Aufgabe 1

Ein Algorithmus soll zwei Videobilder vergleichen und ermitteln, ob sich ein Objekt zwischen den beiden Videobildern verschoben hat. Dazu soll der interessierende Bildausschnitt im ersten Bild mit einer Größe von 100 x 100 Pixeln mit Ausschnitten an anderen Positionen im zweiten Bild verglichen werden. Die Ausschnitte im zweiten Bild sind gegenüber dem Ausschnitt im ersten Bild um (v_x, v_y) verschoben. Der Vergleich soll durch Kreuzkorrelation der Bildpunkte erfolgen. Die Kreuzkorrelation R wird gemäß Gleichung 1 berechnet.

$$R(v_x, v_y) = \sum_{j=0}^{99} \sum_{i=0}^{99} g(i,j) \cdot g'(v_x + i, v_y + j) \tag{1}$$

In der Gleichung gibt g den Grauwert im ersten Bild und g' den Grauwerten im zweiten Bild an. Dieser Bildvergleich soll für insgesamt 10 mögliche Verschiebungen (v_x, v_y) berechnet werden. Die Bildrate beträgt 25 Bilder/s und für jedes Bild soll eine solche Berechnung durchgeführt werden.

a) Die Berechnung soll auf einer CPU durchgeführt werden. Schätzen Sie die Anzahl der erforderlichen Instruktionen pro Sekunde ab. Nehmen Sie eine Load/Store Architektur für die CPU an. Nehmen Sie an, dass Sie für die Berechnung der Speicheradresse des Grauwertes aus i und j 5 Instruktionen benötigen.

b) Schätzen Sie die erforderliche CPU Frequenz ab.

c) Eine alternative Hardwarearchitektur zur Berechnung der Kreuzkorrelationswerte für die 10 Verschiebungen soll mit 5 MHz Taktfrequenz laufen. Wie viele Multiplizierer benötigen Sie mindestens in dieser Schaltung?

d) Reicht ein Single Port Memory, das auch mit 5 MHz getaktet werden soll, aus um alle Daten der Berechnung aus c) zu speichern (Grauwerte, Verschiebungsvektoren, Ergebnisse der Kreuzkorrelation)?

e) Um den Bildalgorithmus zu verbessern soll entweder die Anzahl der Bildverschiebungsvektoren von 10 auf 20 werden oder die Ausschnittgröße von 100x100 auf 200x200 verdoppelt werden. Welche Alternative benötigt weniger Ressourcen?

Lösung

a) Zunächst wird der Algorithmus für die Berechnung der Kreuzkorrelation auf Instruktionsebene skizziert. Ich nehme an, dass die Variablen i und j in Registern gehalten werden.

```
1          reg f = 0;
2          j = 0
3  loop1:  i = 0
4  loop2:  gp = berechne_bildspeicher_adresse(i,j)
5          g'p = berechne_bildspeicher_adresse(v_x + i,vy + j)
6          load gp -> reg c
7          load g'p -> reg d
8          mult reg c * reg d -> reg e
9          add reg e -> reg f
10         inc i;
11         i < 100 ? goto loop2
12         inc j;
13         j < 100 ? goto loop1
```

```
14          store reg f -> Ergebnis Kreuzkorrelation
15          next vx,vy vektor
```

Die innere Schleife loop2 benötigt 16 Instruktionen. Diese Schleife wird 100 mal durchlaufen. Die äußere Schleife hat drei weitere zusätzliche Instruktionen und wird auch 100 mal durchlaufen. Außerhalb der äußeren Schleife gibt es dann nochmal 3 Instruktionen. Insgesamt werden also $Ni_{cc} = 100 \cdot (3 + 100 \cdot 16) + 3 = 160303$ Instruktionen für die Berechnung einer Kreuzkorrelation benötigt. Für die 10 Verschiebungen sind deshalb etwa 1.6 Mio. Instruktionen notwendig.

Die gesamte Instruktionsrate ergibt sich zu:

$$Ni_{tot} = 25\,Bilder/\mathrm{s} \cdot 1.6 \cdot 10^6\,Instr./Bild = 40 Mio.\,Instr./\mathrm{s} \tag{2}$$

b) Die erforderliche CPU Frequenz hängt von der Instruktionsrate und der Anzahl der Taktzyklen für die Ausführung einer Instruktion ab (CPI). Ein typischer CPI Wert ist 2, d.h. für die Ausführung einer Instruktion werden zwei Zyklen benötigt. Die Taktfrequenz der CPU ergibt sich demnach zu 80 MHz.

c) Die Anzahl der erforderlichen Multiplikationen beträgt:

$$Mul_{rate} = 25\,\mathrm{Bilder/s} \cdot 100 \cdot 100\,\mathrm{Mult./Versch.} \cdot 10\,\mathrm{Versch./Bild} = 2500000\,\mathrm{Mult./s} \tag{3}$$

Ein Multiplizierer reicht aus um diese Berechnung durchzuführen.

d) Für jede Multiplikation in der Kreuzkorrelation werden jeweils zwei Grauwerte benötigt, d.h. es sind 5 Mio. Speicherzugriffe pro Sekunde nur für das Lesen der Graphikdaten notwendig. Damit ist das Single-Port Memory schon zu 100 Prozent ausgelastet und es bleibt keine Zeit für den Zugriff auf die Verschiebungsvektoren und das Speichern der Ergebnisdaten.

e) Die Verdopplung der Anzahl der Vektoren geht linear in den Ressourcenverbrauch ein, während die Änderung des Ausschnitts quadratisch mit dem Ressourcenverbrauch zusammenhängt. Von den beiden Alternativen ist also die Erhöhung der Anzahl der Verschiebungsvektoren weniger aufwändig.

Aufgabe 2

Ein Modul eines Graphikprozessors soll entscheiden, ob ein Punkt P(x,y) innerhalb eines Rechtecks, gegeben durch die Eckpunkte R(x,y) und Q(x,y) liegt. In Abbildung 1 sind die Punkte dargestellt.

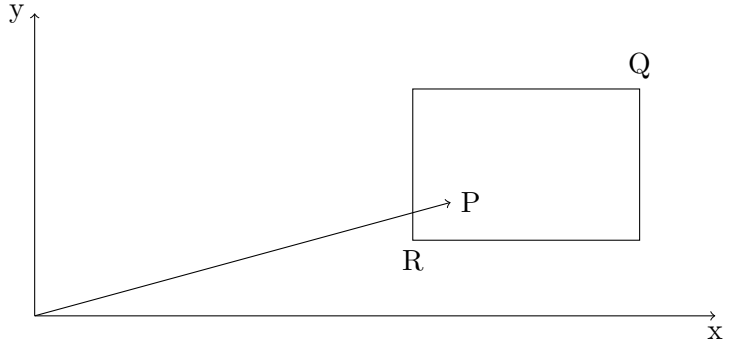

Abbildung 1: Rechteck

Die Bildrate beträgt 50 Bilder/s und es sollen insgesamt 10000 Punkte pro Bild untersucht werden. Das Bild hat eine Auflösung von 1980 x 1080 Punkten.

a) Wieviele und welche Vergleiche sind für die Untersuchung eines Punktes notwendig?

b) Wieviel Bits benötigen Sie für die Darstellung der Koordinaten?

c) Skizzieren Sie eine Architektur, die berechnen kann, ob ein Punkt in einem Rechteck liegt oder nicht. Die Architektur soll einen Punkt pro Takt untersuchen können. Alle Koordinaten von P, R und Q stehen gleichzeitig in jedem Takt zur Verfügung.

d) In einer alternativen Architektur kommen die Koordinatendaten des Punktes und des Rechtecks seriell in der Folge (x,rx,qx,y,ry,qy) in das System. In Abbildung 2 ist der zeitliche Ablauf dargestellt. Zeitgleich mit der x Koordinate des Punktes P ist das "start" Signal aktiv. Sobald das Ergebnis vorliegt, soll das "done" Signal aktiviert werden. Das Ergebnis soll an einem Ausgang "pr" angezeigt werden. Wenn der Punkt im Rechteck liegt, soll pr = "1" sein, ansonsten ist pr = "0". Entwerfen Sie eine Architektur mit der Sie die Berechnung sequentiell durchführen können. Teilen Sie die Architektur in einen Datenpfad und einen Kontrollpfad auf. Stellen Sie nur den Datenpfad und die notwendigen Kontrollsignale dar. Erläutern Sie die Funktion der Kontrollsignale.

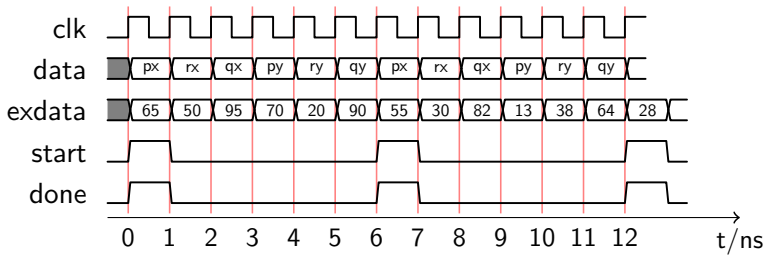

Abbildung 2: Timingdiagramm mit seriellen Eingangsdaten

e) Zeichnen Sie ein Timingdiagramm für Ihre Architektur in dem der zeitliche Ablauf der Steuersignale, die Eingangsdaten, wichtige Zwischenwerte und das Endergebnis der Berechnung für die Beispieldaten "exdata" aus Abbildung 2 erkennbar ist.

f) Entwerfen Sie einen Zustandsautomaten, der als Ausgang die Steuersignale für Ihren Datenpfad bereitstellt.

Lösung

a) Um zu entscheiden ob der Punkt innerhalb des Rechtecks liegt sind die folgenden vier Vergleiche notwendig

a) $x > rx$?

b) $x < qx$?

c) $y > ry$?

d) $y < qy$?

b) Für die Darstellung sind 11 Bit notwendig ($2^{11} = 2048$).

c) In Abbildung 3 ist eine Architektur dargestellt, die in jedem Takt überprüfen kann, ob ein Punkt in einem Rechteck liegt oder nicht. Dazu werden in jedem Takt die vier Vergleiche durchgeführt.

d) In Abbildung 4 ist eine Datenpfadarchitektur dargestellt.

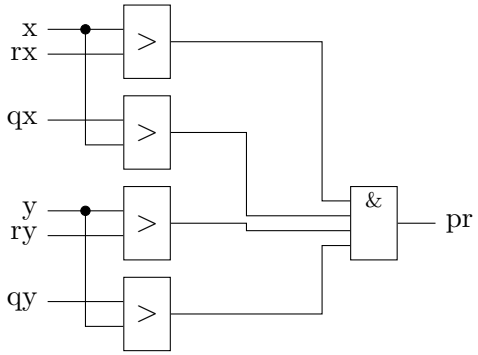

Abbildung 3: Architektur für die Berechnung ob ein Punkt in einem Rechteck liegt

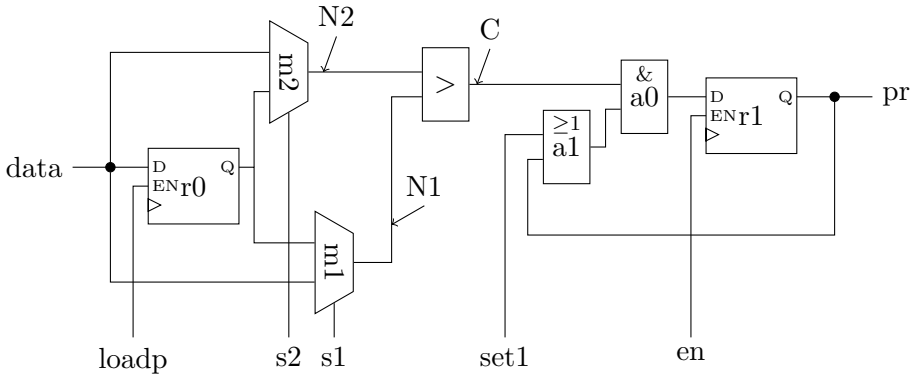

Abbildung 4: Serielle Architektur zur Entscheidung ob der Punkt innerhalb des Rechtecks liegt

Wenn das Signal "loadp" aktiv ist, wird der x oder der y Wert des P Punktes in das Register r0 geladen. Mit den Multiplexern m1 und m2 kann eingestellt werden, ob der Wert aus dem Register r0 oder das ankommende Datensignal an die Eingänge des Multiplexers geschaltet werden. Wenn s1 aktiv ist, wird der Inhalt des Registers r0 an den Komparator geschaltet. Wenn s2 aktiv ist, werden die Eingangsdaten "data" an den Komparator geschaltet. Die UND Operation für die vier Vergleiche soll seriell erfolgen. Wenn "set1" aktiv ist, kann das erste Komparatorergebnis in das Register r1 geladen werden. Wenn "set1" nicht aktiv ist, wird das neue Komparatorergebnis mit dem Inhalt des Registers r1 "verundet".

e) In Abbildung 5 ist der zeitliche Ablauf der Kontrollsignale dargestellt.

Der erste Punkt liegt innerhalb des Rechtecks und deshalb ist pr aktiv, wenn "done" aktiv ist. Der zweite Punkt liegt nicht im Rechteck und deshalb ist "pr" am Ende der Berechnung nicht aktiv.

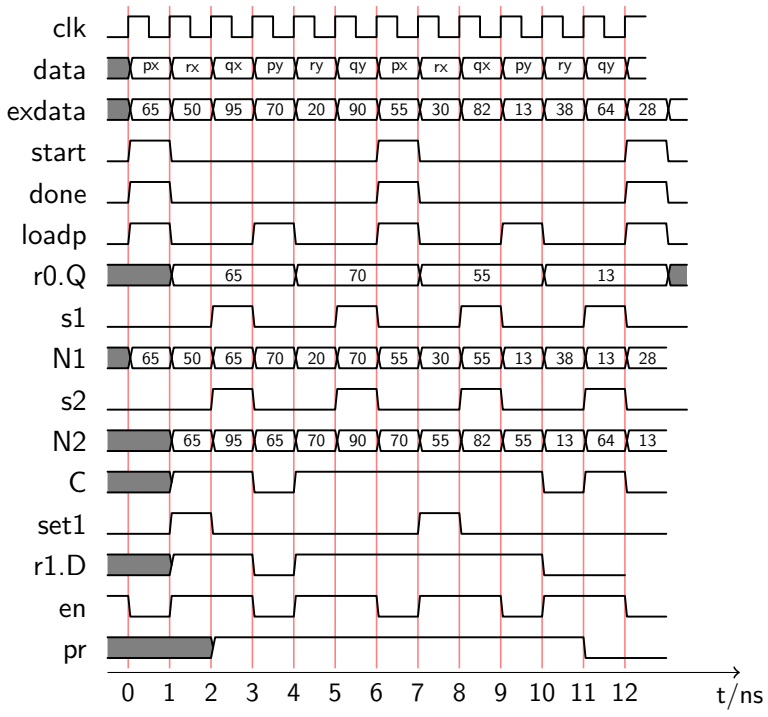

Abbildung 5: Timingdiagramm mit Kontrollsignalen