Sammanfattning
Föreläsning 11 - Digitalteknik

I boken: avsnitt 6.3-6.4, 5.3 (kap 5.5 i Hemert)

Asynkrona Sekvensnät + Mealy/Moore

Syftet med föreläsningen är dels att förstå skillnaden mellan grafer av typen Mealy och Moore. Dels att förstå vad ett asynkront sekvensnät är tillsammans med de speciella krav som ställs i form av realiserbarhet, samt en hasardfri och kapplöpningsfri realisering.

Vår vanliga definition av en graf är följande:

Definition. *En* Mealy *graf definieras av* $\mathcal{M} = \{\mathcal{I}, \mathcal{S}, \mathcal{Z}, \delta, \lambda\}$ *där*

- δ *nästa-tillståndsfunktionen,*
$$\delta : \mathcal{S} \times \mathcal{I} \to \mathcal{S}$$
- λ *är utsignalfunktionen,*
$$\lambda : \mathcal{S} \times \mathcal{I} \to \mathcal{Z}$$

En annan användbar definition (med annan utsignalfunktion) är följande:

Definition (6.6)**.** *En* Moore *graf är definierad som* $\mathcal{M} = \{\mathcal{I}, \mathcal{S}, \mathcal{Z}, \delta, \beta\}$ *där*

- δ *nästa-tillståndsfunktionen,*
$$\delta : \mathcal{S} \times \mathcal{I} \to \mathcal{S}$$
- β *är utsignalfunktionen,*
$$\beta : \mathcal{S} \to \mathcal{Z}$$

Mealy → Moore: Alla Mealy kan skrivas om till Moore om

- utsignalen är fördröjd en tidsenhet.
- vi kan behöva fler tillstånd.

Konvertering:

- Dela alla tillstånd så att alla inkommande bågar in i ett tillstånd har samma utsignalvärde.
- Flytta utsignalen in i tillståndet som bågen pekar på (nästa tillstånd).

Moore → Mealy Alla Moore kan skrivas som Mealy om vi tillåter att utsignalen är en direkt funktion (asynkron) av insignalen.

Konvertering:

- Flytta utsignalen till de inkommande bågarna till varje tillstånd.
- Använd RF-algoritmen.

Asynkrona sekvensnät

Definition (6.7)**.** *Ett tillstånd s är stabilt för insignal i om*

$$\delta(s, i) = s$$

dvs om vi för insignal i har en båge tillbaka till s.

Om det finns en väg för insignal i från tillstånd s_0 till det stabila tillståndet s så är s ett *successortillstånd* till s_0.

Definition (6.8)**.** *En graf är* asynkront realiserbar *om alla tillstånd har successortillstånd för alla insignalvärden.*

Definition. *I en* kapplöpningsfri *tillståndskodning så ändrar endast en tillståndsvariabel sitt värde vid en tillståndsövergång.*

Eftersom fördröjningen i olika komponenter (grindar) är olika, så kan det uppstå transienter i funktioner (felaktigt värde under en mycket kort tid). Detta fenomen kallas *hasard*.

För att undvika hasard måste realiseringen vara *hasardfri*. Detta fås om *alla* primimplikatorer tas med i realiseringen av funktionen.

Ett *asynkront sekvensnät* är inte klockat, utan tillstånden uppdateras kontinuerligt (tiden är kontinuerlig, ej diskret). I realiseringen sätter vi $q^+ = q$.

För att få korrekt beteende behöver vi då att

- grafen är *asynkront realiserbar*.
- tillståndskodningen är *kapplöpningsfri*.
- funktionerna är *hasardfria*.

Exempel: Minneselement kan realiseras som asynkrona sekvensnät.

En *latch* är ett enkelt minneselement. Signalen ϕ kontrollerar utsignalen z så att

$$z = \begin{cases} x, & \text{om } \phi = 1 \\ x_0, & \text{om } \phi = 0 \end{cases}$$

där x är insignal och x_0 är insignalen vid tiden när senast $\phi = 1$.

Ett *D-element* kan realiseras med två latchar och en styrkrets (master-slave konstruktion).