Assessment Two

February 1, 2013

1

1.1

	A	B	C	D	E	F	G	H	I	J
c1	3	-2	0	0	0	0	0	0	0	0
c2	4	0	-1	0	0	0	0	0	0	0
c3	0	4	0	-3	0	0	0	0	0	0
c4	0	0	1	-2	0	0	0	0	0	0
c5	-2	0	0	1	0	0	0	0	0	0
c6	0	0	1	0	-1	0	0	0	0	0
c7	2	0	0	0	0	-4	0	0	0	0
c8	0	0	0	0	1	0	-2	0	0	0
c9	0	0	0	0	0	8	-2	0	0	0
c10	0	0	0	6	0	0	0	-2	0	0
c11	0	0	0	0	0	0	0	1	-3	0
c12	0	0	0	0	0	0	1	0	0	-1
c13	0	0	0	0	0	0	0	0	1	-1
c14	0	0	0	0	2	0	0	0	-X	0

In order the find the consumption rate X of actor I over channel c14 we need to determine the PASS (Periodic Admissible Sequential Schedule). Using the above matrix and multiplying by a column vector should equal zero. The consumption rate for each buffer can be found by finding this column vector.

$$\begin{array}{c} \\ c1 \\ c2 \\ \vdots \\ c14 \end{array} \begin{array}{c} \begin{array}{cccc} A & B & \cdots & J \end{array} \\ \begin{pmatrix} 3 & -2 & \cdots & 0 \\ 4 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \end{array} \begin{pmatrix} a \\ b \\ \vdots \\ j \end{pmatrix} = 0$$

Which gives the following simultaneous equations.

$$3a = 2b, 4a = c, 4b = 3d, c = 2d, d = 2a, c = e, 2a = 4f, e = 2g, 8f = 2g,$$

$$6d = 2g, h = 3i, g = j, i = j, 2e = Xi$$

And therefore

$$12a = 8b = 3c = 6d = 3e = 24f = 6g = 2h = 6i = 6j$$

And $X = 4$ We can get values for $a..j$ by finding the least common integer denominator.

$$a = 2, b = 3, c = 8, d = 4, e = 8, f = 1, g = 4, h = 12, i = 4, j = 4$$

1.2

The sequence can be found by trial and error. By summing the number of tokens displaced for each edge we can get the current number of tokens in each channel. This number must keep in the range of 0..9 and the number of tokens initially in $c5$ can be modified. By the end of the sequence, the number of firings for each buffer $A..J$ must be equal to $a..j$ that have been found above. Every channel must finish with the same number of tokens as it started so the sequence can be repeated.

		A	C	A	B	C	F	E	D	B	C	C	E	H	H	D	H	G	H	H	E	E	G	I	H
c1	0	3	3	6	4	4	4	4	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
c2	0	4	3	7	7	6	6	6	6	6	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4
c3	0	0	0	0	4	4	4	4	1	5	5	5	5	5	5	2	2	2	2	2	2	2	2	2	2
c4	0	0	1	1	1	2	2	2	0	0	1	2	2	2	2	0	0	0	0	0	0	0	0	0	0
c5	4	2	2	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2
c6	0	0	1	1	1	2	2	1	1	1	2	3	2	2	2	2	2	2	2	2	1	0	0	0	0
c7	0	2	2	4	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c8	0	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	0	0	1	2	0	0	0	0
c9	0	0	0	0	0	0	8	8	8	8	8	8	8	8	8	8	8	6	6	6	6	6	4	4	4
c10	0	0	0	0	0	0	0	0	6	6	6	6	6	4	2	8	6	6	4	2	2	2	2	2	0
c11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	2	3	3	4	5	5	5	5	2	3
c12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	2	2	2
c13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
c14	0	0	0	0	0	0	0	2	2	2	2	2	4	4	4	4	4	4	4	4	6	8	8	4	4

	B	C	E	I	C	D	J	J	H	H	H	E	C	I	C	E	D	E	G	H	H	H	I	J	G	J
c1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c2	4	3	3	3	2	2	2	2	2	2	2	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0
c3	6	6	6	6	6	3	3	3	3	3	3	3	3	3	3	3	0	0	0	0	0	0	0	0	0	0
c4	0	1	1	1	2	0	0	0	0	0	0	0	1	1	2	2	0	0	0	0	0	0	0	0	0	0
c5	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4
c6	0	1	0	0	1	1	1	1	1	1	1	0	1	1	2	1	1	0	0	0	0	0	0	0	0	0
c7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c8	0	0	1	1	1	1	1	1	1	1	1	2	2	2	2	3	3	4	2	2	2	2	2	2	0	0
c9	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	2	2	2	2	0	0
c10	0	0	0	0	0	6	6	6	4	2	0	0	0	0	0	0	6	6	6	4	2	0	0	0	0	0
c11	3	3	3	0	0	0	0	0	1	2	3	3	3	0	0	0	0	0	0	1	2	3	0	0	0	0
c12	2	2	2	2	2	2	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0
c13	1	1	1	2	2	2	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	1	1	0
c14	4	4	6	2	2	2	2	2	2	2	2	4	4	0	0	2	2	4	4	4	4	4	0	0	0	0

So the end sequence is:

```
a.fire(); c.fire(); a.fire(); b.fire(); c.fire(); f.fire(); e.fire(); d.fire(); b.fire();
c.fire(2); e.fire(); h.fire(2); d.fire(); h.fire(); g.fire(); h.fire(2); e.fire(2);
g.fire(); i.fire(); h.fire(); b.fire(); c.fire(); e.fire(); i.fire(); c.fire(); d.fire();
j.fire(2); h.fire(3); e.fire(); c.fire(); i.fire(); c.fire(); e.fire(); d.fire();
e.fire(); g.fire(); h.fire(3); i.fire(); j.fire(); g.fire(); j.fire();
```

1.3

The initial number of tokens in the channel $c5$ is four. This is the least number of tokens that makes sure the channel never has negative tokens.

2

2.1

Code is submitted. `Assessment2_withBridge.xml` shows the model with the new BridgeBus class. `Assessment2_reducedBandwidth.xml` is the same but with a 290 bandwidth on the BridgeBus actor. `Assessment2_improvedMapping.xml` is the same but with an improved mapping and 250 bandwidth. Also the VerificationAnalysis actor has been modified to output the missed tasks on a different output but this is not important.

2.2

Lowering the bandwidth to 250 sometimes causes a task to miss the deadline which means the model is unschedulable. Anything between 250..300 is hard to tell whether the model is schedulable or not because we will have to model every possible jitter. The mapping given always uses the second bus for every task so therefore, the model is equivilent to the model with no bus. Because of this, it is impossible to reduce the bandwidth further than in the bus-less model.

2.3

The mapping can be improved with the Kernighan-Lin algorithm. The edge labels need to be normalized This algorithm can be applied recursively on all sixteen tasks. This is the output that was calculated for each processor.

p0	p1	p2	p3	p4	p5	p6	p7
14, 15	7, 8	12, 13	9, 10	1, 4	0, 3	2, 11	5, 6

However the nature of the Kernighan-Lin algorithm always splits the groups into equal sizes which is not a requirement for our purposes. So I can further move tasks around to optimise the mapping.

p0	p1	p2	p3	p4	p5	p6	p7
14, 15	7, 8, 9	12, 13	10	1, 4	0, 3	2, 11	5, 6

Using this improved mapping, it seems as though the model is still schedulable with the bandwidth reduced to 250. It is hard to tell whether the model is really schedulable but I ran the simulation 10 times and it no tasks were missed.

The Kernighan-Lin program used is submitted. Run the `AssessmentKernighanLin` class and input `taskmapping.txt`

2.4

The definition of schedulability in this assessment says a task's end-to-end latency is *always* less than its period for 30000ms. This is a problem because testing if something is always true is a semi-decidable problem - we can know for sure if the model is *not* schedulable but it is impossible to know if it is.