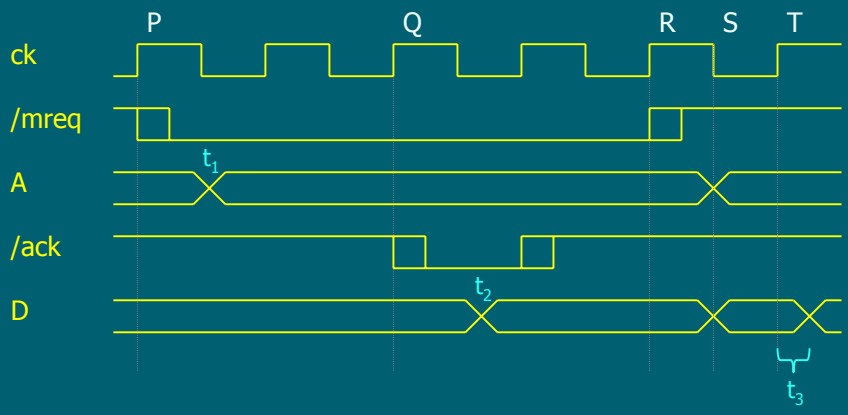
15.2 Busprotokolle, Zustandsdiagramme

15.2.1 Busprotokolle

Kommunikation zwischen Kontrolleinheit der CPU und den Kontrolleinheiten der Speicher erfolgt über ein festes Busprotokoll mit Hilfe der Signale /mreq, /mw, /ack

Lesezugriff hier in 3 Stufen

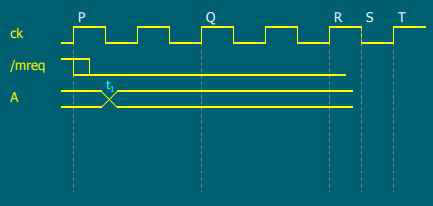


a) Lesen

Lesezugriff hier in 3 Stufen (ff)

1. Lesezugriff von CPU wird initiiert durch Aktivieren von /mreq an steigender Flanke P von ck (/mw inaktiv).
   Nach Zeit t_1 (nach P) garantiert CPU gültige Adressen auf A.

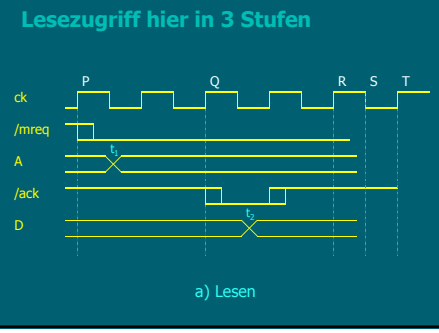
Lesezugriff hier in 3 Stufen



a) Lesen

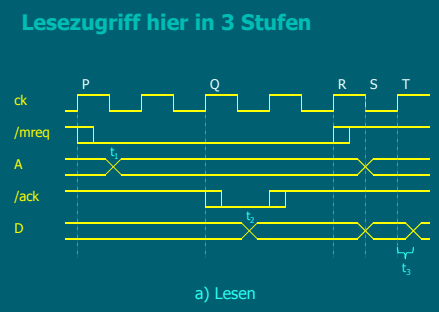
Lesezugriff hier in 3 Stufen (ff)

2. Speicher aktiviert Acknowledge-Signal /ack für genau einen Takt an steigender Flanke Q. Nach Verzögerungszeit t_2 nach Q garantiert Speicher korrekte Daten auf D.

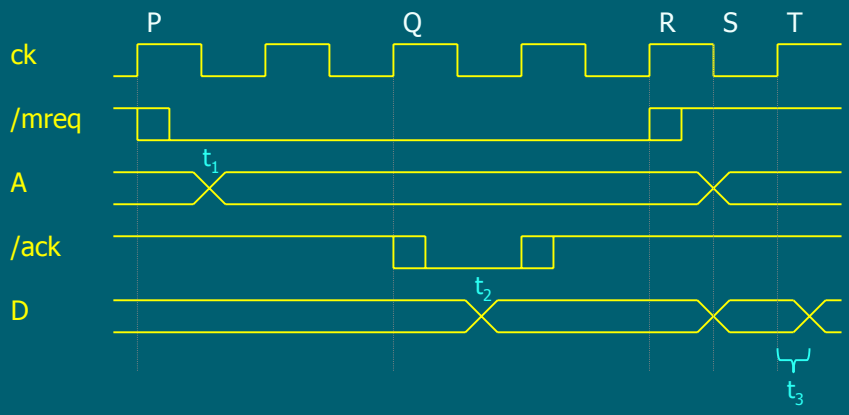


Lesezugriff hier in 3 Stufen (ff)

3. CPU deaktiviert /mreq an steigender Flanke R. Speicher garantiert gültige Daten bis fallende Flanke S nach R und garantiert Disablen der Treiber auf D nach Zeit t_3 nach nachfolgender steigender Flanke T

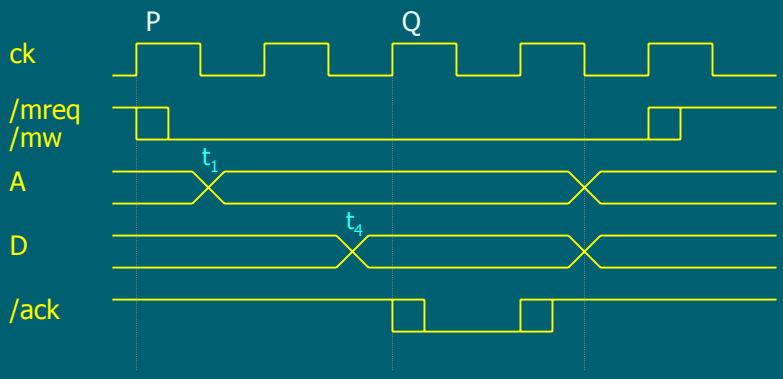


Lesezugriff hier in 3 Stufen



a) Lesen

Schreibzugriff in 2 Stufen

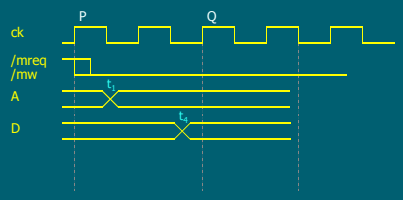


b) Schreiben

Schreibzugriff in 2 Stufen (ff)

1. CPU aktiviert /mreq und /mw an steigender Flanke P von ck.
   Nach t_1 (nach P) gültige Adressen auf A garantiert, nach t_4 auch gültige Daten auf D

Schreibzugriff in 2 Stufen

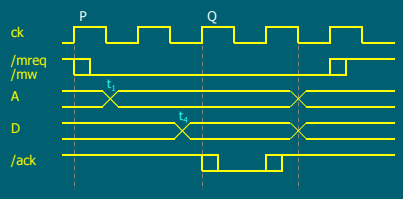


b) Schreiben

Schreibzugriff in 2 Stufen (ff)

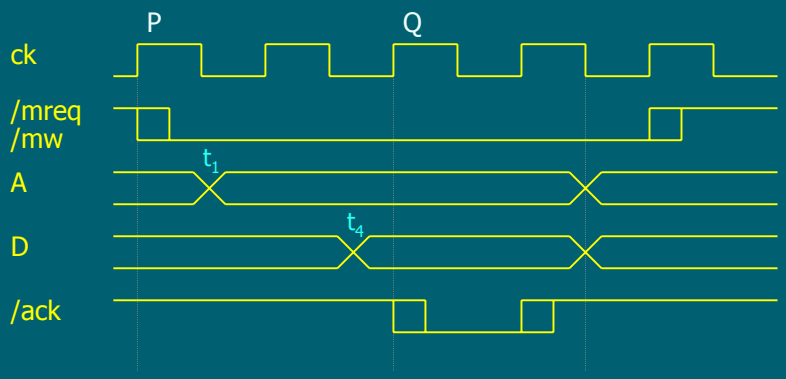
2. Speicher aktiviert /ack an Q für genau einen Takt.
   CPU garantiert stabile Adressen und Daten noch bis zur fallenden Flanke nach Deaktivieren von /ack.
   /mreq und /mw bleiben aktiv mindestens bis steigende Flanke nach Q.

Schreibzugriff in 2 Stufen



b) Schreiben

Schreibzugriff in 2 Stufen



b) Schreiben

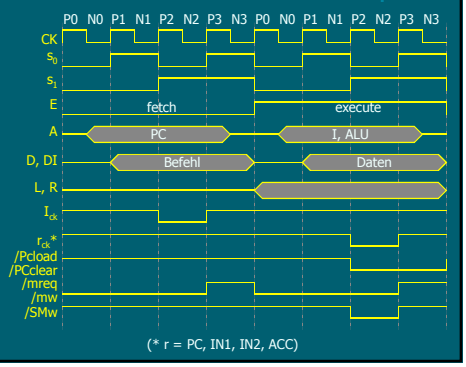
Idealisiertes Timing mit Wait-Zyklen

Timing von /mreq und /mw wird leicht abgeändert:

Bisher:

Aktivieren bei P0,

Deaktivieren bei P3.



Idealisiertes Timing mit Wait-Zyklen

Jetzt:

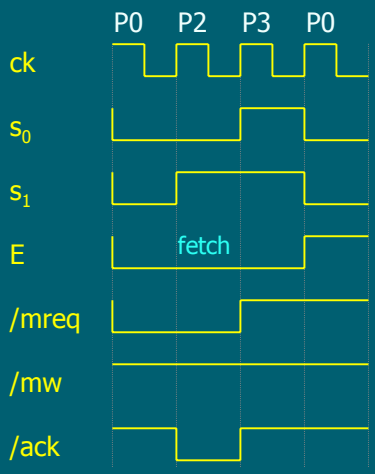
Fetch und Compute memory bleiben wie bisher;
bei Load und Store (außer LOADI, MOVE) wird
Aktivierungszeitpunkt auf P1 verschoben,
um Zeit für Adressrechnung zu haben.

Zunächst SRAM (schneller Speicher)

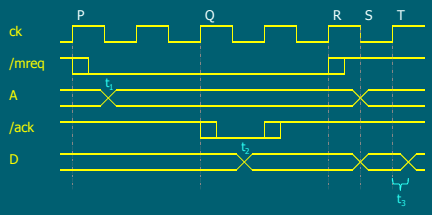
Kontrolle von SRAM wird so entworfen, dass
/ack genau einen Takt nach /mreq aktiv wird.
CPU-Kontrolle soll dann keine Wait-Zyklen erzeugen.

(Vgl. folgende Abbildungen)

Fetch für SRAM

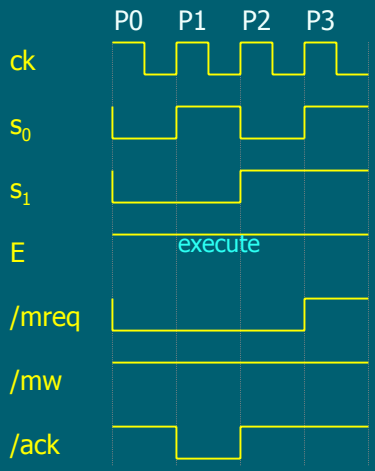


Lesezugriff hier in 3 Stufen

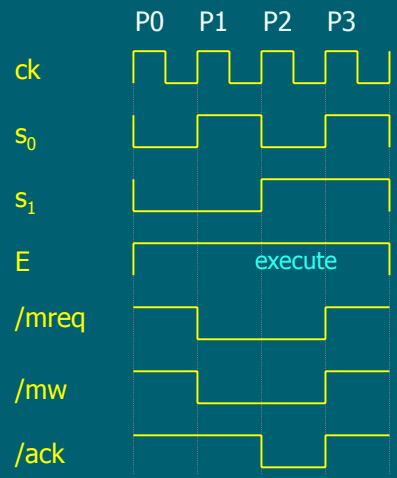


a) Lesen

Compute memory für SRAM



Store für SRAM

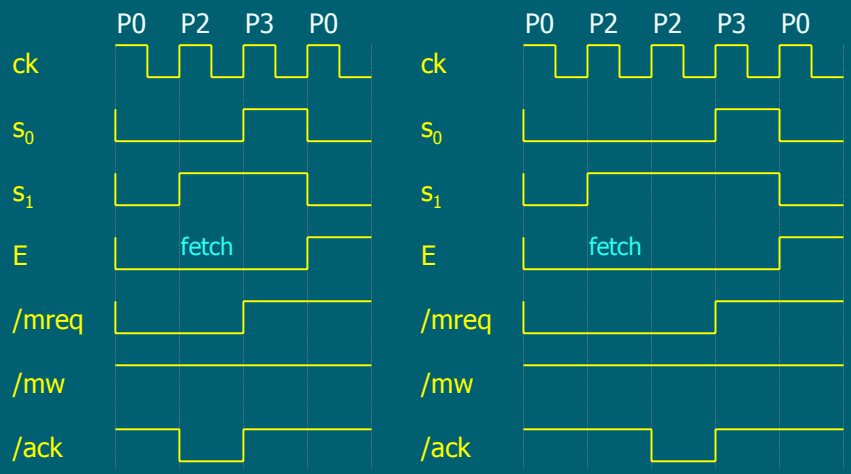


Langsamere Speicher: Wait-Zyklen

Wenn /ack noch nicht aktiv ist, wird der aktuelle Takt wiederholt

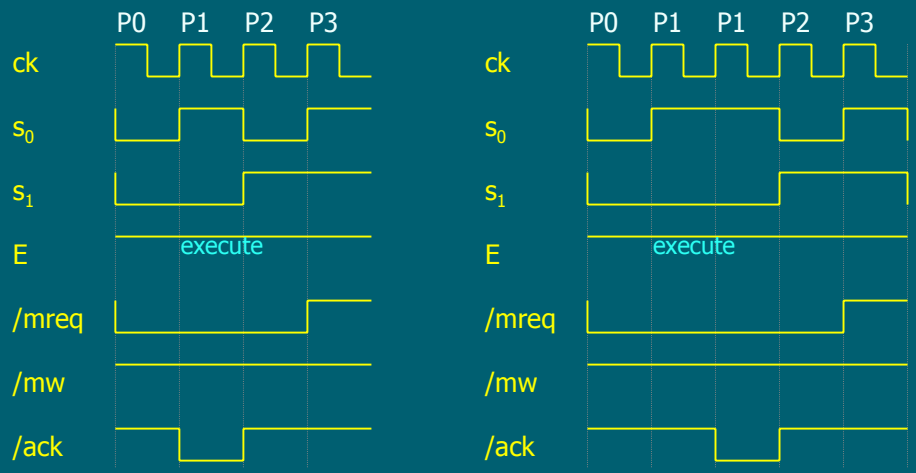
(Vgl. folgende Abbildungen)

Wait-Zyklen für Fetch



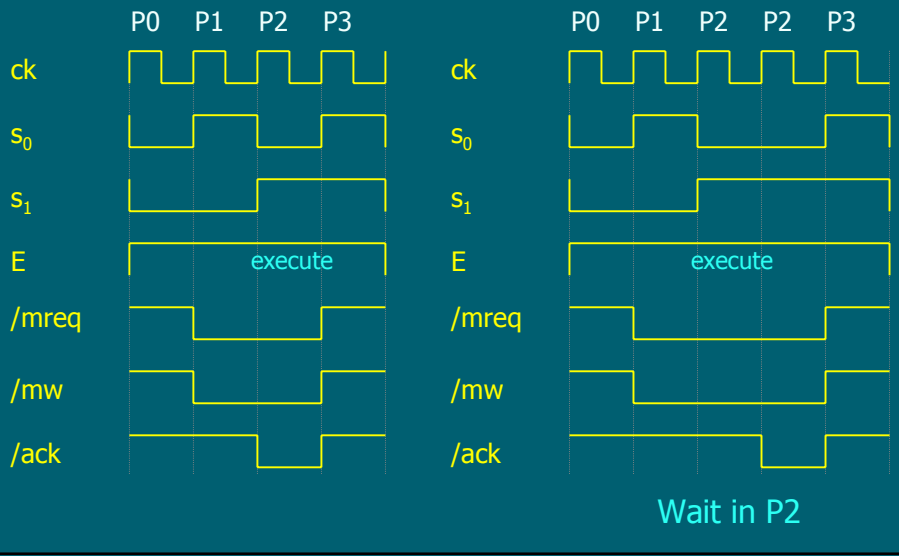
Wait in P2

Wait-Zyklen für Compute memory



Wait in P1

Wait-Zyklen für Store



Wait-Zyklen (ff)

Wenn /ack noch nicht aktiv ist, so wird die aktuelle Uhrzeit wiederholt (wait!!), solange bis /ack aktiv, d.h.
bei Fetch wird P2 wiederholt,
bei Compute memory P1,
bei Load/Store P2.

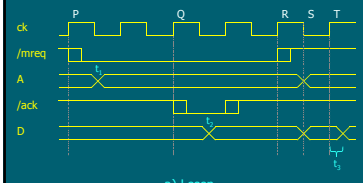
Exaktes Timing mit neuem Protokoll für SRAM

Adressen sind stabil bei (Bezugspunkt ist Flanke bei der \mreq aktiviert wird):

Fetch: (10.5, 18.3) + (2, 8)

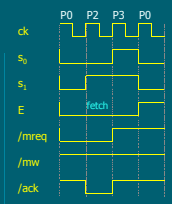
/PCAdoe enable-Zeit PCAd

Lesezugriff hier in 3 Stufen



a) Lesen

Fetch für SRAM



Neue Analyse

1. Neue Analyse von Compute Memory:
   IAd wird enabled bei P0 von Execute, genauer (vgl. vorige Abbildung)
   IAdœ1 aktiv zur Zeit $t_2' = \tau + (8.0, 12.0)$
   P3 (fetch) Bezugspunkt! — Delay PAL
   IAdœ aktiv bei $t_2 = t_2' + (2.5, 6.3) = \tau + (10.5, 18.3)$
   AND-Gatter

Zeitangaben zu Treibern

	Treiber 74F244	min	max
t_{PZL}	Enable-Zeiten	2.0	8.0
t_{PZH}	Enable-Zeiten	2.0	6.7
t_{PLZ}	Disable-Zeiten	2.0	7.0
t_{PHZ}	Disable-Zeiten	2.0	7.0
t_{PLH}	Umschaltverzögerung bei /OE = 0	2.5	6.2
t_{PHL}	Umschaltverzögerung bei /OE = 0	2.5	6.5

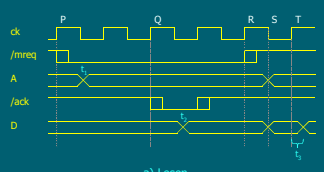
Exaktes Timing mit neuem Protokoll für SRAM (ff)

Adressen sind stabil bei:

Compute: (10.5, 18.3) + (2, 8)

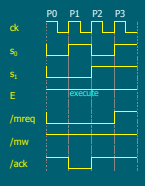
/IAdoe enable-Zeit IAd

Lesezugriff hier in 3 Stufen



a) Lesen

Compute memory für SRAM



Exaktes Timing mit neuem Protokoll für SRA

Adressen sind stabil bei:

LOADINj, STOREINj:

$(8, 12) + (2, 8) + 46 + 6.5 - \tau$

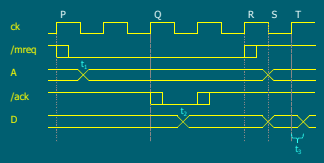
IRdoe — enable-Zeit IRd — ALU — ALUAd

Adressrechnung geht in P0 los, /mreq kommt erst in P1

Zeitangaben zu Treibern

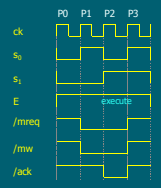
	Treiber 74F244	min	max
t_{PZL}	Enable-Zeiten	2.0	8.0
t_{PZH}	Enable-Zeiten	2.0	6.7
t_{PLZ}	Disable-Zeiten	2.0	7.0
t_{PHZ}	Disable-Zeiten	2.0	7.0
t_{PLH}	Umschaltverzögerung bei /OE = 0	2.5	6.2
t_{PHL}	Umschaltverzögerung bei /OE = 0	2.5	6.5

Lesezugriff hier in 3 Stufen



a) Lesen

Store für SRAM



Exaktes Timing mit neuem Protokoll für SRAM (ff)

Wegen $\tau \geq 52.0$ gilt:

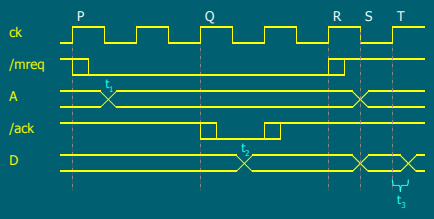
$(8, 12) + (2, 8) + 46 + 6.5 - \tau \leq 72.5 - 52 = 20.5$

Insgesamt gilt:

Adressen sind stabil bei

$t_1 = 26.3$

Lesezugriff hier in 3 Stufen



a) Lesen

Exaktes Timing mit neuem Protokoll für SRAM: Lesezugriff

Zu t_2 :

Gültige Daten liegen auf dem D-Bus zur Zeit

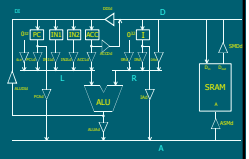
26.3 + 6.5 + 45 + 6.5 = 84.3

t_1 ASMd SM SMDd,

falls wie bisher früh genug enabled

Daten auf D

→ Daten auf D spätestens zur Zeit



Treiber-verzögerung

$t_8 = \max(\max(t_5) + 6.5, \max(t_7))$

schon enabled, wenn Daten gültig **

nicht „*rechtzeitig*" enabled

= max(3/2 τ + 84.0, 2τ + 20.0)

** genauer: Enablen geschieht > 1.5 = 8.0 – 6.5 ns bevor Daten gültig werden

Exaktes Timing mit neuem Protokoll für SRAM: Lesezugriff (ff)

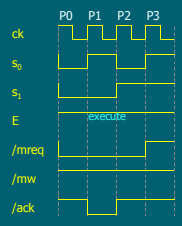
Dies ist

$t_2 = 84.3 - \tau \leq 84.3 - 52 = 32.3$

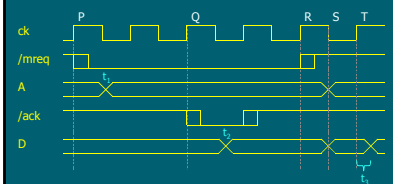
nach P1, bei der /ack für

Compute memory aktiviert

Compute memory für SRAM



Lesezugriff hier in 3 Stufen



a) Lesen

Lesezugriff hier in 3 Stufen (ff)

2. Speicher aktiviert Acknowledge-Signal /ack für genau einen Takt an steigender Flanke Q. Nach Verzögerungszeit t_2 nach Q garantiert Speicher korrekte Daten auf D.

Exaktes Timing mit neuem Protokoll für SRAM: Lesezugriff (ff)

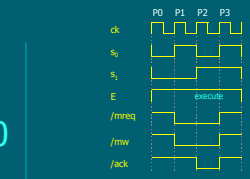
Zu t_3 :

Bei P3 wird /mreq deaktiviert,

Frühestens 2 Takte später anderer Treiber auf D-Bus, also muss gelten

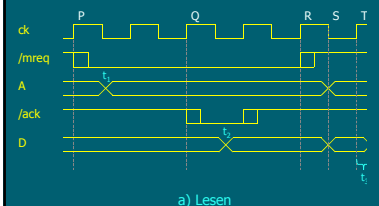
$t_3 \leq \tau$, problemlos möglich bei $\tau \geq 52.0$

Store für SRAM



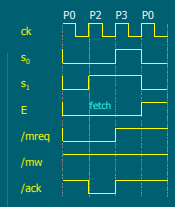
P0 N0 P1 N1 P2 N2 P3 N3 P0 N0 P1 N1 P2 N2 P3 N3

Lesezugriff hier in 3 Stufen



a) Lesen

Fetch für SRAM



/P

Lesezugriff hier in 3 Stufen (ff)

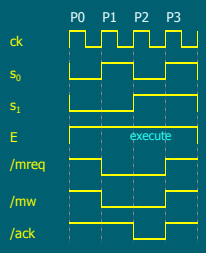
3. CPU deaktiviert /mreq an steigender Flanke R. Speicher garantiert gültige Daten bis fallende Flanke S nach R und garantiert Disablen der Treiber auf D nach Zeit t_3 nach nachfolgender steigender Flanke T

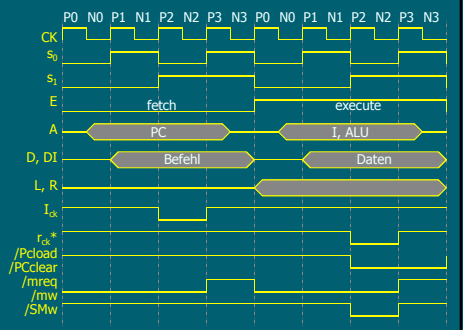
Exaktes Timing mit neuem Protokoll für SRAM: Schreibzugriff

Nach Lesezugriff jetzt noch Schreibzugriff:

/SMw und /ack sind zeitgleich bei P2 aktiv

Store für SRAM





(* r = PC, IN1, IN2, ACC)

Exaktes Timing mit neuem Protokoll für SRAM: Schreibzugriff (ff)

Überprüfe der Reihe nach Bedingungen zum Funktionieren des SRAMS und Einhalten des Protokolls

RAM CY7C191-45

Symbol	Bezeichnung	min	max
t_{acc}	Lesezugriffszeit	3	45
t_{SAW}	Setup-Zeit von A bis W	0	
t_{SAEW}	Setup-Zeit von A bis Ende W	35	
t_{HWA}	Hold-Zeit von A nach W	0	
w	Schreibpulsweite	22	
t_{SDEW}	Setup-Zeit von D bis Ende W	15	
t_{HWD}	Hold-Zeit von D nach W	0	

Schreibzugriff in 2 Stufen (ff)

1. CPU aktiviert /mreq und /mw an steigender Flanke P von ck.
   Nach t_1 (nach P) gültige Adressen auf A garantiert, nach t_4 auch gültige Daten auf D

Exaktes Timing mit neuem Protokoll für SRAM: Schreibzugriff (ff)

Adressen sind nach $t_1 = 26.3$ auf A garantiert,

26.3 + 6.5 = 32.8 nach P1 liegen sie an SM an,

(t_1, ASMd)

wegen $\tau \geq 52.0$ ist dies vor P2;

t_{SAW} und t_{SAEW} werden eingehalten

RAM CY7C191-45

Symbol	Bezeichnung	min	max
t_{acc}	Lesezugriffszeit	3	45
t_{SAW}	Setup-Zeit von A bis W	0	
t_{SAEW}	Setup-Zeit von A bis Ende W	35	
t_{HWA}	Hold-Zeit von A nach W	0	
w	Schreibpulsweite	22	
t_{SDEW}	Setup-Zeit von D bis Ende W	15	
t_{HWD}	Hold-Zeit von D nach W	0	

Schreibzugriff in 2 Stufen (ff)

1. CPU aktiviert /mreq und /mw an steigender Flanke P von ck.
   Nach t_1 (nach P) gültige Adressen auf A garantiert, nach t_4 auch gültige Daten auf D

Exaktes Timing mit neuem Protokoll für SRAM: Schreibzugriff (ff)

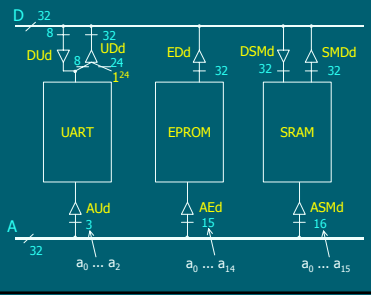
Ebenso wird die Schreibpulsweite eingehalten,

wir kümmern uns nun um D:

RAM CY7C191-45

Symbol	Bezeichnung	min	max
t_{acc}	Lesezugriffszeit	3	45
t_{SAW}	Setup-Zeit von A bis W	0	
t_{SAEW}	Setup-Zeit von A bis Ende W	35	
t_{HWA}	Hold-Zeit von A nach W	0	
w	Schreibpulsweite	22	
t_{SDEW}	Setup-Zeit von D bis Ende W	15	
t_{HWD}	Hold-Zeit von D nach W	0	

Datenpfade (graphisch)

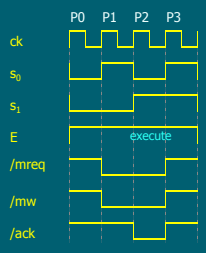


Exaktes Timing mit neuem Protokoll für SRAM: Schreibzugriff (ff)

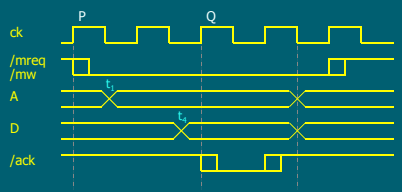
stabile Daten hinter DSMd sollen an P2

vorhanden sein,

Store für SRAM



Schreibzugriff in 2 Stufen



b) Schreiben

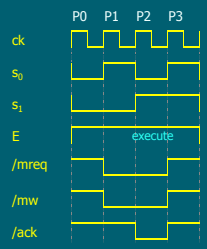
Exaktes Timing mit neuem Protokoll für SRAM: Schreibzugriff (ff)

d.h. einen Takt nach P1 (Aktivierung von /mreq, /mw),

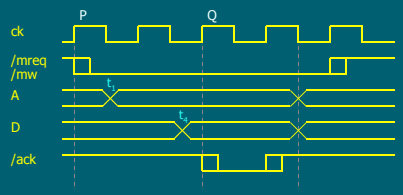
also $t_4 + 6.5 \leq \tau$

muss gelten.

Store für SRAM



Schreibzugriff in 2 Stufen



b) Schreiben

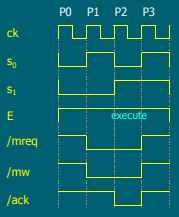
Exaktes Timing mit neuem Protokoll für SRAM: Schreibzugriff (ff)

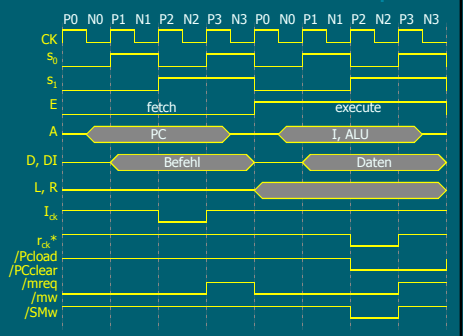
Wegen $\tau \geq 52.0$ genügt es $t_4 \leq 45.5$ einzuhalten.

Dies geschieht, da ACCDd zur Zeit P1 enabled wird,

Daten sind also zur Zeit $t^+ + (2, 8) = (10, 20)$ stabil.

Store für SRAM



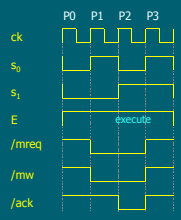


(* r = PC, IN1, IN2, ACC)

Exaktes Timing mit neuem Protokoll für SRAM: Schreibzugriff (ff)

Damit ist insbesondere die Zeit t_{SDEW} für das SRAM eingehalten, es bleibt der Nachweis der Hold-Zeiten und der Konsistenz mit Schritt 2 des Protokolls beim Schreibzugriff

Store für SRAM



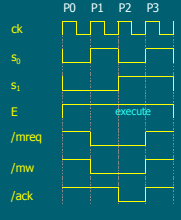
RAM CY7C191-45

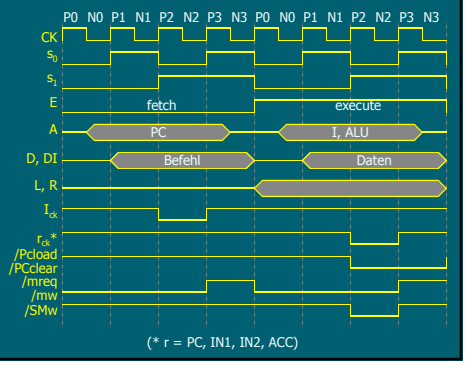
Symbol	Bezeichnung	min	max
t_{acc}	Lesezugriffszeit	3	45
t_{SAW}	Setup-Zeit von A bis W	0	
t_{SAEW}	Setup-Zeit von A bis Ende W	35	
t_{HWA}	Hold-Zeit von A nach W	0	
w	Schreibpulsweite	22	
t_{SDEW}	Setup-Zeit von D bis Ende W	15	
t_{HWD}	Hold-Zeit von D nach W	0	

Exaktes Timing mit neuem Protokoll für SRAM: Schreibzugriff (ff)

Zur Einhaltung des Protokolls müssen Adressen und Daten mindestens bis N3 stabil bleiben (dies ist stärker als die vom Speicher geforderten Hold-Zeiten)

Store für SRAM





(* r = PC, IN1, IN2, ACC)

Exaktes Timing mit neuem Protokoll für SRAM (ff)

Wir zeigen im folgenden für alle Speicherbausteine:

Das Protokoll wird eingehalten mit

$t_1 = 26.3$

$t_2 = 32.3$

$t_3 \leq 52$

$t_4 \leq 45.5$

15.2.2 Zustandsdiagramme

Aufgaben:

- Erzeuge Uhrzeit in Abhängigkeit von Befehlen und /ack-Signal
- Erzeuge für jeden Speicher eine Kontrolleinheit

→ Zustandsdiagramme