>
</tr>
<tr>
<td>1100</td>
<td>-4</td>
</tr>
<tr>
<td>1110</td>
<td>-2</td>
</tr>
</tbody>
</table>

- **Chunk #1**: 1001, 1010, 1100, 1110
- **Chunk #2**: 0001, 0010, 0011, 0101, 0110, 0111

Arrows indicate the mapping between chunks.
INI
I <= 0;
J <= 0;

DONE

ACK

Reset

Start

Start

LS2C
Locate
Start of
2nd chunk

C221
Copy
Chunk #2
of M
to
Chunk #1
of N

C122
Copy
Chunk 1
of M
to
Chunk 2
of N
<table>
<thead>
<tr>
<th>LS2C</th>
<th>Locate Start of the 2nd Chunk in array $M$ (increment $I$ until MSB of $M[I]$ is a &quot;1&quot;; but what if there are no numbers starting with a &quot;1&quot;?)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C221</td>
<td>Copy 2 to 1 (= Copy Chunk 2 of $M$ to Chunk 1 of $N$)</td>
</tr>
<tr>
<td>C122</td>
<td>Copy 1 to 2 (= Copy Chunk 1 of $M$ to Chunk 2 of $N$)</td>
</tr>
</tbody>
</table>

**Maximum Clocks?**
- All positive
- All negative
- One negative at the very end of the array $M$

**Minimum Clocks?**
- Task ends when $I = I_{\text{max}}$ or $J = J_{\text{max}}$ or both?

**Increment I first**
- Then increment $I$ and $J$.

**I is expected to reach its max first.**
- Then, roll it over to zero.
LS2C
Look for start of the 2nd Chunk

So, you only look for the 2nd chunk, or can also help with some transfers?

Some transfers?

Well how about one transfer, just before we move to the next state?

You mean, if I do find a negative number, I should transfer the first negative number?

Yes :)
If I reached 9, there can't be any more negative numbers. So go to C122.
Can we think of rolling over \( I \) to zero, when \( I \) reaches 9, irrespective of the state, unconditionally? 

Yes / No

What would you do if \( J \) reaches 9?

Go to DONE/Roll over/both/neither
Does $J$ reach $I_{\text{max}}$ first or does $I$ reach $J_{\text{max}}$ first?

How do we know that the CHUNK #2 in M has finished?
How do we know when Copy 1 to 2 is done?
RTL

Same or Different
Rollover of I or J
necessary
or
optional?
Implementation #2

Start

Reset

INI

I <= 0;
J <= 0;

DONE

ACK

ACK

Start

LS2C

Locate
Start of 2nd chunk

C221

Copy Chunk #2
of M

to Chunk #1
of N

C122

Copy Chunk 1
of M

to Chunk 2
of N

Combine
**LS2C** Same as before

**CBC** Copy Both Chunks
(Of course, Chunk after Chunk in one single state.)

Artificial separation of C221 and C122 is avoided! Both RTL and state transitions are simplified.

Once you start copying an element of $M$ to an element of $N$, do you need to distinguish between the chunks?!
Transition from LS2C to CBC:

Once you find a number with $M[I][3] = 1$?

or

Once you reach the bottom of the array $M$? $I = I_{\text{max}}$

or

any one of them?

or

both of them?
LS2C

INI
I <= 0;
J <= 0;

DONE

ACK

Start

Reset

LS2C

CBC

ACK
Transition from CBC to DONE:

Once I reaches Imax?

or

Once J reaches Jmax?

or

any one of them?

or

both of them?
INI:
I <= 0;
J <= 0;

DONE:

CSV:

Start:

Reset:

LS2C:

ACK:

DONE:

ACK:
Compare Imp #1 and Imp #2

Imp #1

INI
I <= 0;
J <= 0;

DONE

Start

Reset

Search, and
Copy one element if possible

LS2C

C221
Copy 2 to 1

C122
Copy 1 to 2

ACK

DONE

ACK
Compare Imp #1 and Imp #2

**Imp #2**

- **INI**
  - I <= 0;
  - J <= 0;

- **DONE**
  - ACK

- **LS2C**
  - Search, and
  - Copy one element if possible

- **CBC**
  - Copy both
Based on the number of clocks taken in different cases, the implementation #2 is _______________________________ (superior to / inferior to / sometimes superior sometimes inferior / always at the same level as) the implementation #1.

For a good hardware design, how many clocks do you spend in the CBC state?

______________________________________________________

Student #1:  Well there are 10 elements in the array. So 10 clocks.

Student #2:  No, not 10, it is 9 clocks. You copy one element in LS2C.

Student #3:  But, you may not have a chance to copy an element in LS2C always. What if there is no element with MSB = 1?

Student #4:  So, are you saying it is data-dependent? Sometimes, 9 clocks and sometimes 10 clocks?