Aufgabe 1

Ein C Compiler speichert ein Array so, dass die beiden Zuweisungen für h und b aus folgendem C Code identisch sind:

```
1 int array[512][640][800];
2 int i,j,k;
3 int a,b;
4
5 h = array[i][j][k];
6 b = *(array + i*640*800 + j*800 + k);
```

Es sollen alle Elemente des Arrays mit folgendem Code aufsummiert werden.

```
1 int array[512][640][800];
2 int i,j,k;
3 int summe;
4
5 summe = 0;
6 for(k=0;k<800;k++)
7   for(j=0;j<640;j++)
8     for(i=0;i<512;i++)
9       summe += array[i][j][k];
```

Die Rechnerarchitektur hat einen voll assoziativen Datencache mit einer Blockgröße von 32 Bytes. Insgesamt ist der Cache 1 kByte groß. Nehmen Sie an, dass nur array Daten im Cache gespeichert sind und alle anderen Variablen in Registern gehalten werden. Ein Integer kann eine 32 Bit Zahl speichern. Nehmen Sie an, dass das array cache aligned ist, d.h. das der Speicherbereich des arrays genau am Anfang eines Cacheblocks beginnt.

a) Geben Sie die Cache Miss Rate für obigen Algorithmus an.

b) Nehmen Sie an, dass die Anzahl der Blöcke im Cache beliebig groß werden kann, die Blockgröße jedoch gleich bleibt. Wie groß kann die Cache Hit Rate maximal werden?

c) Nehmen Sie an, der Cache hat insgesamt 512 Blöcke mit einer Blockgröße von 1600*4 Bytes. Wie groß ist die Cache Hit Rate?

d) Ändern Sie den Algorithmus so ab, dass die Funktion gleich bleibt, die Cache Hit Rate jedoch maximal wird.

e) Schlagen Sie eine alternative Blockgröße bei gleicher Gesamtgröße des Caches vor, mit der Sie die Cache Hit Rate für den modifizierten Algorithmus aus d) maximieren. Wie groß wird dann die Cache Hit Rate?

Lösung

a) Die Cache Miss Rate beträgt 100%. Im Cache sind insgesamt 32 Blöcke mit jeweils 32 Bytes. In einen Block passen 8 Arrayelemente, da jeder Integerwert 4 Byte groß ist. Es gibt jedoch schon in der inneren Schleife 512 Speicherzugriffe im Abstand von 640*800*4 Bytes. Beim 33sten Lesezugriff müssen deshalb Daten aus dem Cache gegen neue Daten getauscht werden.

b) Jedes Element wird nur einmal gelesen. Deshalb kann ein Cache Hit nur innerhalb eines Blockes erfolgen. In einen Block passen 8 Integerwerte, davon ist der erste Zugriff in jedem Fall ein Cache Miss. Die maximal mögliche Cache Hit Rate ist deshalb 7/8 = 87,5%.

c) Die innere Schleife wird 512 mal durchlaufen und führt zu Speicherzugriffen im Abstand von 640*800*4 Bytes. Wenn die innere Schleife das zweite Mal durchlaufen wird, kommt es zu einem Cache Hit, da der Zugriff innerhalb des ersten Blocks erfolgt. Beim dritten Durchlauf kommt es allerdings zu einem Cache Miss, da der Zugriff auf Adresse 2*800*4 gerade außerhalb des Blocks ist, der beim ersten Speicherzugriff geladen wurde. Die Cache Hit Rate beträgt also 50%.

d) Für den Algorithmus ist die Reihenfolge der Schleifendurchläufe gleichgültig. Die innere und die äußere Schleife können deshalb getauscht werden.

```
1 int array[512][640][800];
2 int i,j,k;
3 int summe;
4
5 summe = 0;
6 for{i=0;i<512;i++)  /* i ist jetzt in der inneren Schleife */
7   for(j=0;j<640;j++)
8     for(k=0;k<800;k++)
9       summe += array[i][j][k];
```

Mit diesem modifizierten Algorithmus wird auf den Speicher sequentiell zugegriffen, d.h. die Folge der Adressen beim Zugriff auf den Speicher ist (0,4,8,12..). Mit diesem Zugriffsmuster wird die Cache Hit Rate maximal.

e) Die Cache Hit Rate wird bei dem modifizierten Algorithmus gemäß d) maximal wenn die Blockgröße maximal wird. Für diesen Algorithmus ist also ein Cache mit einem Block und einer Blockgröße von 1kByte optimal. Die Cache Hit Rate beträgt dann 1023/1024 = 99,9%.

Aufgabe 2

Für einen Algorithmus zur Positionsbestimmung müssen zwei Matrizen A und B multipliziert werden $C = A \cdot B$. Jedes Matrixelement ist eine 16 Bit Zahl. Die Matrixmultiplikation ist jedesmal notwendig wenn ein neuer Messwert von den Abstandssensoren vorliegt. Die Sensoren liefern Daten mit einer Rate von 10 MSamples/s. Die Werte der Elemente der Matrizen A und B ändern sich mit jedem neuen Sensorwert.

$$c_{ij} = \sum_{k=0}^{2} a_{ik} \cdot b_{kj}; \qquad A = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \tag{1}$$

Der Algorithmus zur Matrixmultiplikation ist in Gleichung (1) angegeben. Die beiden Matrizen A und B, sowie die Ergebnismatrix C befinden sich in einem Speicher mit 16 Bit Breite. Der Speicher kann in jedem Takt ein Datum lesen oder schreiben. Sie sollen eine Schaltung zur Matrixmultiplation entwerfen.

a) Berechnen Sie die notwendige Taktfrequenz mit dem der Speicher getaktet werden muss um die Daten für jede Berechnung aus dem Speicher auszulesen und das Ergebnis im Speicher abzulegen. Nehmen Sie also an, das Ihre Schaltung keine Matrixelemente zwischenspeichert.

b) Die erforderliche Taktfrequenz für den Speicher aus der vorherigen Aufgabenstellung ist zu hoch, d.h. Sie können keinen Speicher mit einer so hohen Taktfrequenz bauen. Nehmen Sie jetzt an, dass Sie beliebig viele Matrixelemente in ihrer Schaltung für die Berechnung zwischenspeichern können. Wie hoch ist dann die minimal mögliche Speicherfrequenz?

c) Um den Speichertakt zu reduzieren sollen in drei 16 Bit Registern r0, r1, r2 die drei Elemente einer Zeile der A Matrix zwischengespeichert werden. Das Timing der Eingangsdaten ist in Abbildung 1 dargestellt.

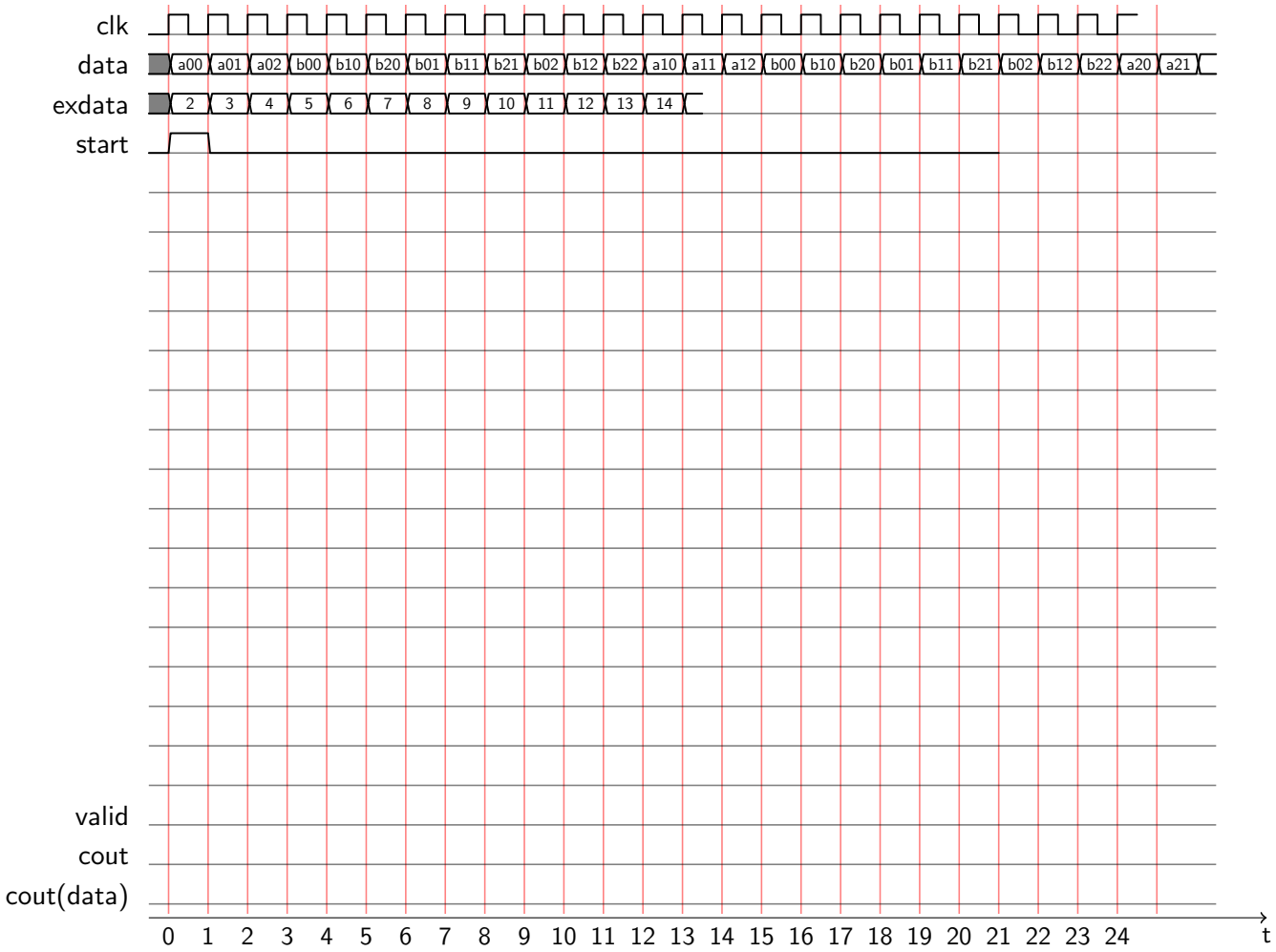

Abbildung 1: Timingdiagramm zur Matrixmultiplikation

Zusammen mit dem Element a_{00} ist das Signal "start" aktiv. Die ersten drei Daten sind die erste Zeile der A Matrix. Dann folgt die erste Spalte der B Matrix, dann die zweite Spalte der B Matrix und dann die dritte Spalte der B Matrix. Dann folgt die zweite Zeile der A Matrix. Zeichnen Sie einen Datenpfad für die Berechnung der Matrixmultiplikation und die dazugehörigen Steuersignale. Zeichnen Sie das Timing der Steuersignale in das Timingdiagramm ein. Zeichnen Sie die Werte von wichtigen Signalen im Datenpfad in das Diagramm ein. Nehmen Sie als Werte für die Matrixelemente die Werte aus der Zeile "exdata" an. Der Ausgang ihrer Schaltung ist ein valid und ein 16 Bit cout Signal. Valid soll aktiv werden, sobald ein berechnetes Element der C Matrix am Datenausgang vorliegt.

Geben Sie in Zeile "cout" an welches Matrixelement der Matrix C am Ausgang vorliegt, also beispielsweise "c01". Geben Sie in der Zeile "cout(data)" den zugehörigen Datenwert als Zahl an.

Lösung

a) Für jedes Element c_{ij} sind 6 Lesezugriffe für die Elemente der Matrizen A und B, sowie ein Schreibzugriff für das Abspeichern notwendig. Insgesamt gibt es neun Elemente in der C Matrix. Deshalb sind insgesamt 7 * 9 = 63 Zugriffe für eine Matrixmultiplikation notwendig. Bei 10 MSamples/s und somit 10 Mio. Matrixmultiplikationen pro Sekunde ist deshalb eine Taktfrequenz von 630 MHz für

den Speicher notwendig.

b) Es müssen mindestens die jeweils 9 Elemente aus den Matrizen A und B gelesen und die 9 Elemente für C abgespeichert werden. Es sind also 27 Zugriffe notwendig und es ergibt sich eine Taktfrequenz von 270 MHz.