

Kapitel 5

Timing:

1. Physikalische Eigenschaften
2. Timing wichtiger Komponenten
3. **Exaktes Timing von ReTI**

Albert-Ludwigs-Universität Freiburg

Prof. Dr. Christoph Scholl

Institut für Informatik
WS 2015/16

Es gilt:

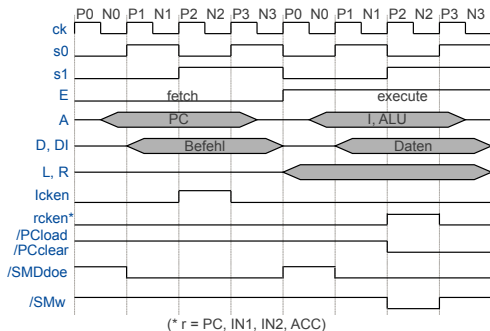
- Bei hinreichend langsamem Takt funktioniert der Rechner.

Frage:

- Wie schnell kann man den Rechner takten?
Wie lange muss ein Takt mind. sein?

→ Ersetze idealisiertes Timing durch exakte Timinganalyse

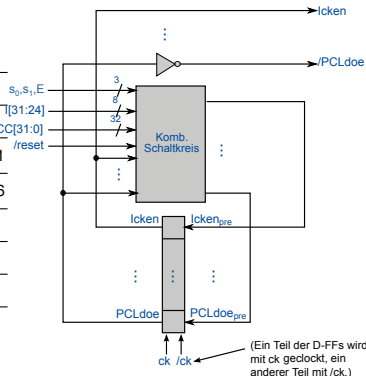
→ Gesucht:
Untere Grenze für Zykluszeit t_c



- 1 Einhalten von Setup- und Hold-Zeiten der Kontrolllogik
 - 2 Vermeidung von Bus-Contention
 - 3 PC-Inkrementierung
 - 4 Compute-Befehle:
OE: Compute Memory
 - 5 *Fetch, Load, Store, Jump*
- Wir werden uns auf **Compute-Befehle** beschränken.

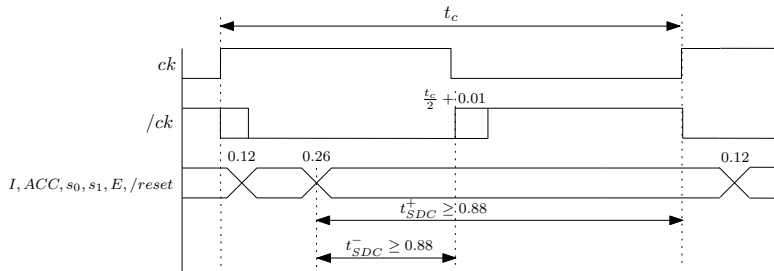
Timing der Kontrolllogik (1/3)

Symbol	Bezeichnung	t_{\min}	t_{\max}
$\tau_{p,ah}^+$	Verzögerungszeit ck bis Q , active high	0.12	0.26
$\tau_{p,al}^+$	Verzögerungszeit ck bis Q , active low	0.12	0.41
$\tau_{p,ah}^-$	Verzögerungszeit ck bis Q (von $/ck$ angesteuert, active high)	$t_c/2 + 0.13$	$t_c/2 + 0.41$
$\tau_{p,al}^-$	Verzögerungszeit ck bis Q (von $/ck$ angesteuert, active high)	$t_c/2 + 0.13$	$t_c/2 + 0.56$
t_{SDC}^+	Setupzeit von D bis ck	0.88	
t_{SDC}^-	Setupzeit von D bis $/ck$	0.88	
t_{HCD}^+	Holdzeit von D nach ck	0.06	
t_{HCD}^-	Holdzeit von D nach $/ck$	0.06	



- **Setupzeit** der Dateneingänge bis ck ist $t_{SDC}^+ \geq 0.88$, **Setupzeit** der Dateneingänge bis $/ck$ ist $t_{SDC}^- \geq 0.88$.
- Dateneingänge ($s_0, s_1, E, I, ACC, reset$) müssen rechtzeitig bereit sein.
- Alle Dateneingänge sind Ausgangssignale von FFs, die mit ck getaktet werden.

Timing der Kontrolllogik (2/3)



- Wähle eine beliebige steigende Taktflanke P_i als zeitlichen Bezugspunkt.
- Die Dateneingänge sind also bereit zur Zeit $\tau_{PCQ} = [0.12, 0.26]$ (Verzögerung eines D-FF).
- Die nächste steigende Taktflanke von $/ck$ ist bei $\frac{t_c}{2} + [0.01, 0.15]$, die nächste steigende Taktflanke von ck bei t_c .

$$\Rightarrow \frac{t_c}{2} + \underbrace{0.01}_{\text{Inverterverzögerung}} \geq \underbrace{0.88}_{t_{SDC}^-} + \underbrace{0.26}_{\tau_{PCQ} \text{ für FFs}}, \text{ d.h. } t_c \geq 2.26.$$

$$\Rightarrow t_c \geq \underbrace{0.88}_{t_{SDC}^+} + \underbrace{0.26}_{\tau_{PCQ} \text{ für FFs}} = 1.14$$

- Hold-Zeiten sind unkritisch:

- FFs, die mit ck getaktet sind:

- $t_{HDC}^+ \geq 0.06$ und Eingangsdaten werden mindestens noch 0.12 ns nach steigender Flanke von ck gehalten (Verzögerung D-FF).

- FFs, die mit $/ck$ getaktet sind:

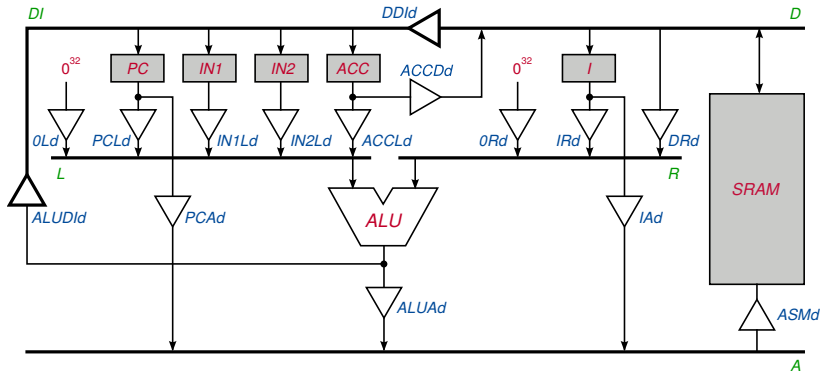
- $t_{HDC}^- \geq 0.06$ und Eingangsdaten werden sowieso noch einen halben Takt gehalten nach steigender Flanke von $/ck$ gehalten (+ D-FF-Verzögerung).

Constraints

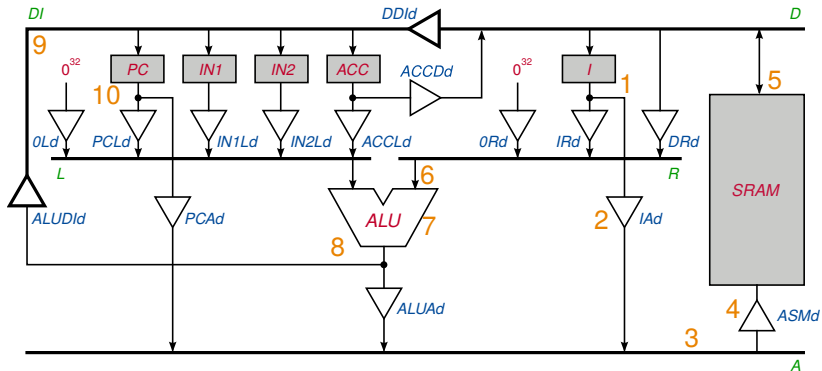
- $t_c \geq 2.26$

Compute-Befehle

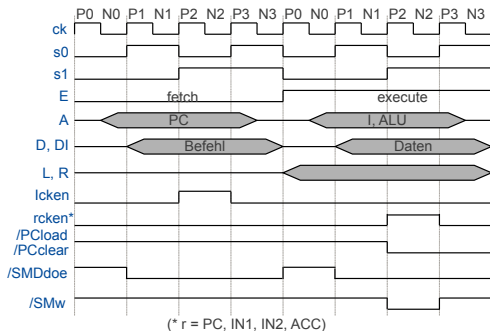
- Am zeitkritischsten ist **Compute memory**!



$$[r] := [r] + [M(\langle i \rangle)]$$



- Beginn der Analyse bei **P3** von Fetch als zeitlicher Bezugspunkt.
- Bei **P3** von Fetch wird der Befehl ins Instruktionsregister übernommen.



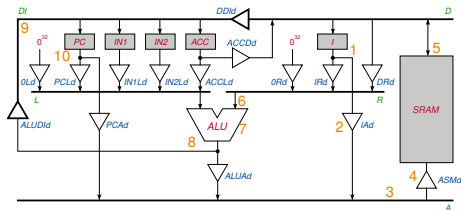
I-Ausgänge (1/2)

I-Ausgänge gültig bei

$$\tau_1 = \underbrace{[0.12, 0.26]}_{\tau_{PCQ} \text{ von Register } I}$$

- $0^8 I_{23} \dots I_0$ wird über Treiber IAd auf Adressbus gegeben.
- 0^8 ist eine Konstante und steht daher ebenfalls zu τ_1 bereit.

D-FF	Bezeichnung	t_{\min}	t_{\max}
t_{SDC}	Setupzeit von D bis ck	0.08	
t_{HCD}	Holdzeit von D nach ck	0.14	
τ_{PCQ}	Verzögerungszeit von ck bis Q	0.12	0.26
τ_{PDQ}	Verzögerungszeit von D bis Q	0.10	0.21



Constraints

- $\tau_1 = [0.12, 0.26]$

- $t_c \geq 2.26$

I-Ausgänge (2/2)

Ad enabled bei *N0* von Execute, d.h.

/IA doe aktiv zur Zeit

$$\tau_2' = t_c + \tau_{p,al}^-$$

$$= t_c + \frac{t_c}{2} + [0.13, 0.56]$$

$$= \frac{3}{2}t_c + [0.13, 0.56]$$

/IA_{doe} wird verteilt auf 32

Tristate-Treiber von Ad , also

Treiberbaum mit Tiefe 2.

/IAdoe kommt bei IAd an zur Zeit

$$\tau_2 = \tau_2' + 2 \cdot [0.02, 0.11]$$

$$= \frac{3}{2}t_c + [0.17, 0.78]$$

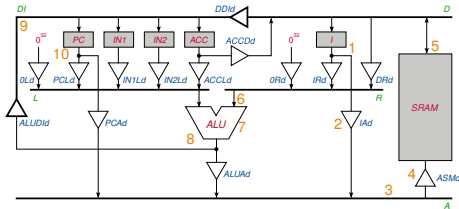
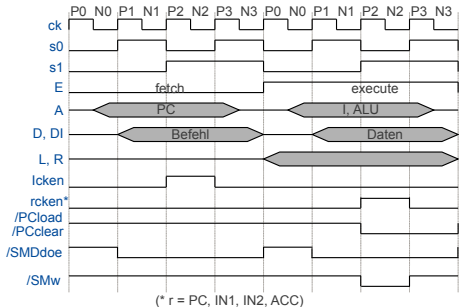
/ schon gültig vor Aktivierung von IAd
bei Punkt 2, falls

$$\max(\tau_1) \leq \min(\tau_2) \Leftrightarrow$$

$$0.26 \leq \frac{3}{2}t_c + 0.17 \Leftrightarrow$$

$$\frac{3}{2}t_c \geq 0.09 \Leftarrow$$

$$t_c \geq 0.06$$



Constraints

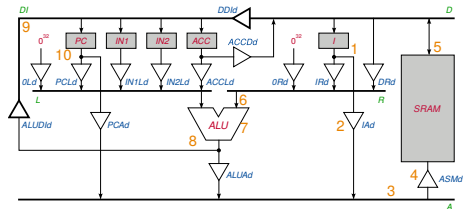
- $\tau_1 = [0.12, 0.26]$
- $\tau_2 = t_c + \tau_{p,al}^- = \frac{3}{2}t_c + [0.17, 0.78]$
- $t_c \geq 2.26$
- $t_c \geq 0.06$

Gültiges A (1/2)

→ A gültig zur Zeit

$$\begin{aligned}\tau_3 &= \tau_2 + \underbrace{[0.03, 0.11]}_{\text{Enable Zeit Treiber}} \\ &= \frac{3}{2} t_c + [0.20, 0.89]\end{aligned}$$

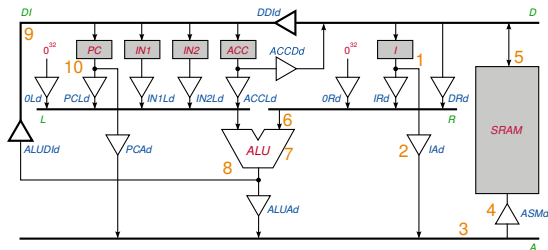
	Tristate-Treiber	min	max
τ_{PZL}	Enable-Zeiten	0.03	0.10
τ_{PZH}	Enable-Zeiten	0.03	0.11
τ_{PLZ}	Disable-Zeiten	0.03	0.11
τ_{PHZ}	Disable-Zeiten	0.03	0.10
τ_{PLH}	Umschaltverzögerung bei $/OE = 0$	0.02	0.07
τ_{PHL}	Umschaltverzögerung bei $/OE = 0$	0.03	0.10



Constraints

- $\tau_1 = [0.12, 0.26]$
 - $\tau_2 = t_c + \tau_{p,al}^- = \frac{3}{2}t_c + [0.17, 0.78]$
 - $\tau_3 = \tau_2 + [0.03, 0.11] = \frac{3}{2}t_c + [0.20, 0.89]$
-
- $t_c \geq 2.26$
 - $t_c \geq 0.06$

Gültiges A (2/2)



■ *ASMd* immer enabled

→ nur Treiber-Verzögerung berücksichtigt

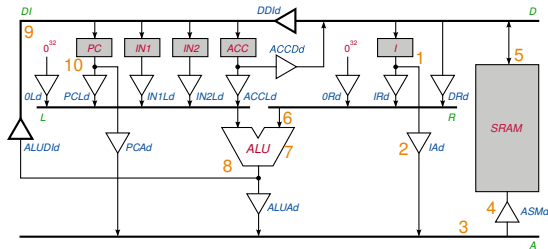
→ *A* an *SM* bei

$$\tau_4 = \tau_3 + [0.02, 0.10] = \frac{3}{2}t_c + [0.22, 0.99]$$

Constraints

- $\tau_1 = [0.12, 0.26]$
 - $\tau_2 = t_c + \tau_{p,al}^- = \frac{3}{2} t_c + [0.17, 0.78]$
 - $\tau_3 = \tau_2 + [0.03, 0.11] = \frac{3}{2} t_c + [0.20, 0.89]$
 - $\tau_4 = \tau_3 + [0.02, 0.10] = \frac{3}{2} t_c + [0.22, 0.99]$
- $t_c \geq 2.26$

Daten am Speicherausgang (1/3)



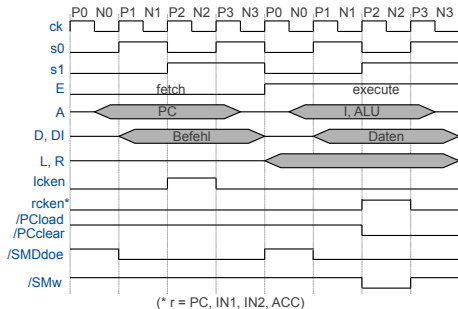
- Lesezugriffszeit von *SRAM*: [0.0, 12.0] (siehe Daten von CY7C1079DV33)

→ Gültige Daten am Speicherausgang bei

$$\begin{aligned}
 \tau_5 &= \tau_4 + [0.0, 12.0] \\
 &= \frac{3}{2} t_c + [0.22, 0.99] + [0.0, 12.0] \\
 &= \frac{3}{2} t_c + [0.22, 12.99]
 \end{aligned}$$

- Das ist aber nur korrekt, wenn der interne Ausgangstreiber durch */SMDdoe* rechtzeitig enabled ist!

Daten am Speicherausgang (2/3)



SRAM CY7C1079DV33			
Symbol	Bezeichnung	t_{\min}	t_{\max}
t_{acc}	Lesezugriffszeit		12.0
t_{OED}	Zeit von $/SMDdoe = 0$ bis D		7.0
...			

- $/SMDdoe$ aktiviert zur Zeit $\tau' = 2 \cdot t_c + \tau_{p,al}^+ = 2 \cdot t_c + [0.12, 0.41]$.
 - Daten am Speicherausgang aufgrund Treiber-Enable gültig zur Zeit $\tau'' = \tau' + [0.0, 7.0] = 2 \cdot t_c + [0.12, 7.41]$.
- ⇒ Daten am Speicherausgang gültig spätestens zur Zeit $t''' = \max(\max(\tau_5), \max(\tau''))$.

Daten am Speicherausgang (3/3)

- Daten am Speicherausgang gültig zur Zeit $t''' = \max(\max(\tau_5), \max(\tau''))$.
- Bedingung für $\max(\tau_5) \geq \max(\tau'')$:

$$\frac{3}{2}t_c + 12.99 \geq 2 \cdot t_c + 7.41 \Leftrightarrow$$

$$\frac{1}{2}t_c \leq 5.58 \Leftrightarrow$$

$$t_c \leq 11.16$$

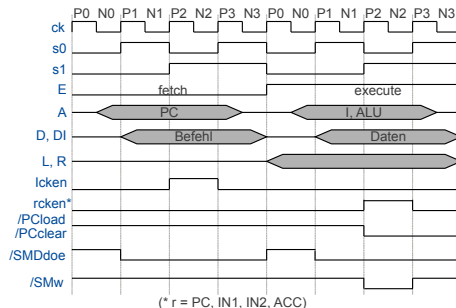
- Wir nehmen ab jetzt an, dass die minimale Taktperiode $t_c \leq 11.16$ und rechnen mit τ_5 weiter.
- (Es gilt auf jeden Fall $\min(\tau_5) (= \frac{3}{2}t_c + 0.22) < \min(\tau'') (= 2 \cdot t_c + 0.12)$)
- Sollte sich später ergeben, dass die minimale Taktperiode $t_c > 11.16$, dann müssten wir die Rechnung nochmals korrigieren.

Constraints

- $\tau_1 = [0.12, 0.26]$
- $\tau_2 = t_c + \tau_{p,al}^- = \frac{3}{2}t_c + [0.17, 0.78]$
- $\tau_3 = \tau_2 + [0.03, 0.11] = \frac{3}{2}t_c + [0.20, 0.89]$
- $\tau_4 = \tau_3 + [0.02, 0.10] = \frac{3}{2}t_c + [0.22, 0.99]$
- $\tau_5 = \tau_4 + [0.0, 12.0] = \frac{3}{2}t_c + [0.22, 12.99]$
- $t_c \geq 2.26$
- $t_c \geq 0.06$
- $t_c \leq 11.16$

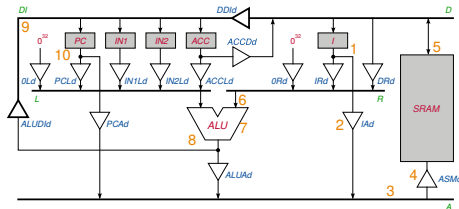
Daten auf *R*

DRd enabled bei *P0* von
Execute, also einen Takt vor
Ausgangstreiber von *SM*
→ Enable nicht kritisch
→ Daten auf *R* spätestens bei



$$\tau_6 = \tau_5 + [0.02, 0.10] \text{ (Treiber-Verzögerung)}$$

$$= \frac{3}{2} t_c + [0.24, 13.09]$$



Constraints

- $\tau_1 = [0.12, 0.26]$
- $\tau_2 = t_c + \tau_{p,al}^- = \frac{3}{2} t_c + [0.17, 0.78]$
- $\tau_3 = \tau_2 + [0.03, 0.11] = \frac{3}{2} t_c + [0.20, 0.89]$
- $\tau_4 = \tau_3 + [0.02, 0.10] = \frac{3}{2} t_c + [0.22, 0.99]$
- $\tau_5 = \tau_4 + [0.0, 12.0] = \frac{3}{2} t_c + [0.22, 12.99]$
- $\tau_6 = \tau_5 + [0.02, 0.10] = \frac{3}{2} t_c + [0.24, 13.09]$
- $t_c \geq 2.26$
- $t_c \geq 0.06$
- $t_c \leq 11.16$

- Registerausgänge $r \in \{PC, ACC, IN1, IN2\}$ schon seit letzter Execute-Phase gültig.

→ nicht kritisch

- Treiber rLd enabled bei $P0$ von Execute, d.h. wie auch bei DRd ist Zeit zum Enablen unkritisch im Vergleich zu τ_6 .

- $f[2 : 0], c_{in}$ werden durch den kombinatorischen Schaltkreis der Kontrolllogik aus l_{31}, \dots, l_{24} berechnet.
 - I -Ausgänge aber schon gültig bei $\tau_1 = [0.12, 0.26]$.
 - Verzögerungszeit des kombinatorischen Schaltkreises $< t_{SDC}^+ = 0.88$
 - $f[2 : 0], c_{in}$ gültig spätestens bei $t_7 = 0.26 + 0.88 = 1.14$.
- völlig unkritisch verglichen mit $\max(\tau_6) = \frac{3}{2}t_c + 13.09$

Constraints

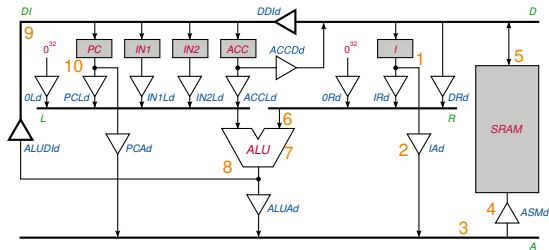
- $\tau_1 = [0.12, 0.26]$
- $\tau_2 = t_c + \tau_{p,al}^- = \frac{3}{2} t_c + [0.17, 0.78]$
- $\tau_3 = \tau_2 + [0.03, 0.11] = \frac{3}{2} t_c + [0.20, 0.89]$
- $\tau_4 = \tau_3 + [0.02, 0.10] = \frac{3}{2} t_c + [0.22, 0.99]$
- $\tau_5 = \tau_4 + [0.0, 12.0] = \frac{3}{2} t_c + [0.22, 12.99]$
- $\tau_6 = \tau_5 + [0.02, 0.10] = \frac{3}{2} t_c + [0.24, 13.09]$
- $t_7 = 1.14$
- $t_c \geq 2.26$
- $t_c \geq 0.06$
- $t_c \leq 11.16$

Voraussetzungen für die exakte Timinganalyse von *Compute memory*

- *ALU*
- Analyse der ALU (32-Bit mit Conditional Sum) unter folgender Annahme:
 - Funktionsselect-Signale liegen 0.28 ns vor den Daten an (unkritisch, da $t_7 + 0.28 = 1.14 + 0.28 = 1.42 < \min(\tau_6) = \frac{3}{2}t_c + 0.24$).
 - Resultatsausgänge gültig 3.25 ns, nachdem die Daten anliegen.

Symbol	Bezeichnung	t^{\min}	t^{\max}
t_{select}		0.28	
t_{ALU}	Verzögerungszeit von a , b bzw. c_{in} bis Ausgang		3.25

ALU-Ausgänge



■ Spätestens gültig bei

$$\begin{aligned}
 t_8 &= \max(\tau_6) + \underbrace{3.25}_{\text{Delay ALU}} = \frac{3}{2}t_c + 13.09 + 3.25 \\
 &= \frac{3}{2}t_c + 16.34
 \end{aligned}$$

Constraints

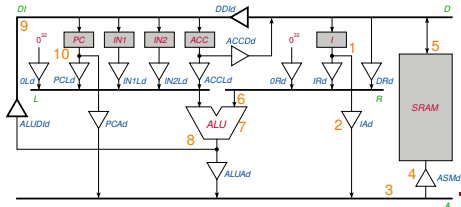
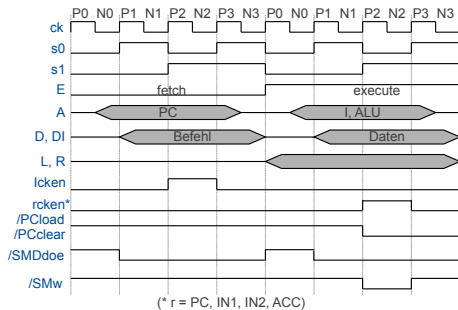
- $\tau_1 = [0.12, 0.26]$
- $\tau_2 = t_c + \tau_{p,al}^- = \frac{3}{2} t_c + [0.17, 0.78]$
- $\tau_3 = \tau_2 + [0.03, 0.11] = \frac{3}{2} t_c + [0.20, 0.89]$
- $\tau_4 = \tau_3 + [0.02, 0.10] = \frac{3}{2} t_c + [0.22, 0.99]$
- $\tau_5 = \tau_4 + [0.0, 12.0] = \frac{3}{2} t_c + [0.22, 12.99]$
- $\tau_6 = \tau_5 + [0.02, 0.10] = \frac{3}{2} t_c + [0.24, 13.09]$
- $t_7 = 1.14$
- $t_8 = \max(\tau_6) + 3.25 = \frac{3}{2} t_c + 16.34$
- $t_c \geq 2.26$
- $t_c \geq 0.06$
- $t_c \leq 11.16$

- $/ALUDldoe$ wird wie $/SMDdoe$ aktiviert bei $P1$ von Execute
- Daten kommen an $ALUDld$ später an als an als Daten am internen $SRAM$ -Treiber
- Enable-Zeit von $ALUDld$ jedoch kürzer als bei $SRAM$
- Mit $t_c \leq 11.16$ ist auf jeden Fall auch für $ALUDld$ gewährleistet, dass Treiber enabled, wenn Daten kommen.

→ Berücksichtige nur Treiberverzögerung.

→ Gültig spätestens bei

$$\begin{aligned}
 t_9 &= t_8 + 0.10 \\
 &= \frac{3}{2} t_c + 16.34 + 0.10 \\
 &= \frac{3}{2} t_c + 16.44
 \end{aligned}$$



Constraints

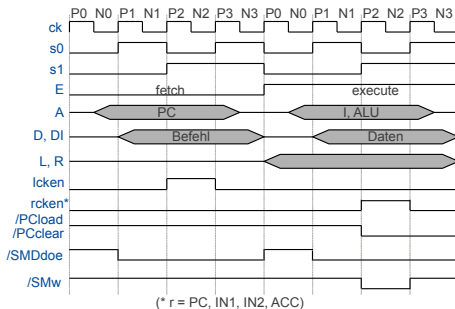
- $\tau_1 = [0.12, 0.26]$
- $\tau_2 = t_c + \tau_{p,al}^- = \frac{3}{2} t_c + [0.17, 0.78]$
- $\tau_3 = \tau_2 + [0.03, 0.11] = \frac{3}{2} t_c + [0.20, 0.89]$
- $\tau_4 = \tau_3 + [0.02, 0.10] = \frac{3}{2} t_c + [0.22, 0.99]$
- $\tau_5 = \tau_4 + [0.0, 12.0] = \frac{3}{2} t_c + [0.22, 12.99]$
- $\tau_6 = \tau_5 + [0.02, 0.10] = \frac{3}{2} t_c + [0.24, 13.09]$
- $t_7 = 1.14$
- $t_8 = \max(\tau_6) + 3.25 = \frac{3}{2} t_c + 16.34$
- $t_9 = t_8 + 0.10 = \frac{3}{2} t_c + 16.44$
- $t_c \geq 2.26$
- $t_c \geq 0.06$
- $t_c \leq 11.16$

Datenübernahme in Register r

- Clocksignale bei $P3$ von Execute

→ steigende Flanke bei $\tau_{10} = 4t_c$

- Minimale Taktperiode wird aus Setup-Zeit von r berechnet (Setup-Zeit am größten, wenn $r = PC$)



Constraints

- $\tau_1 = [0.12, 0.26]$
- $\tau_2 = t_c + \tau_{p,al}^- = \frac{3}{2} t_c + [0.17, 0.78]$
- $\tau_3 = \tau_2 + [0.03, 0.11] = \frac{3}{2} t_c + [0.20, 0.89]$
- $\tau_4 = \tau_3 + [0.02, 0.10] = \frac{3}{2} t_c + [0.22, 0.99]$
- $\tau_5 = \tau_4 + [0.0, 12.0] = \frac{3}{2} t_c + [0.22, 12.99]$
- $\tau_6 = \tau_5 + [0.02, 0.10] = \frac{3}{2} t_c + [0.24, 13.09]$
- $t_7 = 1.14$
- $t_8 = \max(\tau_6) + 3.25 = \frac{3}{2} t_c + 16.34$
- $t_9 = t_8 + 0.10 = \frac{3}{2} t_c + 16.44$
- $\tau_{10} = 4 \cdot t_c$
- $t_c \geq 2.26$
- $t_c \geq 0.06$
- $t_c \leq 11.16$

- Aus einer Analyse des Zählers in einer Implementierung gemäß Kapitel 4.1 (aber mit zusätzlichem Clock-Enable für das Register!) ergeben sich folgende Zeiten:

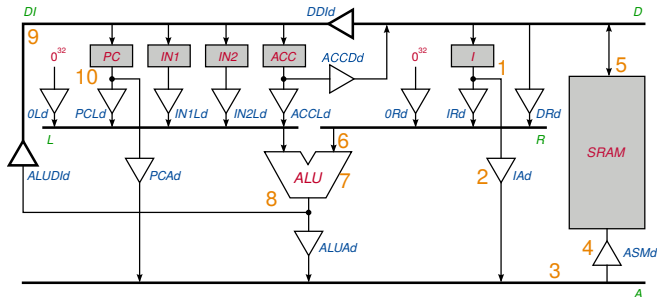
Symbol	Bezeichnung	t^{\min}	t^{\max}
t_{SDC}	Setup-Zeit von D vor ck	0.53	
t_{HDC}	Hold-Zeit von D nach ck	0.05	
t_{SLC}	Setup-Zeit von $/L$ vor ck	0.76	
t_{HLC}	Hold-Zeit von $/L$ nach ck	0.02	
t_{SEC}	Setup-Zeit von $PCcken$ vor ck	0.46	
t_{HEC}	Hold-Zeit von $PCcken$ nach ck	0.08	
...

Setup-Zeit von Zähler

- Setup-Zeit: $t_{SDC} = 0.53 \text{ ns}$ (siehe Aufbau Zähler)

→ Bedingung:

- $t_9 + 0.53 \leq \min(\tau_{10})$
 $\Leftrightarrow \frac{3}{2}t_c + 16.97 \leq 4t_c$
 $\Leftrightarrow \frac{5}{2}t_c \geq 16.97$
 $\Leftrightarrow t_c \geq 6.78$

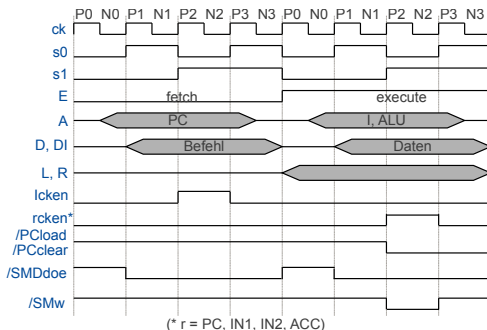


Constraints

- $\tau_1 = [0.12, 0.26]$
- $\tau_2 = t_c + \tau_{p,al}^- = \frac{3}{2} t_c + [0.17, 0.78]$
- $\tau_3 = \tau_2 + [0.03, 0.11] = \frac{3}{2} t_c + [0.20, 0.89]$
- $\tau_4 = \tau_3 + [0.02, 0.10] = \frac{3}{2} t_c + [0.22, 0.99]$
- $\tau_5 = \tau_4 + [0.0, 12.0] = \frac{3}{2} t_c + [0.22, 12.99]$
- $\tau_6 = \tau_5 + [0.02, 0.10] = \frac{3}{2} t_c + [0.24, 13.09]$
- $t_7 = 1.14$
- $t_8 = \max(\tau_6) + 3.25 = \frac{3}{2} t_c + 16.34$
- $t_9 = t_8 + 0.10 = \frac{3}{2} t_c + 16.44$
- $\tau_{10} = 4 \cdot t_c$
- $t_c \geq 2.26$
- $t_c \geq 0.06$
- $t_c \leq 11.16$
- $t_c \geq 6.78$

Es bleiben zu beachten:

- Hold-Zeit t_{HCD}
- Maximal bei $r \in \{ACC, IN1, IN2\}$, $t_{HCD} = 0.11$
 - **Unproblematisch**, da alle Treiber noch mindestens $\frac{1}{2}$ Takt nach rck enabled sind.

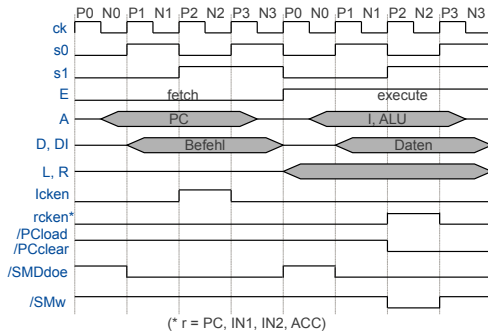


Setup- und Hold-Zeiten von *rcken*

- *rcken* aktiv bei *P2* von Execute, inaktiv bei *P3* von Execute, d.h. aktiv von:
 - $\tau_{11} = 3t_c + \tau_{p,ah}^+ = 3t_c + [0.12, 0.26]$ bis
 - $\tau_{12} = 4t_c + \tau_{p,ah}^+ = 4t_c + [0.12, 0.26]$
- Setup-Zeit:
 - Für alle $r \in \{PC, ACC, IN1, IN2\}$: $t_{SEC} = 0.46$
 - ⇒ $\max(\tau_{11}) + 0.46 \leq 4t_c$, d.h. $3t_c + 0.26 + 0.46 \leq 4t_c$ bzw.
 - ⇒ $t_c \geq 0.72$ (unkritisch im Vergleich zu bisherigen Constraints)
- Hold-Zeit:
 - Für alle $r \in \{PC, ACC, IN1, IN2\}$: $t_{HEC} = 0.08$
 - ⇒ $\min(\tau_{12}) \geq 4t_c + 0.08$, d.h. $0.12 \geq 0.08$
- (Analog auch für alle anderen Takte.)

Setup- und Hold-Zeiten / *PCload* beim Zähler

- Setup /L bis *ck*: $t_{SLC} = 0.76$
- Hold /L nach *ck*: $t_{HLC} = 0.02$
- */PCload* (benötigt, wenn **neue** Werte in Zähler kommen) aktiv bei *P2* von Execute, inaktiv bei *P0* von Fetch



Bedingungen für Setup- und Hold-Zeiten / *PCload* beim Zähler

- Setup: Am kritischsten, wenn /*PCload* seinen Wert bei der vorangegangenen Taktflanke geändert hat, d.h. bei P3 von execute bzw. P1 von fetch.

$$\max(\tau_{p,al}^+) + 0.76 \leq t_c \Leftrightarrow$$

$$0.41 + 0.76 \leq t_c \Leftrightarrow$$

$$t_c \geq 1.18$$

- Hold: Am kritischsten, wenn sich /*PCload* ändert, d.h. bei P2 von execute und P0 von fetch.

$$\min(\tau_{p,al}^+) \geq 0.02 \Leftrightarrow$$

$$0.12 \geq 0.02$$

→ Beide unkritisch im Vergleich zu bisherigen Constraints.

Fazit: Zykluszeit und Befehlsrate

- Vorläufiges Ergebnis: Falls sich durch andere Befehle keine schärferen Bedingungen an die Zykluszeit ergeben, dann lautet sie:

- $t_c \geq 6.78 \text{ ns}$

- Taktfrequenz:

- $\nu = \frac{1}{6.78} \cdot 10^9 \text{ Hz} = 147.4 \text{ MHz}$

- 8 Takte pro Befehl \rightarrow 18.4 Millionen Befehle pro Sekunde, d.h. Befehlsrate von **18.4 MIPS** (= Million Instructions per Second)

■ Beschleunigung

- Schnellere Komponenten, z.B. ALU
- Schnellere Adressberechnung (Treiber schon bei P0 öffnen)
- Evtl. Überdenken des kompletten Schemas des idealisierten Timings (z.B.: Verkürzung des Fetch-Zyklus um 1 Takt)
- schnellerer Speicher, Speicherhierarchie mit „Caches“
- Pipelining
- ...

Anmerkungen zur Timing–Analyse

- Eine „echte“ Timing–Analyse müsste noch Leitungslaufzeiten auf dem Chip berücksichtigen.
 - Dazu muss dann aber schon das Layout des Chips bekannt sein, um die Leitungslängen und -kapazitäten zu berechnen.
 - Leitungslaufzeiten waren früher bei einem Aufbau mit diskreten Bausteinen vernachlässigbar, sind es bei den heutigen Technologien aber nicht mehr.
- ⇒ Exakte Timing-Analysen sind heute daher kaum ohne maschinelle Unterstützung durchführbar.
- Synthesetools sind in der Lage, durch Optimierung der Treiberstärken von Grundgattern (verschiedene Versionen in der Bibliothek!) Laufzeiten zu minimieren.
 - Wird das SRAM nicht auf dem Chip integriert (d.h. stattdessen ein kommerzielles externes SRAM angeschlossen), dann muss man noch Verzögerungszeiten für I/O–Pads des Chips mit eventueller Anpassung von Spannungspegeln berücksichtigen.