Del 10: Sekvensielle kretser

YNGVAR BERG

I. Innhold

GRUNNLEGGENDE problematikk ved sekvensiering blir gjennomgått. Sekvenseringsmetoder med vipper, tofase transparente latcher og latcher som styres av klokkepulser blir presentert. Begrensinger for maksimal og minimal frorsinkelse i kombinatorisk logikk mellom sekvenseringselementene blir diskutert. Fordeling av tid mellom klokkefaser blir kort presentert og effekten av klokkeskew blir omhandlet. Alle henvisninger til figurer er relevant for Weste & Harris [1].

1. *Innhold.*
2. *Introduksjon til sekvensielle kretser.* Kapittel 7.1 side 383 - 384.
3. *Sekvenseringsmetoder.* Kapittel 7.2.1 side 385 - 387.
4. *Begrensinger for maks forsinkelse.* Kapittel 7.2.2 side 388 - 392.
5. *Begrensinger for minimum forsinkelse.* Kapittel 7.2.2 side 392 - 396.
6. *Fordeling av tid mellom klokkefasene.* Kapittel 7.2.4 side 396 - 399.
7. *Klokke skew.* Kapittel 7.2.5 side 399 - 402.

II. Introduksjon til sekvensielle kretser
(Kapittel 7.1 side 383 - 384)

Dynamiske kretser kan karakteriseres ved at utgangene på kretsene, eller portene, er en funksjon av inngangene i samme tidsperiode. *Sekvensielle kretser* derimot kan karakteriseres ved at utgangene på kretsene er en funksjon av inngangene i forrige tidsperiode og samme tidsperiode. Vi sier at en sekvensiell krets har en *tilstand*. Tilstandsmaskiner og pipeline systemer er viktige eksempler på sekvensielle kretser.

Sekvenseielle kretser lages oftest ved hjelp av *latcher* eller *vipper* og kalles ofte *hukommelse*. Formålet med latcher og vipper er ikke primært hukommelse, men å skille en tilstand fra forrige tilstand i en sekvens. Vi kaller derfor slike latcher og vipper sekvensielle kretser.

Det er vanlig å skille statiske- og dynamiske sekvensielle kretser. Det er viktig å være klar over at med *statiske sekvensielle kretser* mener vi ikke kretser uten klokkeinnganger, men latcher eller vipper som vil holde en verdi uten signifikant lekkasje. *Dynamiske sekvensielle kretser* vil bare holde en verdi en meget begrenset tid og er derfor avhengig av hyppig oppfriskning av signalet. Statiske sekvensielle kretser vil ha en eller annen form for tilbakekobling som bidrar til å holde et signal uavhengig av klokkefrekvens i et system.

A. Mål

Kunne skille ulike typer statiske- og dynamiske sekvensielle kretser.

B. Notater

III. Introduksjon til sekvensering av statiske kretser
(Kapittel 7.2 side 384)

* *Latcher og vipper[2]. (INF3400 Del 1: Kapittel XI side 10 - 11 og Kapittel XII side 11 - 12)*

* *RC forsinkelsesmodeller[3]. (INF3400 Del 4: Kapittel VII side 7 - 8)*

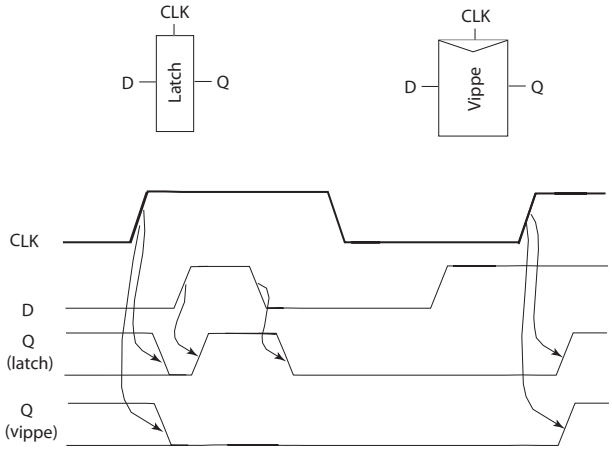

Fig. 1. *Latcher og vipper. (FIG7.1)*

De vanligste sekvenseringselementene er latcher og vipper som vist i Fig. 1. Latcher og vipper har inngang (D) og styres av klokkesignal (CLK), og utgang (Q).

En *latch* er transparent når $CLK = 1$ som betyr at utgangen er følsom for endringer på inngangen. Når $CLK = 0$ vil ikke utgangen på en latch påvirkes av inngangen. En latch er *nivåfølsom*, dvs. utgangen følger inngangen når $CLK = 1$, og utgangen er stabil (buffer) når $CLK = 0$.

En *vippe* er en *kantfølsom* krets, dvs. ved en postiv klokketransisjon vil verdien på inngangen D kopieres til utgangen Q, og utgangen vil ikke påvirkes av endringer på inngangen D i resten av klokkeperioden.

A. Mål

Forstå den grunnleggende virkemåten til latcher og vipper, og få et inntrykk av forskjellen på latcher og vipper.

B. Notater

IV. Sekvenseringsmetoder
(Kapittel 7.2.1 side 385 - 387)

* *Latcher og vipper[2]. (INF3400 Del 1: Kapittel XI side 10 - 11 og Kapittel XII side 11 - 12)*

Dersom vi har delkrets i form av *kombinatorisk logikk*, dvs. porter eller delkretser uten synkroniseringssignaler (klokker) som i praksis vil si en transparent delkrets, vil det som regel være behov for å synkronisere den kombinatoriske delkretsen med andre signaler fra andre delkretser. Det er ulike metoder for å synkronisere signaler eller ulike sekvenseringsmetoder som kan anvendes. Synkroniseringselementene eller sekvenseringselementene som benyttes er typisk latcher eller vipper.

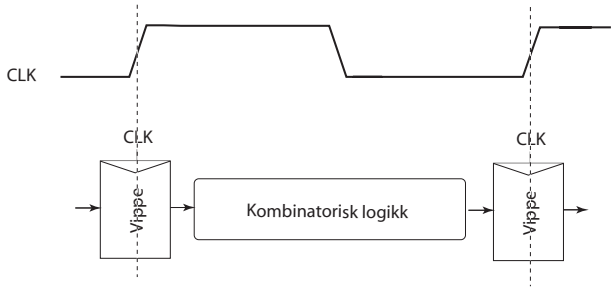

Fig. 2. *Sekvensering (synkronisering) med vipper. (FIG7.2)*

I Fig. 2 er det vist kombinatorisk logikk synkronisert ved hjelp av *kantfølsomme vipper*. Synkronsieringspunktet er gitt av en positiv klokkeflanke. Ved en positiv klokkeflanke vil vippene kopiere inngangsignalet til utgangen og holde verdien inntil neste positive klokkeflanke. Ved å invertere synkroniseringssignalene vil man få synkroniseringspunkt ved negative klokkeflanker.

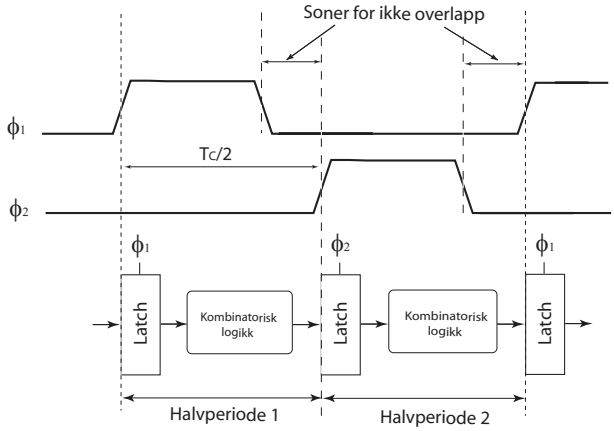

Fig. 3. *Sekvensering (synkronisering) med latcher. (FIG7.2)*

Synkronisering ved hjelp av latcher er vist i Fig. 3. I et to-fase system er det vanlig å bruke to-fase ikke-overlappende klokker, der klokkefasene (klokkesignalene ϕ_1 og ϕ_2) ikke er høye samtidig. Hele klokkeperioden er T_c. En halvperiode vil bestå av en tid der en av klokkesignalene er høye etterfulgt av en tid der ingen av klokkesignalene er høye (ikke-overlappende).

Synkronisering ved hjelp av latcher som styres av pulser er vist i Fig. 4. Som vi ser er klokkesignalet anderledes enn for vanlige latcher eller vipper. Klokkesignalet kan karakteriseres som korte pulser, dvs. med en duty cycle som er vesentlig mindre enn 50%. Latchene styres med korte klokkepulser som trigger en

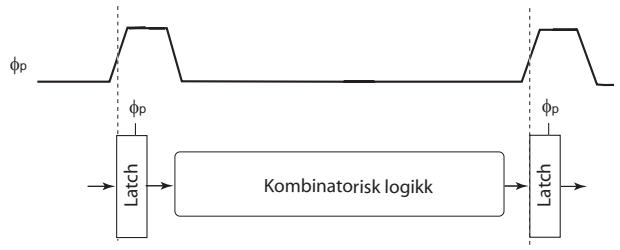

Fig. 4. *Sekvensering (synkronisering) med latcher som styres av pulser. (FIG7.2)*

innlesning (sampling) av inngangen. Latchen vil selv etter kort tid sørge for å holde den samplede verdien når klokkesignalet er lavt. Samplingstiden er meget kort sammenlignet med tidsforsinkelsen i kombinatorisk logikk mellom latchene. På denne måten vil kombinatorisk logikk virke som en lagring av en tilstand.

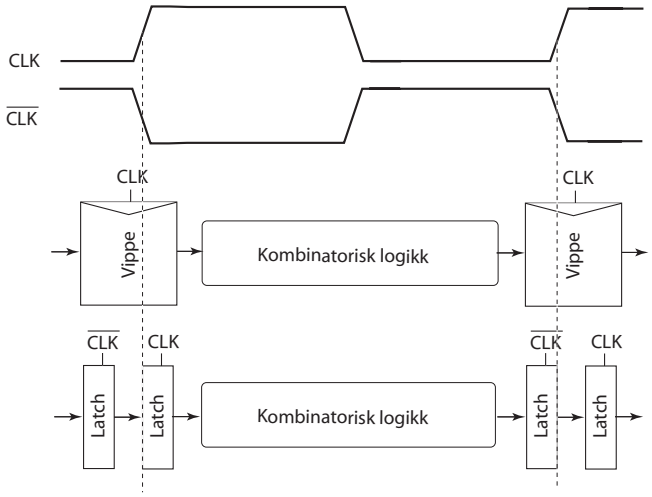

Fig. 5. *Sekvensering (synkronisering) med vipper realisert som latcher med klokke og invertert klokke. (FIG7.3)*

Sekvensering ved hjelp av vipper styrt av klokke (CLK) og invertert klokke ($\overline{CLK}$) er vist i figur 5. Her er to lacher plassert intill hverandre. Vi må da forutsette at de to latchene ikke er transparente samtidig, dvs. vi kan ikke tillate at CLK og $\overline{CLK}$ er høye samtidig. I praksis må vi sikre oss ved å ha en ikke-overlappende periode der CLK og $\overline{CLK}$ er lave samtidig. Et slikt system setter strenge krav til forholdet mellom CLK og $\overline{CLK}$.

Term	Kommentar
t_{pd}	Logisk propagering forsinkelse
t_{cd}	Logisk contamination forsinkelse
t_{pcq}	Latch/vippe klokke til Q propagering forsinkelse
t_{ccq}	Latch/vippe klokke til Q contamination forsinkelse
t_{pdq}	Latch D til Q propagering forsinkelse
t_{cdq}	Latch D til Q contamination forsinkelse
t_{setup}	Latch/vippe setup tid
t_{hold}	Latch/vippe hold tid

TABLE I
Notasjon for sekvenseringselementer.

I tabell I er de ulike utrykkene (termene) for forsinkelse som

legger begrensinger for timing i sekvensielle kretser.

A. *Timing for kombinatorisk logikk*

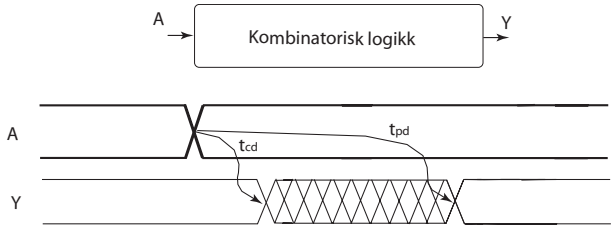

Fig. 6. *Timing for kombinatorisk logikk. (FIG7.4a)*

Tidsforsinkelse i kombinatorisk logikk er gitt av *contamination forsinkelse* og *propagering forsinkelse* og vist i Fig. 6. Utgangen kan begynne å endre seg etter tiden gitt av contamination forsinkelse t_{cd} og utgangen vil bli stabil etter maksimal tidsforsinkelse fra inngang til utgang som er gitt av propageringsforsinkelse t_{pd}. Contamination forsinkelse og propageringsforsinkelse kan være forskjellig på grunn av forskjellige kjeder eller signalveier i logikken.

B. *Timing for vippe*

Dersom vi har sekvensielt system bestående av vipper som klokkes med CLK og $\overline{CLK}$ må vi forutsette at disse signalene er presist i motfase.

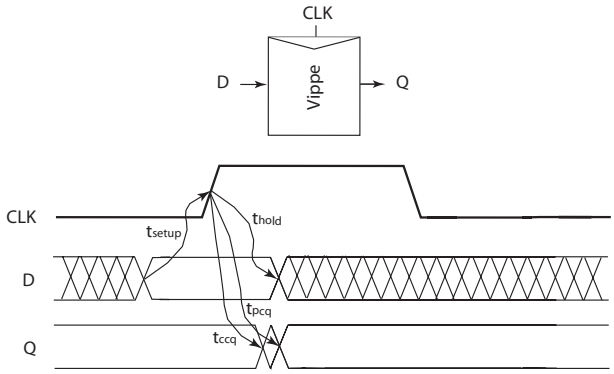

Fig. 7. *Timing for vippe. (FIG7.4b)*

Tidsforsinkelse i en vippe er vist i Fig. 7. Vippen vil holde en tilstand i hele perioden med unntak av lagring av ny verdi en kort stund etter at klokkesignalet skifter fra 0 til 1. Vippen er avhengig av *setup-* og *hold tid*. Dette betyr at inngangen D må være stabil en stund før klokkesignakel CLK endres fra 0 til 1 og forbli stabil tilstrekkelig lenge etter at klokkesignalet har blitt endret. Setup- og hold tid er sikkerhetsmarginer for å sikre riktig vippe funksjon.

Timing detaljer for vipper er vist i Fig. 8, der vippene er realisert som to latcher i motfase. Når klokkesignalet CLK er lavt vil inngangslatcher i vippenene være transparente og kontinuerlig sample inn $\overline{QM} = D$, samtidig med at utgangen Q oppfriskes ved hjelp av to invertere i tilbakekobling. Vippens funksjon er å sample inn D i slutten av tidsperioden når $CLK = 0$, dvs. vi ønsker å laste inn verdien D akkurat når CLK endres fra 0 til 1. Transmisjonsport 1 (TP1) stenger og TP2 åpner slik at signalet som ligger på $Z1 = QM$ vil påvirke X via TP2 og deretter $\overline{QM}$. Vi har en situasjon der $Z1 = X(= D)$. Dersom D forandres og påvirker X før TP2 åpner helt kan vi få

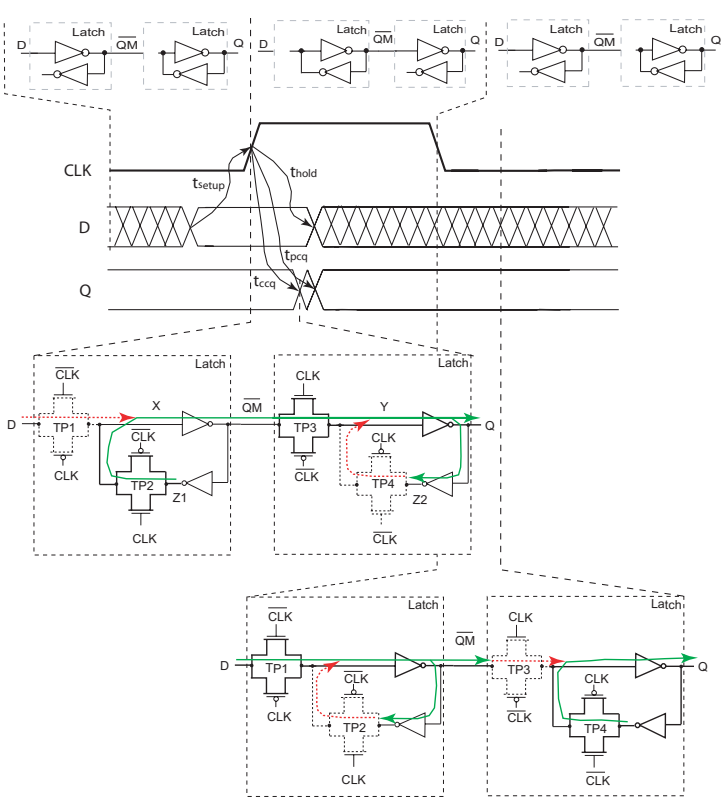

Fig. 8. *Timing for vippe. (FIG7.4b)*

en endring i X og deretter i $\overline{QM}$ som er uønsket. En hold tid på D vil sikre at $\overline{QM}$ ikke endres uønsket. I tillegg vil det være gunstig at TP1 stenger før TP2 åpner slik at X ikke kan drives via TP1 når tilbakekoblingen skal være aktiv. For latchen som kontrollerer utgangen på vippen (Q) har vi nå en situasjon der TP3 skal overstyre TP4. Det kan være gunstig at TP4 stenger før TP3 åpner for å redusere støy på utgangen. Vi ser at contamination forsinkelse og propageringsforsinkelse er tilnærmet like og er tidsforsinkelsen fra $\overline{QM}$ til Q når TP3 åpner, dvs fra stigende klokkeflanke. Ved fallende klokkeflanke vil det kunne oppstå en transparent situasjon dersom TP3 og TP1 er åpne samtidig. Det er viktig at TP3 stenger før TP1 åpner for å sikre at ikke inngangsignalet D påvirker Q. Vi ønsker i denne situasjonen når $CLK = 0$ at vippen skal holde sin lagrede verdi.

C. *Timing for Latch*

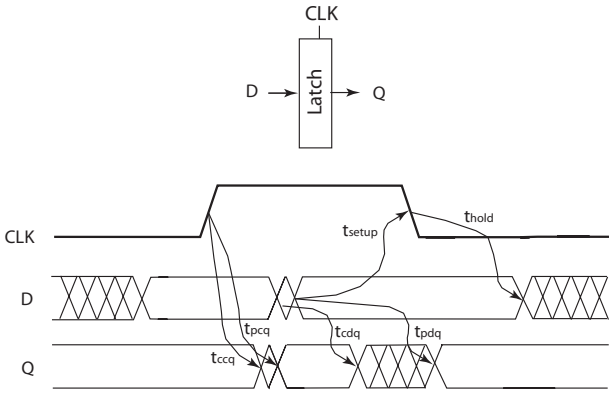

Fig. 9. *Timing for latch. (FIG7.4c)*

Timing for en latch er vist i Fig. 9. Samplingstidspunktet for latchen er ved fallende klokkeflanke. Vi må forutsette at inngangssignalet D kommer fra en latch styrt av et annet klokkesignal, for eksempel invertert klokke som gir klokkesignaler i motfase. Mellom latchene er det typisk kombinatorisk logikk.

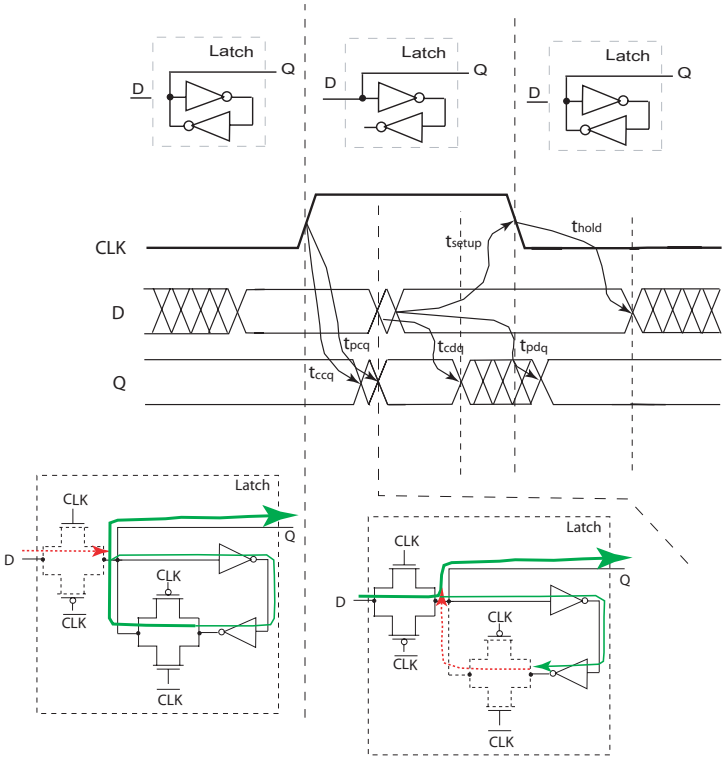

Fig. 10. *Timing for latch. (FIG7.4c)*

Timingdetaljer for en latch er vist i Fig. 10. Når klokkesignalet CLK er lavt vil latchen fungere som et minneelement med tilbakekobling. Dersom vi ser på latchen og kombinatorisk logikk som prosesserer signalet fra latchen og neste latch får vi et system som vist i Fig. 3 med $CLK = \phi_1$ og $\overline{CLK} = \phi_2$. Fra stigende klokkeflanke får vi klokke til Q forsinkelse t_{ccq} og t_{pcq} som vist i Fig. 10. Latchen er transparent når klokkesignalet er høyt og enhver forandring på D vil påvirke Q. Vi må forutsette at D er stabil en liten stud før fallende klokkeflanke slik at latchen rekker å sample riktig verdi. Vi kaller denne tiden setup tid. For å være sikker på riktig sampling må D være stabil en stund etter at klokkesignalet har blitt 0. Vi kaller dette for hold tid.

D. Mål

Forstå detaljer ved timing av sekvensielle kretser.

E. Notater

V. Begrensninger for maks forsinkelse
(Kapittel 7.2.2 side 388 - 392)

Ideelt vil vi ønske hele klokkeperioden tilgjengelig for prosessering av signaler. Sekvenseringen vil medføre en viss overhead som er knyttet til sekvenseringselementene. Dersom tidsforsinkelsen i kombinatorisk logikk er for stor vil en vippe som skal lagre resultatet få signalet for sent slik at situasjonen ikke tilfredstiller setup tid. Vi kaller en slik feil for set feil eller maks forsinkelse feil.

A. Vipper

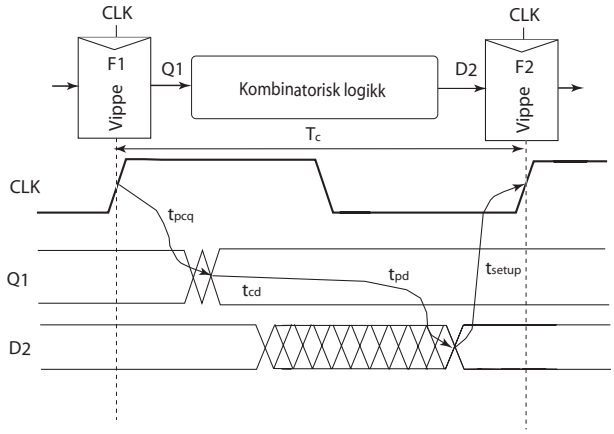

Fig. 11. *Begrensninger for maks forsinkelse. (FIG7.5)*

Begrensninger for maks forsinkelse er vist i Fig. 11. Dersom vi antar at vippene $F1$ og $F2$ styres av identiske klokkesignaler, dvs. at klokkesignalene har transisjoner på nøyaktig samme tidspunkt, er maksimal tid fra en vippe til neste vippe gitt av T_c:

$$T_c \geq t_{pcq} + t_{pd} + t_{setup}, \tag{1}$$

der t_{pcq} er propageringsforsinkelse for klokke til utgang (Q) for vippe, t_{pd} er propageringsforsinkelse i kombinatorisk logikk og t_{setup} er setup tid for vippe.

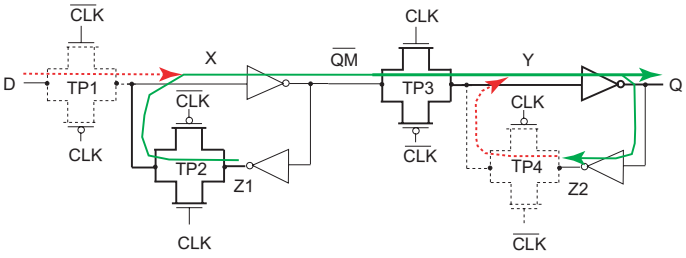

Fig. 12. *Propageringsforsinkelse for klokke til utgang (Q) for vippe.*

Propageringsforsinkelse for klokke til utgang (Q) for vippe er vist i Fig. 12. Når klokkesignalet er lavt vil inngangslatchen sample inn D til $\overline{QM}$ kontinuerlig. Når klokke signalet endres fra 0 til 1 lukker transmisjonsportene $TP1$ og $TP4$ mens $TP2$ og $TP3$ åpner. verdien som er ligger på $\overline{QM}$ (lagret) vil transmitteres via $TP3$ til Q. Propageringstidsforsinkelsen i dette tilfellet er gitt av en transmisjonsport og en inverter i kjede og last på Y og utgangen Q.

Propageringsforsinkelse i kombinatorisk nettverk mellom de to vippene i Fig. 11 er gitt av en den signalveien mellom $Q1$ og $D2$ som har størst tidsforsinkelse. Vi kaller den aktuelle signalveien en *kritisk signalvei*.

Setuptiden for vippen er en sikkerhetsmargin for å sample riktig verdi i inngangslatchen i vippen. Setuptid varierer med prosess og implementasjon av vippen.

Vi kan uttrykke begrensning for maks tidsforsinkelse på en annen form:

$$t_{pd} \quad \leq \quad T_c - (t_{setup} + t_{pcq}), \tag{2}$$

der $(t_{setup} + t_{pcq})$ er overhead ved sekvenseringen. Vi ser at ved å redusere t_{setup} og t_{pcq} til et minimum får vi mer av tiden som er tilgjengelig til å prosessere signaler i kombinatorisk logikk (gjøre beregninger).

Tiden tilgjengelig i en klokkeperiode T_c er direkte avhengig av t_{setup} og t_{pcq} for vipper og tidsforsinkelse i kritisk signalvei i kombinatorisk logikk mellom vipper. Klokkefrekvens, eller maksimal klokkefrekvens, er direkte knyttet til klokkeperioden:

$$f_{max} \quad = \quad \frac{1}{T_c}. \tag{3}$$

B. Latcher

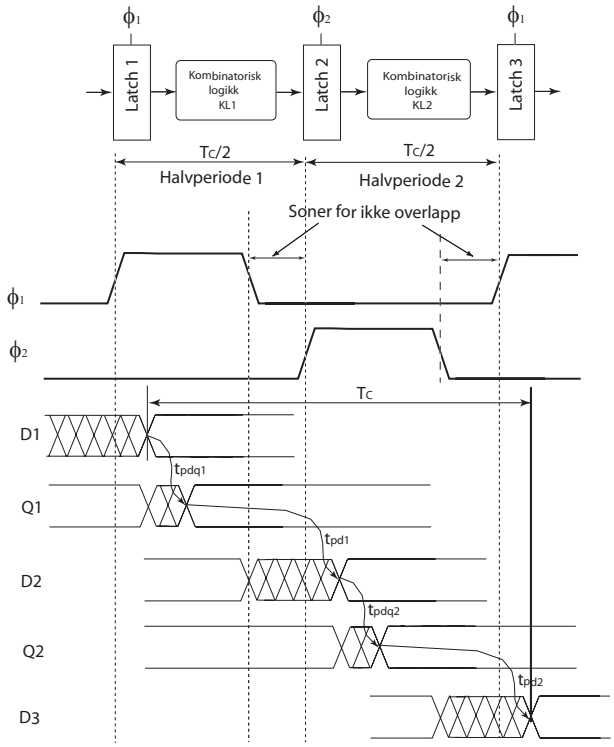

Fig. 13. *Begrensinger for maks forsinkelse i et sekvenseringssystem med latcher styrt av tofase klokker (FIG7.7).*

Timingdetaljer i et sekvenseringssystem med transparente latcher som styres av tofase klokker er vist i Fig. 13. Vi antar at inngangen $D1$ ankommer latch 1 når $\phi_1 = 1$ og propagerer i kombinatorisk logikk $KL1$ fordi latch 1 er transparent når $\phi_1 = 1$. Kritisk signalvei i KL1 vil bestemme maksimal tidsforsinkelse t_{pd1} for KL1. Vi må forutsette at $D2$ er stabil i god tid før ϕ_2 svinger fra 1 til 0 slik at vi får riktig verdi samplet (latchet) i latch 2. Tilsvarende argumentasjon gjelder for latch 3 osv. Vi kan uttrykke en klokkeperiode T_c som:

$$T_c \quad \geq \quad t_{pdq1} + t_{pd1} + t_{pdq2} + t_{pd2}. \tag{4}$$

Dersom vi løser med hensyn på total propageringsforsinkelse i hele klokkeperioden får vi:

$$\begin{aligned} t_{pd} \quad &= \quad t_{pd1} + t_{pd2} \\ &\leq \quad T_c - (2t_{pdq}), \end{aligned} \tag{5}$$

der $(2t_{pdq2})$ er overhead gitt av propageringsforsinkelse i latchene, som vi antar er lik for de aktuelle latchene.

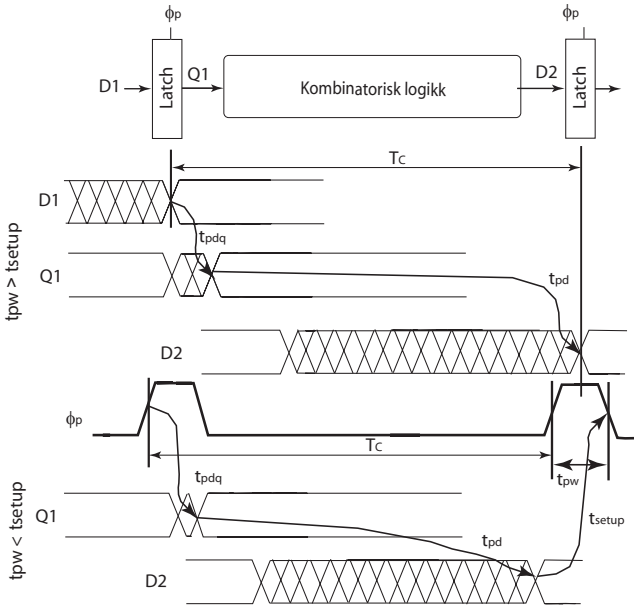

Fig. 14. *Begrensninger for maks forsinkelse i et sekvenseringssystem med latcher styrt av pulser. (FIG7.8).*

Begrensninger for maks forsinkelse i et sekvenseringssystem med latcher styrt av pulser er vist i Fig. 14. Dersom pulsbredden t_{pw} er større enn setuptiden t_{setup} har vi situasjonen som vist øverst i figuren. Vi trenger i dette tilfellet ikke å forutsette at utgangen av kombinatorisk logikk $D4$ er stabil før klokkepulsen kommer. Dersom klokkepulsen er mindre enn setup tiden har vi situasjonen som vist nederst, og vi må da forutsette at inngangen til latchen er stabil før klokkepulsen kommer. Vi kan beskrive dette som:

$$T_c \quad \geq \quad \max\left(t_{pdq} + t_{pd}, t_{pcq} + t_{pd} + t_{setup} - t_{pw}\right), \tag{6}$$

som gir:

$$t_{pd} \quad \leq \quad T_c - \max\left(t_{pdq}, t_{pcq} + t_{setup} - t_{pw}\right), \tag{7}$$

der $\max\left(t_{pdq}, t_{pcq} + t_{setup} - t_{pw}\right)$ er overhead inkludert forsinkelse i en latch.

C. Mål

Forstå hva som begrenser utnyttelse av en klokkeperiode ved sekvensering.

D. Notater

VI. Begrensninger for minimum forsinkelse
(Kapittel 7.2.3 side 392 - 396)

Sekvenseringselementer bør kunne plasseres inntil hverandre uten vesentlig kombinatorisk logikk mellom elementene. Eksempel på sekvensielle systemer med minimal kombinatorisk logikk mellom sekvenseringselementene er *pipeline systemer*.

Dersom hold tid er stor og contamination forsinkelsen er liten kan data propagere gjennom to sekvenseringselementer ved en klokkeflanke. En slik feil kalles *race feil*, *hold tid feil* eller *minimum-forsinkelse feil*.

A. Vipper

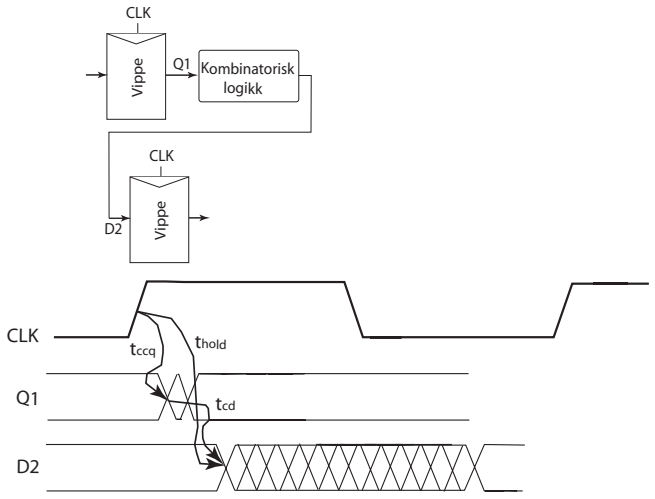

Fig. 15. *Begrensinger på minimumsforsinkelse for vipper. (FIG7.9).*

Begrensninger for minimum tidsforsinkelse for vipper er vist i Fig. 15 der vi antar at klokkesignalene til de to vippene er helt i fase (like).

Detaljer for et system med to vipper som er koblet sammen uten (minimalt) kombinatorisk logikk for $CLK = 0$ er vist i Fig. 16. Vi ser at så lenge $CLK = 0$ vil latch 1-1 i den første vippen følge inngangen D, dvs. vi latcher inn D i $\overline{Q1M}$. Utgangen på den første vippen $Q1$ holdes stabil ved hjelp av tilbakekobling i latch 1-2, og $Q1$ føres via eventuell kombinatorisk logikk til den neste vippen, nærmere bestemt latch 2-1 som latcher (sampler) inn $Q1$ til $\overline{Q2M}$. Vi ser at i slutten av perioden hvor $CLK = 0$ vil TP1, TP4, TP5 og TP8 være helt åpne og TP2, TP3, TP6 og TP7 være helt lukket. Vi må forutsette at inngangen D er stabil en stund før stigende klokkeflanke (t_{setup}). En kritisk situasjon som medfører feil er dersom TP3 og TP5 er åpne samtidig slik at $\overline{Q1M}$ blir transmittert til $\overline{Q2M}$ via $Q1$. Vippenes funksjon er å lagre verdier i etterfølgende klokkeperioder.

Dersom vi ser på detaljene rundt stigende klokkeflanke, som er vist i Fig. 17, ser vi at $\overline{Q1M}$ skal endre $Q1$ med klokke til Q contamination forsinkelse t_{ccq}. Det vil si at vi får en endring, men ikke nødvendigvis stabil verdi, på $Q1$ ved tidspunktet t_{ccq} etter stigende klokketransisjon. Ideelt sett har nå den neste vippen lukket TP5 og kan ikke påvirkes av endringer på $D2 = Q1$. For å sikre at en vippe ikke endres feilaktig er det påkrevd at det defineres en hold tid for inngangen. I dette tilfellet betyr det at den siste vippen forutsetter at inngangen $D2$ er stabil en liten stund etter stigende klokkeflanke. Det er avgjørende at ikke $D2$ endres som følge av endring på $Q1$ idet ved stigende klokkeflanke før vippens setup tid er over. Vi kan uttrykke dette

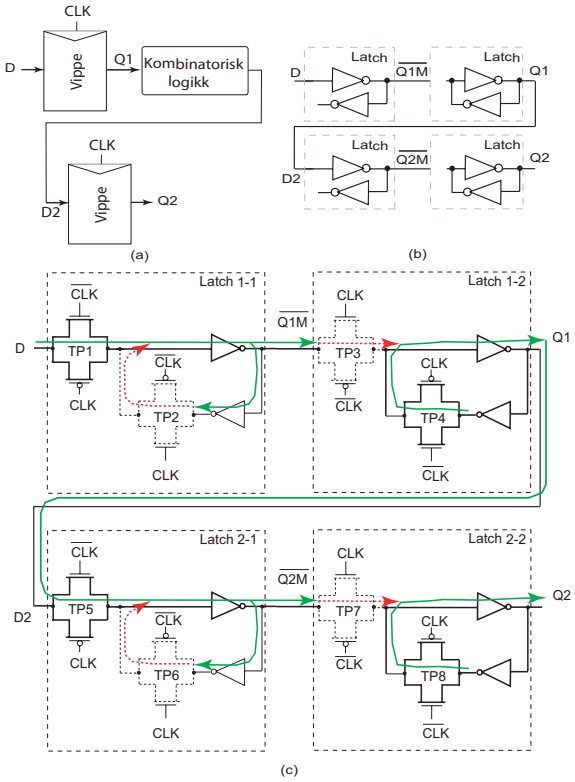

Fig. 16. *Begrensinger på minimumsforsinkelse for vipper. Detaljer i timing når $CLK = 0$. (FIG7.9).*

som

$$t_{cd} \quad \geq \quad t_{hold} + t_{ccq}, \tag{8}$$

der t_{cd} er contamination forsinkelse i kombinatorisk logikk[1] mellom vippene. Med andre ord, det er viktig at tidsforsinkelsen mellom vippene er så stor at inngangen til vippe nummer 2 ikke har har fått ny verdi fra latch 1-1 før setup tiden til vippe 2 er over. Dersom $D2$ endres før setuptiden er over vil latch 1-2 og latch 2-1 være transparente samtidig slik at $\overline{Q2M}$ blir lik $\overline{Q1M}$, som vil medføre at $Q2$ blir lik $Q1$ i neste omgang. Dersom contamination forsinkelse, dvs. klokke til Q forsinkelse, for vippen er større enn hold tid kan vippene plasseres helt inntil hverandre. I dette tilfellet vil ikke vippe nummer 2 rekke å reagere på endringer på inngangen for tidlig.

B. Latcher

I Fig. 18 er begrensninger på minimumsforsinkelse for latcher som er styrt av to fase lokker vist. Latchene styres av tofase ikkeoverlappende klokker som skal garantere at to latcher som styres av hver sin klokkefase ikke er åpne samtidig. Når begge klokkefasene ϕ_1 og ϕ_2 er lave samtidig skal begge latchene være lukket slik at utgangene ikke skal kunne påvirkes av inngangene. Ved stigende klokkeflanke på ϕ_1 åpner latchene som er styrt av ϕ_1 slik at $D1$ latches inn til $Q1$. Merk at latchene er *nivåfølsomme*, dvs. utgangen på latchene vil påvirkes av inngangen så lenge latchen er åpen, i motsetning til en vippe som er *kantfølsom*. Vi forutsetter derfor at latchen som styres av ϕ_2 har en hold tid som går utover tiden når $\phi_2 = 1$. Vi kan anta

[1] I eksemplet er det ikke kombinatorisk logikk mellom vippene slik at contamination forsinkelse blir minimum forutsatt at ikke avstanden mellom vippene er stor.

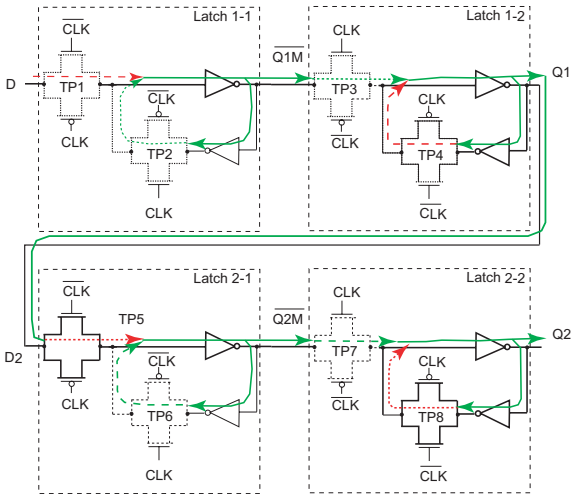

Fig. 17. *Begrensinger på minimumsforsinkelse for vipper. Detaljer i timing når* $CLK = 0 \rightarrow 1$. *(FIG7.9).*

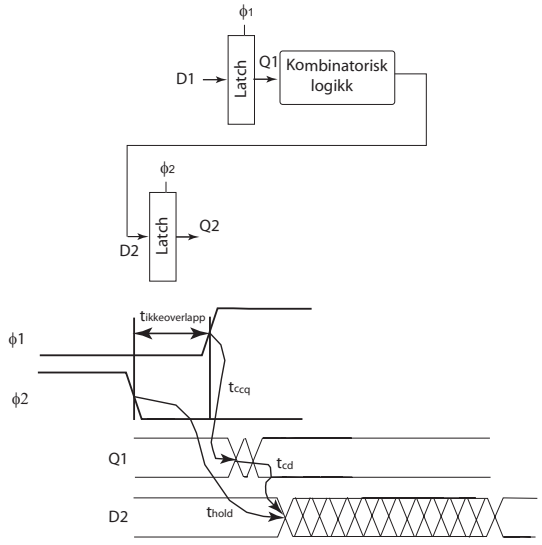

Fig. 18. *Begrensninger på minimumsforsinkelse for latcher som er styrt av to fase klokker. (FIG7.10).*

at denne hold tiden t_{hold} er så lang at den kan påvirke utgangen $Q2$ etter at $Q1$ og $D2$ er endret som følge av latcing ved tidspunktet når ϕ_1 svinger fra 0 til 1.

Timing detaljer for latcher som er styrt av to fase lokker ved stigende transisjon på ϕ_1 er vist i Fig. 19. Dersom hold tiden for latch styrt av ϕ_2 er for lang i forhold til tidsforsinkelse mellom latchene kan vi latche inn feil verdi. Vi har en situasjon der TP1 er PÅ slik at latch styrt av ϕ_1 er åpen og TP3 ikke er helt AV slik at latch styrt av ϕ_2 er delvis åpen. I denne situasjonen er den ene latchen åpen og den andre delvis åpen slik at de to latchene satt sammen blir delvis transparent. Vi kan uttrykke betingelser for korrekt latching ved å sette en nedre grense for contamination forsinkelse for kombinatorisk logikk mellom latchene:

$$t_{cd1}, t_{cd2} \quad \geq \quad t_{hold} - t_{ccq} - t_{ikkeoverlapp}, \tag{9}$$

der t_{hold} er hold tid for latchene, t_{ccq} er klokke til Q contamination forsinkelse for latchene og $t_{ikkeoverlapp}$ er tiden der begge klokkefasene er lave. Alternativt kan vi utrykke dette som en begrensning på hold tiden:

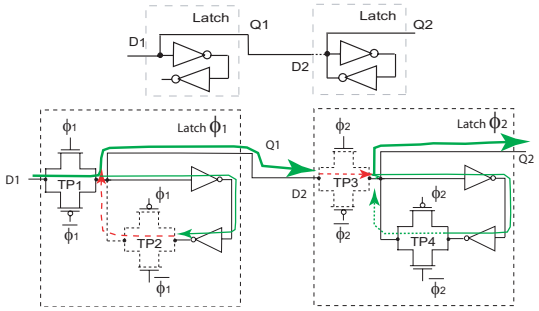

Fig. 19. *Begrensninger på minimumsforsinkelse for latcher som er styrt av to fase lokker. Detaljer ved stigende transisjon på* ϕ_1. *(FIG7.10).*

$$t_{hold} \quad \leq \quad t_{ikkeoverlapp} + t_{ccq} + t_{cd}. \tag{10}$$

Dersom tiden der begge klokkefasene er lave og $t_{ikkeoverlapp}$ er tilstrekkelig lang vil vi ikke få problem med for liten tidsforsinkelse i kombinatorisk logikk mellom to latcher.

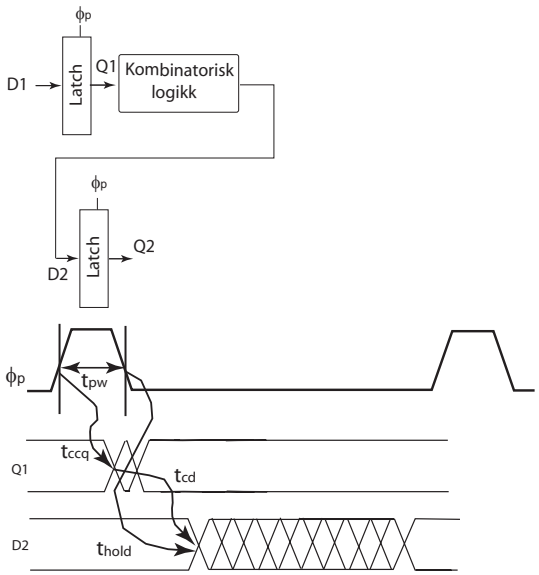

Fig. 20. *Begrensninger på minimumsforsinkelse for latcher som er styrt av klokkepulser. (FIG7.11).*

Latcher som styres av klokkepulser er vist i Fig. 20. Det er tilsvarende begrensninger for minimum tidsforsinkelse i kombinatorisk logikk mellom latchene som for tofase latcher. Vi kan utrykke dette som:

$$t_{cd} \quad \geq \quad t_{hold} - t_{ccq} + t_{pw}. \tag{11}$$

C. Mål

Forstå hva som begrenser utnyttelse av en klokkeperiode ved sekvensering.

D. Notater

VII. Fordeling av tid mellom klokkefasene
(Kapittel 7.2.4 side 396 - 399)

I et system med vipper vil data være tilgjengelig på utgangen ved stigende klokkeflanke. Den neste vippen (etterfølgende) skal være være klar til å lache ved neste stigende klokkeflanke. Dersom data ankommer tidligere vil vippen blokkere for inngangen ved at inngangslatchen sampler inn alle forandringer på inngangen, men slipper ikke gjennom signalet til utgangen før neste stigende klokkeflanke.

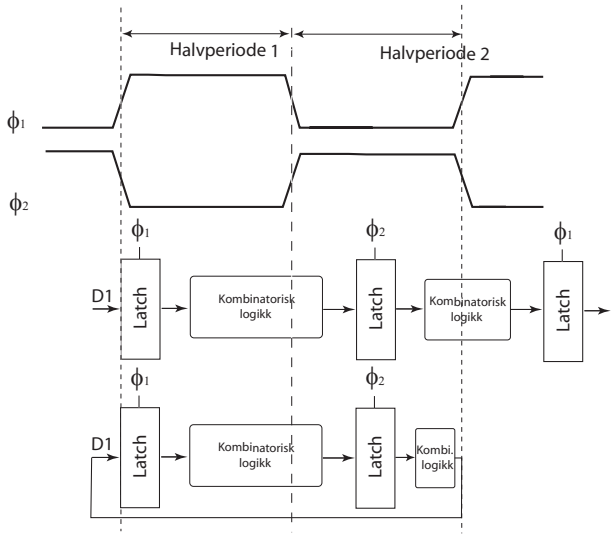

Fig. 21. *Fordeling av tid mellom klokkefaser. (FIG7.12).*

I et system med latcher derimot er ikke latchetidspunktet knyttet til klokkeflanker, men til klokkenivåer. Med andre ord er latchene transparente i en tidsperiode slik at presist tidspunkt for latchingen ikke er viktig. Vi kan med andre ord lage systemer der tidsforsinkelse i kombinatorisk logikk er forskjellig mellom ulike latcher uten at dette reduserer klokkefrekvensen. Dette kalles fordeling av tid mellom klokkefaser (time borrowing) og er vist i Fig. 21.

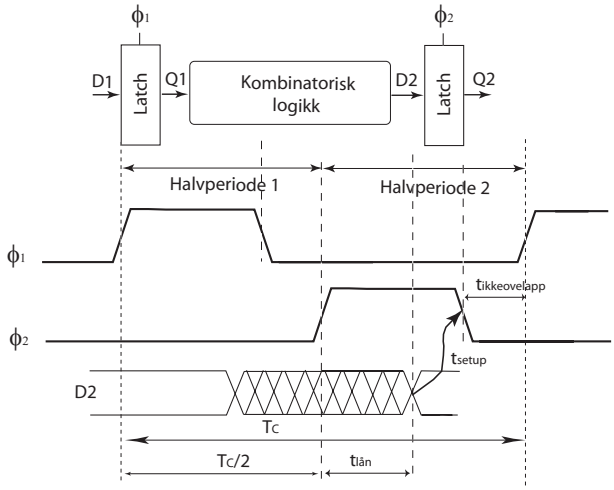

Fig. 22. *Fordeling av tid mellom klokkefaser. (FIG7.13).*

Maksimal fordeling av tid mellom to ikkeoverlappende klokkefaser er vist i Fig. 22. På grunn av at inngangsdata ikke må være stabil før ved fallende klokkesignal for mottager latch kan en fase låne opptil en halvpart av en halvperiode for en klokke frekvens:

$$t_{lån} \quad \leq \quad \frac{T_c}{2} - (t_{setup} + t_{ikkeoverlapp})\,. \tag{12}$$

I praksis vil fordeling av tid mellom klokkesykler i et system med latcher styrt av klokkepulser være svært begrenset og gitt av:

$$t_{lån} \quad \leq \quad t_{pw} + t_{setup}, \tag{13}$$

der t_{pw} er bredden på klokkepulsen.

I praksis vil ofte fordeling av tid mellom klokkefaser ikke være aktuelt ved design, men vil kunne fungere som en ekstra sikkerhet for at et system vil virke på grunn av ulik tidsforsinkelse i kombinatorisk logikk mellom latcher i et system.

A. Mål

Forstå hvordan man kan fordele tiden i en klokkeperiode ujevnt mellom to faser eller halvperioder.

B. Notater

Vi har til nå forutsatt ideelle klokkesignaler uten *skew* eller forskyvning. I praksis vil klokkeflankene komme til litt forskjellige tider for ulike latcher og vipper. Dette skyldes at sekvenseringselementene vil være plassert ulike steder på en krets og dermed representere ulik last[2] for klokkesignalet.

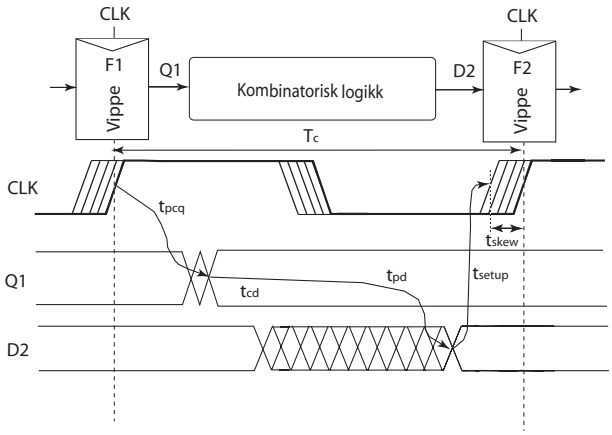

Fig. 23. *Klokkeskew og vipper. (FIG7.15a).*

I Fig. 23 er det vist et system med vipper som styres av et klokkesignal hvor tykk linje for CLK markerer det seneste tidspunktet for klokketransisjonene. Klokketransisjonene kan komme tidligere som vist i figuren. Den kritiske situasjonen for maksimal forsinkelse i et system med vipper er om vippen som sender et signal får klokketransisjonen sent og mottager vippen får klokketransisjonen tidlig. I dette tilfellet må klokkeskew trekkes fra den tiden systemet har tilgjenglig for å prosessere signaler i kombinatorisk logikk mellom vippene. Man regner da klokkeskew som en del av ovreheaden ved sekvenseringen.

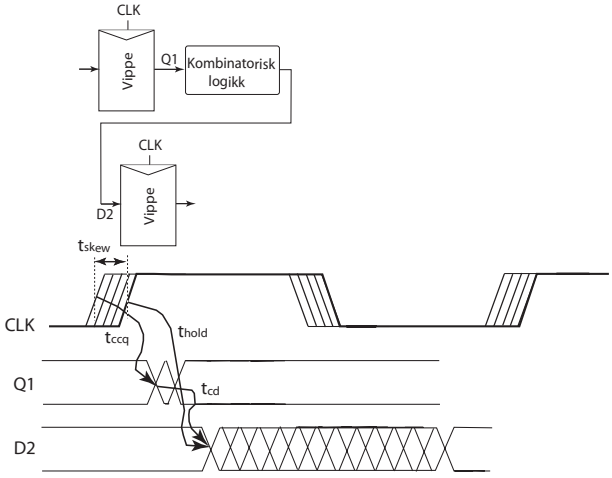

Fig. 24. *Klokkeskew og vipper. (FIG7.15b).*

Den kritiske situasjonen for minimum tidsforsinkelse har vi når sendervippen får klokketransjonen tidlig og mottager vippen får klokketransisjonen sent som vist i Fig. 24. I dette tilfellet vil den effektive holdtiden øke og vi får begrensingene:

[2]Spesielt ulik kapasitans og motstand på grunn av interkonnekt.

$$t_{pd} \leq T_c - (t_{pcq} + t_{setup} + t_{skew}), \quad (14)$$

der $(t_{pcq} + t_{setup} + t_{skew})$ er overhead i sekvenseringen. Vi har da:

$$t_{cd} \geq t_{hold} - t_{ccq} + t_{skew}). \quad (15)$$

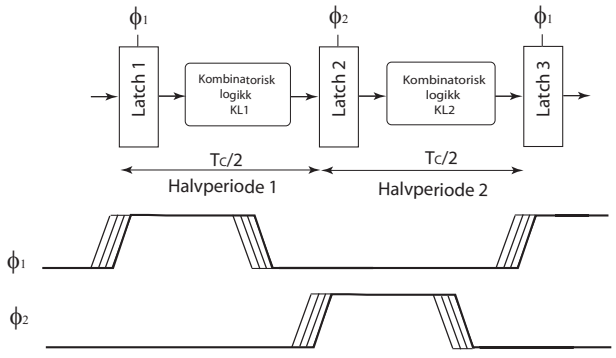

Fig. 25. *Klokkeskew og transparente latcher. (FIG7.16).*

I et tofase system med transparente latcher vil ikke klokkeskew redusere ytelsen, som vist i Fig. 25, sålenge de to klokkefasene er ikke-overlappende. Vi kaller et system med transparente latcher med tofase ikke-overlappende klokker for *skew tolerant*. Muligheten for å fordele tid mellom klokkefasene vil imidlertid bli mer begrenset:

$$\begin{aligned} t_{pd} &\leq T_c - 2t_{pdq} \\ t_{cd1}, t_{cd2} &\geq t_{hold} - t_{ccq} - t_{ikkeoverlapp} + t_{skew}, \end{aligned} \quad (16)$$

der t_{cd} er contamination forsinkelse for kombinatorisk logikk mellom latcher.

For latcher som styres av klokkepulser vil fordeling av tid i kombinatorisk nettverk mellom latcher bli svært begrenset fordi klokkeskew kan føre til at klokkepusene blir svært smale.

A. Mål

Forstå hvordan klokkeskew påvirker et sekvenseringssystem

B. Notater

IX. Indeks

References

[1] Neil H.E. Harris og David Harris"CMOS VLSI DESIGN, A circuit and system perspective" tredje utgave 2005, ISBN: 0-321-26977-2, *Addison Wesley*,
[2] Yngvar Berg"INF3400 Del:1"
[3] Yngvar Berg"INF3400 Del:4"