Digitale Signalverarbeitung auf FPGAs

Kap. 1 – Zeitdiskrete Signale und LTI-Systeme im Zeitbereich

2016

Prof. Dr. Christian Münker

Überblick Kapitel 1

- Grundelemente von zeitdiskreten Systemen
- Impulsantwort und zeitdiskrete Faltung
- Einführung in die *z*-Transformation
- Implementierungen zeitdiskreter Systeme

Digitale Signalverarbeitung auf FPGAs

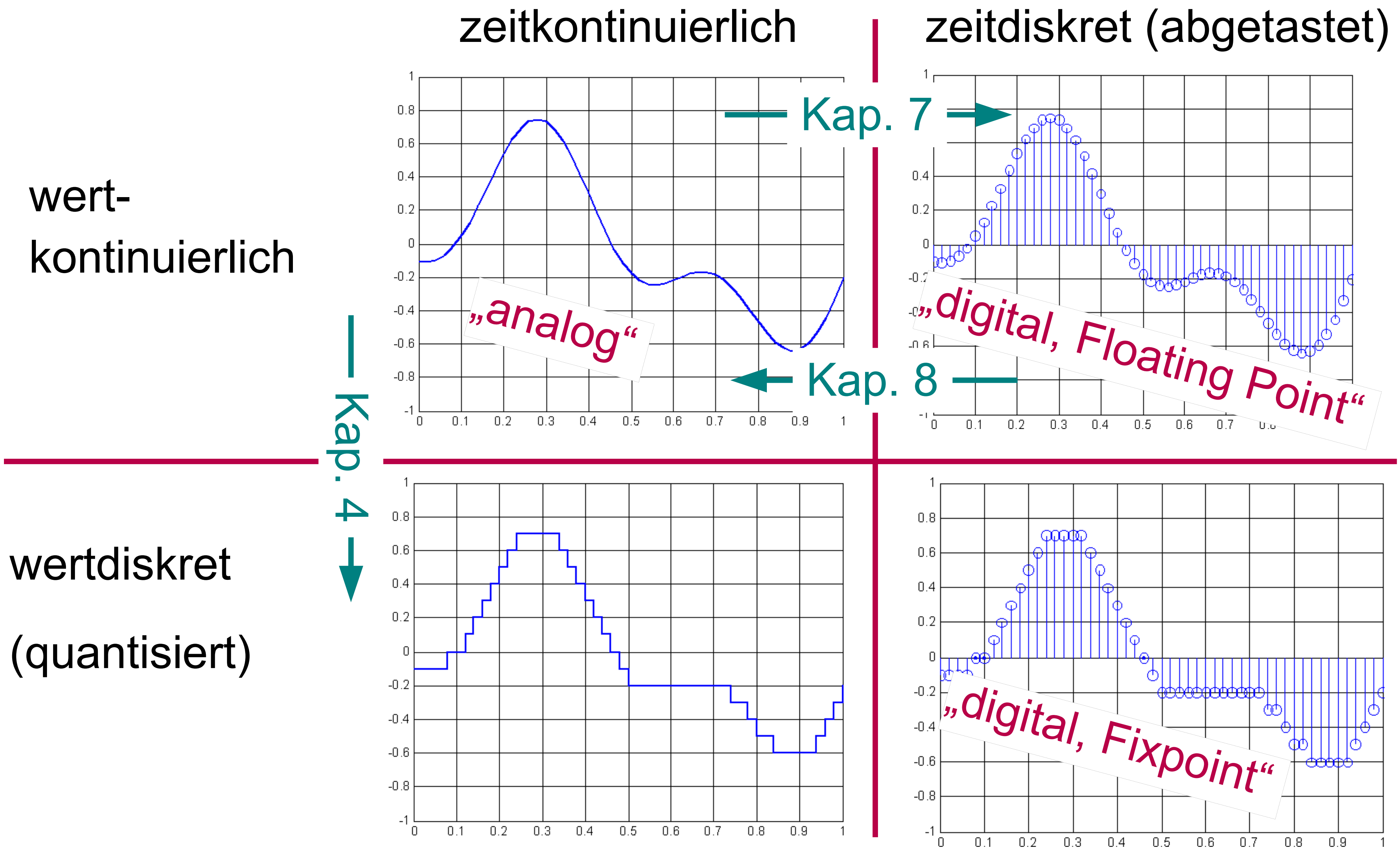
Kap. 1 Zeitdiskrete Signale und LTI-Systeme im Zeitbereich

Teil 1 *Grundelemente von zeitdiskreten Systemen*

2016

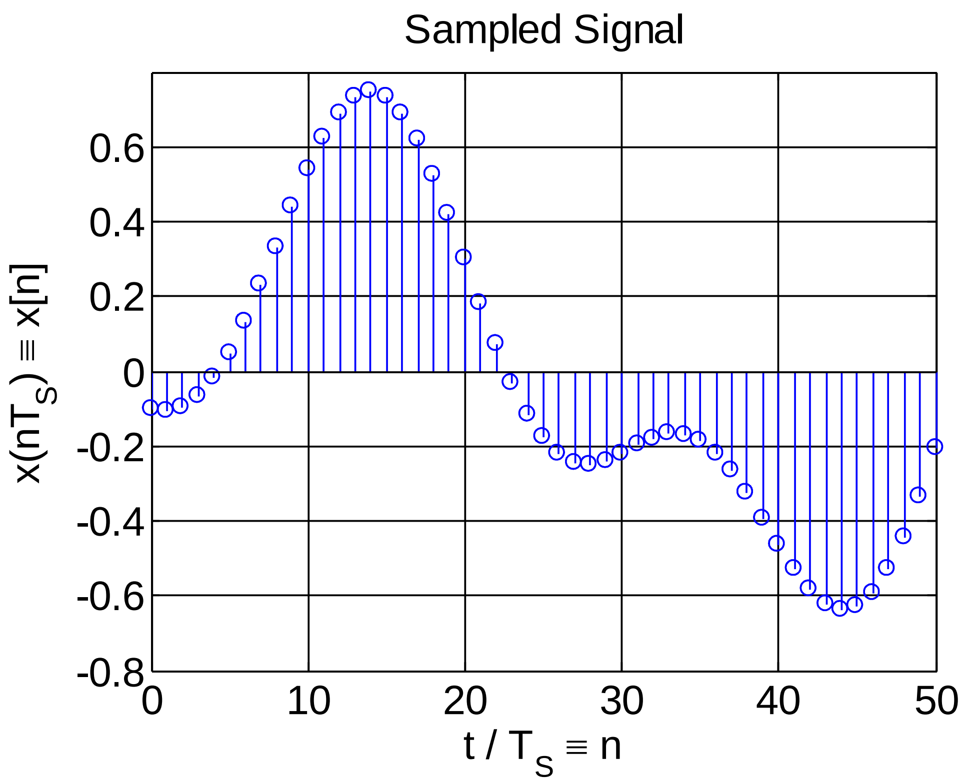
Dr. Christian Münker

Überblick: Signaltypen



Zeitdiskretes Signal oder System

- Abgetastet in regelmäßigen Abständen T_S
- Der *n*-te Abtastwert repräsentiert Zeitpunkt nT_S:
  $x[n] \equiv x(nT_S)$
- Zwischen Abtastwerten ist Signal bzw. Systemantwort nicht definiert

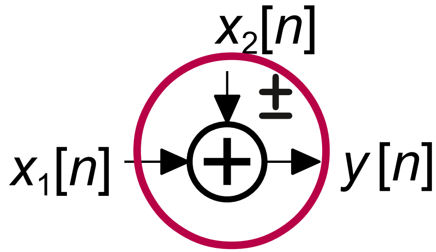


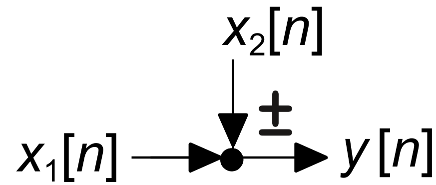
Wir nehmen zunächst an, dass das Signal „so oft“ abgetastet wurde, dass keine „wesentliche“ Information beim Abtasten verloren ging.

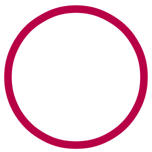
→ Nyquist, Kap. 7

Addition / Subtraktion

$$y[n] = x_1[n] \pm x_2[n]$$







: In dieser Vorlesung bevorzugte Schreibweise

Signale entsprechen Bussen (nicht gesondert gekennzeichnet)

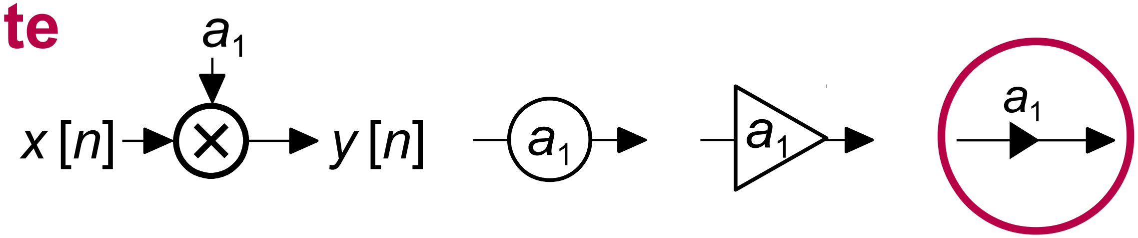
FPGA: Aus Gattern bzw. Look-Up Tables zusammengebaut (kombinatorische Logik), Fläche (= Kosten) nimmt mit Wortbreite zu

uC / DSP: Assemblerbefehle für Addition, nur fixe Wortbreiten (z.B. 8 / 16 / 32 Bit) möglich

Grundelemente zeitdiskreter Systeme (2)

Multiplikation mit Konstante

$$y[n] = a_1 x[n]$$



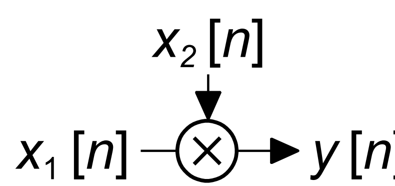
FPGA: Aus Addierern und Logik zusammengestellt, großer Resourcenverbrauch, viele FPGAs haben spezielle kompakte und schnelle Multiplizierer / Addierer („MAC-Units“)

uC: bei leistungsfähigen Modellen Befehle für schnelle Multiplikation

DSP: spezielle Befehle für kombinierte schnelle Multiplikation und Addition

Besonders „teuer“: Allgemeiner Multiplizierer

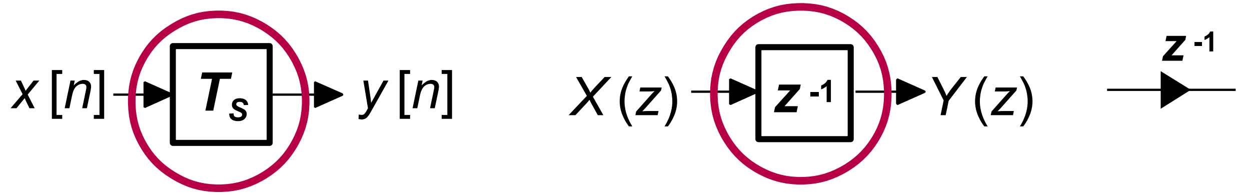
- selten gebraucht, z.B. für Leistungsberechnung
- nicht-lineare Operation!



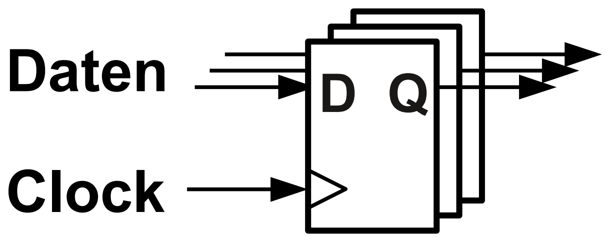
Grundelemente zeitdiskreter Systeme (3)

Einheitsverzögerung

$y[n] = x[n\text{-}1]$



In abgetasteten Systemen werden Daten mit jedem Takt einen „Platz“ weitergeschoben, diese Taktfrequenz ist ein wichtiger Designparameter.



Datenworte benötigten natürlich für jedes Bit ein eigenes FlipFlop, zusammen nennt man die FlipFlops *Register.*

Register können bei Hard- oder Software-Implementierungen z.B. durch Pointer auf einen Speicherbereich oder Ringbuffer implementiert und adressiert werden, speziell bei Hardware-Implementierungen auch Schiebereregister.

FPGAs enthalten einzelne FlipFlops im Fabric sowie spezielle Block RAMs, bei Bedarf kann auch (langsameres) externes RAM verwendet werden.

Digitale Signalverarbeitung auf FPGAs

Kap. 1 Zeitdiskrete Signale und LTI-Systeme im Zeitbereich

Teil 2 *Impulsantwort und zeitdiskrete Faltung*

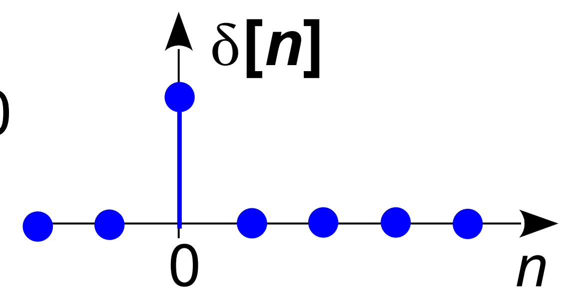
2016

Dr. Christian Münker

Impulsantwort zeitdiskreter Systeme

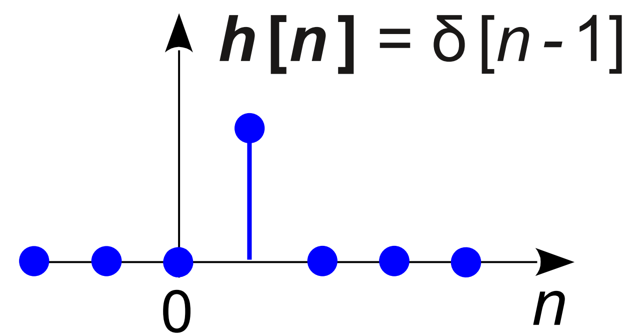
- Systeme lassen sich im Zeitbereich durch ihre *Impulsantwort h* charakterisieren, die sich durch Anregung mit Dirac-Impuls ergibt:
  - Zeitkontinuierliches System: $x(t) = \boldsymbol{\delta(t)} \rightarrow y(t) = h(t)$
  - Zeitdiskretes System: $x[n] = \boldsymbol{\delta[n]} \rightarrow y[n] = h[n]$

- Zeitdiskreter Dirac-Impuls: $\delta[n] = \begin{cases} 1 & \text{für } n = 0 \\ 0 & \text{sonst} \end{cases}$



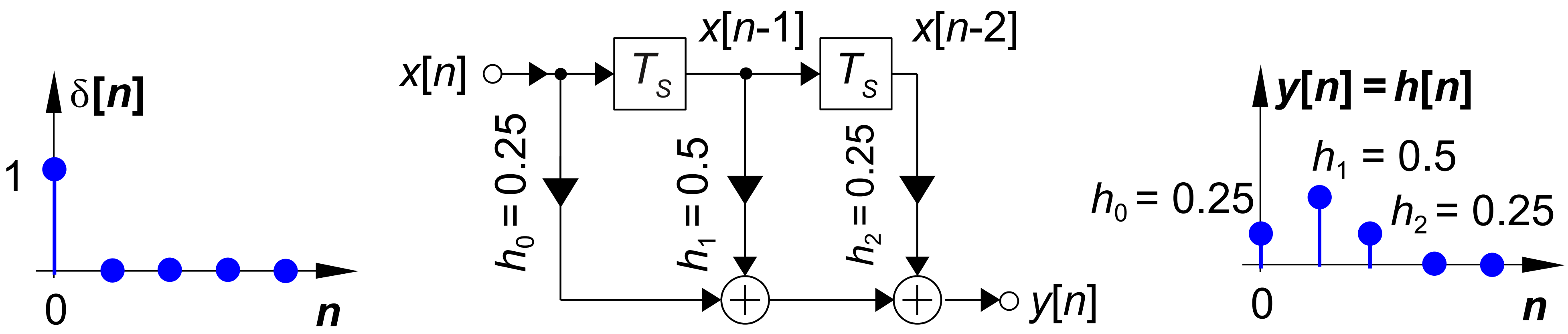
- Beispiel: Impulsantwort der Einheitsverzögerung:

$$x[n] = \delta[n] \rightarrow \boxed{T_s} \rightarrow y[n] = h[n]$$



Impulsantwort eines einfachen Filters

$x[n] = \delta[n] \rightarrow$ Ausgang ist Impulsantwort, $y[n] = h[n]$



Signalflussgraph (SFG)

Impulsantwort: $h[n] = h_0\,\delta[n] + h_1\,\delta[n\text{-}1] + h_2\,\delta[n\text{-}2]$

Andere Schreibweise: $h[n] = \{h_0; h_1; h_2\}$

Bei diesem Filtertyp sind Koeffizientenwerte = Impulsantwort!

Allgemeine Antwort $y[n]$ eines LTI Systems H

- Ausgangsfolge $y[n]$ für $x[n] = \delta[n]$ ist Impulsantwort $h[n]$.

  Wie bestimme ich $y[n]$ für allgemeine Eingangsfolge $x[n]$?

➔ $x[n]$ kann dargestellt werden als Summe von gewichteten und verzögerten Einheitsimpulsen, $x[n] = \sum x_i \delta[n - i]$

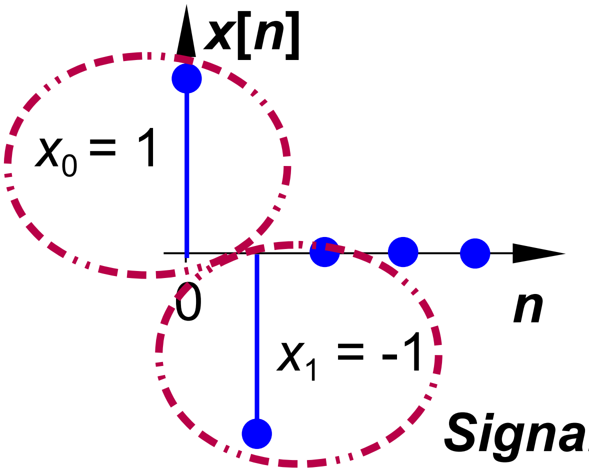
➔ Wegen LTI – Eigenschaften von H kann Ausgangsfolge $y[n]$ bestimmt werden als Überlagerung von gewichteten („L“) und verzögerten („T“) Impulsantworten (*diskrete Faltung*):

$$y[n] = \sum_{i=0}^{\infty} x_i h[n-i] = \sum_{i=0}^{\infty} x_i \delta[n-i] h[n-i] = \sum_{i=0}^{\infty} x[n] h[n-i] \equiv x[n] * h[n]$$

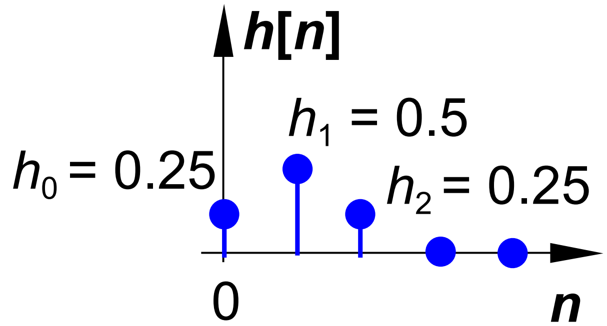
➔ $y[n]$ ist gewichtete Summe (= Filterung) der Eingangswerte!

Java Applet: http://www.jhu.edu/signals/ → Joy of Convolution

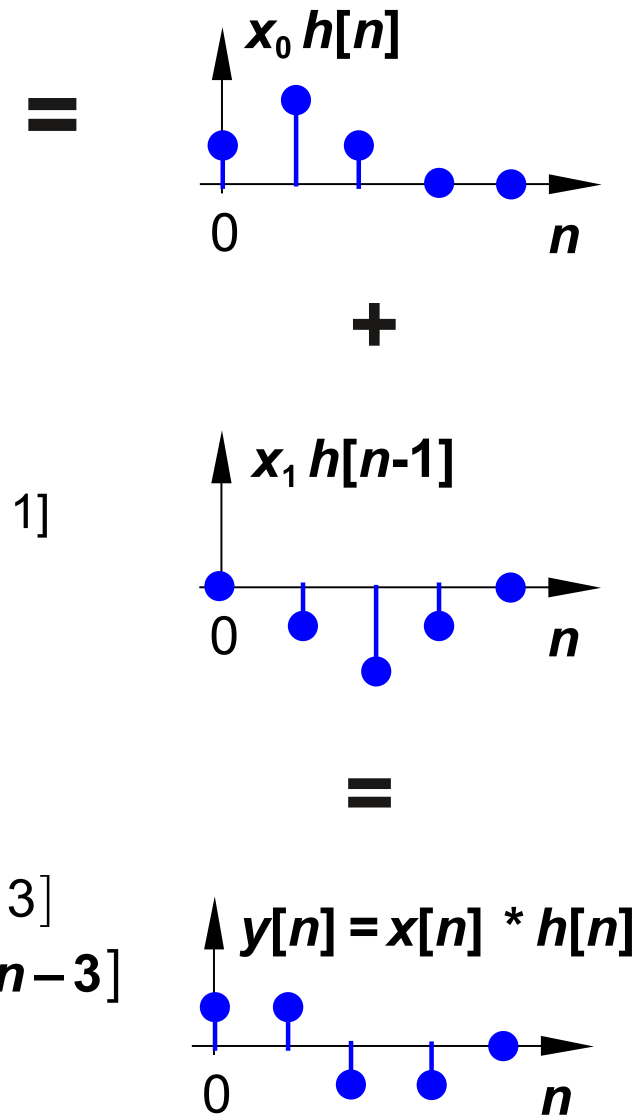
Beispiel für Berechnung der Antwort mit Faltung



*



=



Signal

$\boldsymbol{x[n]} = \delta[n] - \delta[n-1]$

System

$\boldsymbol{h[n]} = 0.25\,\delta[n] + 0.5\,\delta[n-1] + 0{,}25\,\delta[n-2]$

$$\boldsymbol{y[n]} = x[n] * h[n] = \sum_{i=0}^{\infty} x[n]h[n-i] = \sum_{i=0}^{3} x[n]h[n-i]$$

$$= 0.25\,\delta[n] + 0.5\,\delta[n-1] + 0.25\,\delta[n-2]$$

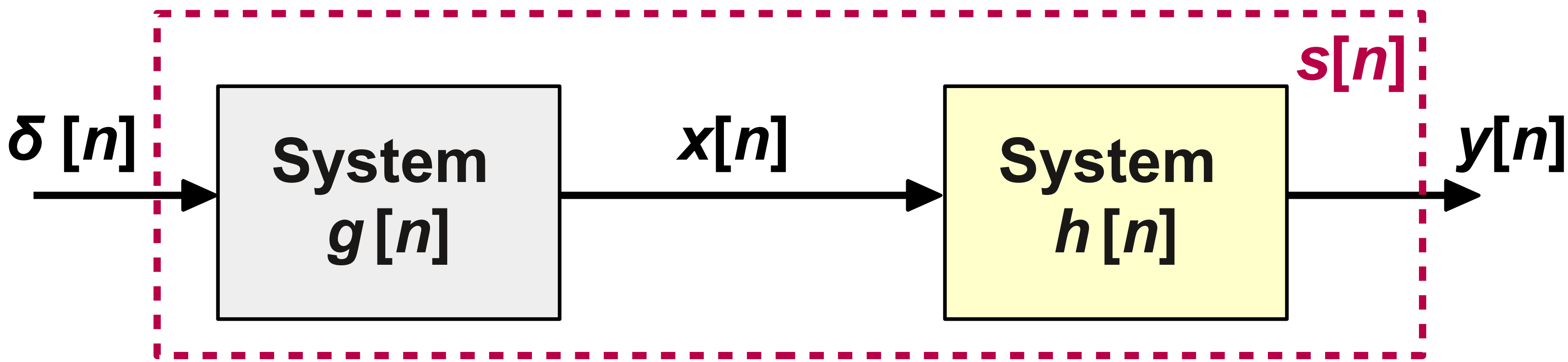
$$- 0.25\,\delta[n-1] - 0.5\,\delta[n-2] - 0.25\,\delta[n-3]$$

$$= \mathbf{0.25}\,\delta[\boldsymbol{n}] + \mathbf{0.25}\,\delta[\boldsymbol{n-1}] - \mathbf{0.25}\,\delta[\boldsymbol{n-2}] - \mathbf{0.25}\,\delta[\boldsymbol{n-3}]$$

Länge: $L_y = L_x + L_h - 1 = 2 + 3 - 1 = 4$

Impulsantwort kaskadierter Systeme

Wie lautet Gesamtimpulsantwort $s[n]$ von zwei kaskadierten Systemen?



$\mathbf{g[n]} = \delta[n] - \delta[n-1]$

$\mathbf{h[n]} = 0.25\,\delta[n] + 0.5\,\delta[n-1] + 0.25\,\delta[n-2]$

Rechnung über Faltung wie auf voriger Folie:

$$s[n] = y[n] = x[n] * h[n] = g[n] * h[n]$$

Gesamtimpulsantwort entspricht der Faltung der Einzelimpulsantworten!

Impulsantwort kaskadierter Systeme (2)

Berechnung der Ausgangsfolge für beliebige Eingangssignale über zeitdiskrete Faltung ist bei komplexeren Systemen aufwändig.

Wir werden sehen, dass Darstellung und Rechnung in der komplexen Frequenzebene vorteilhafter ist.

Digitale Signalverarbeitung auf FPGAs

Kap. 1 Zeitdiskrete Signale und LTI-Systeme im Zeitbereich

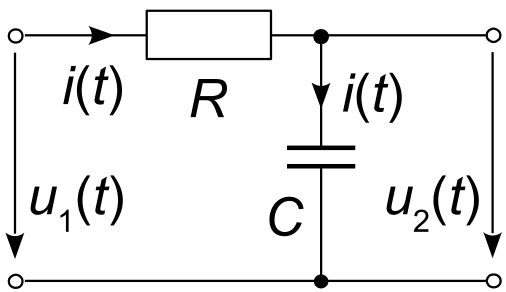
Teil 3 *Einführung in die z-Transformation*

2016

Dr. Christian Münker

Wdh.: Zeit**kontinuierliche** LTI - Systeme (1)

Das Verhalten zeitkontinuierlicher Systeme wird durch *Differenzialgleichungen* (DGL) beschrieben, z.B.



$$u_1(t) = u_R(t) + u_2(t) = i(t)R + u_2(t) = C\frac{\mathrm{d}u_2(t)}{\mathrm{d}t}R + u_2(t)$$

$$\Rightarrow \quad \frac{\mathrm{d}u_2(t)}{\mathrm{d}t} + \frac{u_2(t)}{RC} = \frac{u_1(t)}{RC} \quad \text{mit} \quad i(t) = C\,\mathrm{d}u_2(t)$$

$$\Rightarrow \quad h(t) = ? \qquad u_2(t) = ?$$

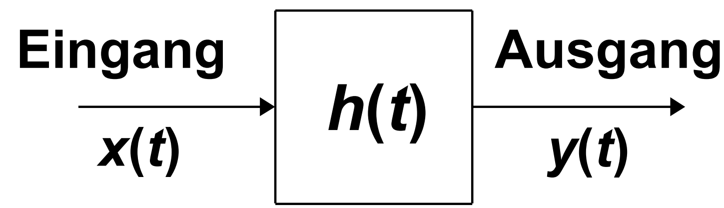
Nur für *lineare, zeitinvariante Systeme* (linear time-invariant, LTI) mit konstanten Koeffizienten sind leistungsfähige mathematische Methoden wie Fourier- und Laplacetransformation definiert, z.B.:

$$\circ\!-\!\bullet \quad s\,U_2(s) + \frac{U_2(s)}{RC} = \frac{U_1(s)}{RC} \quad \Rightarrow \quad \boldsymbol{H(s) = \frac{U_2(s)}{U_1(s)} = \frac{1}{1+sRC}}$$

$$\bullet\!-\!\circ \quad \boldsymbol{h(t) = 1 - \mathrm{e}^{-t/RC}}$$

Wdh.: Zeit**kontinuierliche** LTI - Systeme (2)

LTI = linear time-invariant $y(t) = h\{x(t)\}$:



Linear: $x(t) = a \cdot x_1(t) + b \cdot x_2(t) \rightarrow y(t) = a \cdot y_1(t) + b \cdot y_2(t)$

Gegenbeispiele: Transistor, Quantisierer, Verstärker in Begrenzung

Zeitinvariant: $x(t - t_1) \rightarrow y(t - t_1)$

Gegenbeispiel: Abtaster, automatische Verstärkungsreglung

Kausal: $y(t = t_1)$ ist unabhängig von $x(t \geq t_1)$:

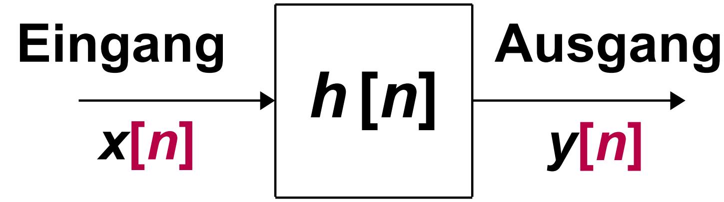
Wirkung folgt Ursache: Für $x(t) = \delta(t)$ muss $y(t < 0) = 0$ sein.

Gegenbeispiel: rect(f) ●-○ si(t) erstreckt sich von $-\infty < t < +\infty$

Kausalität ist mathematisch nicht notwendig, aber akausale Systeme lassen sich technisch nicht realisieren!

Zeit**diskrete** LTI-Systeme (1)

LTI = linear time-invariant $y[n] = h\{\, x[n] \,\}$:



Linear: $x[n] = a \cdot x_1[n] + b \cdot x_2[n] \rightarrow y[n] = a \cdot y_1[n] + b \cdot y_2[n]$

Gegenbeispiele: Multiplizierer, Überlauf- und Sättigungseffekte!

Zeitinvariant: $x[n - n_1] \rightarrow y[n - n_1]$

Gegenbeispiele: Dezimator, adaptives Filter, Downsampling

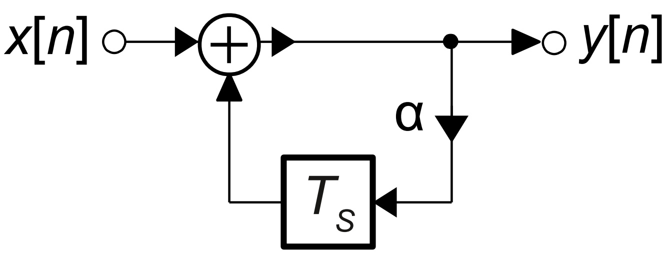
Kausal: $y[n = n_1]$ ist unabhängig von $x[n \geq n_1]$

Gegenbeispiele: rect(F) ●-○ si[n], $y[n] = x[n+1] - x[n]$ (forward difference)

Viele akausale zeitdiskrete Systeme lassen sich „offline“ d.h. mit gespeicherten Werten berechnen, da man hier ja auf “zukünftige” Werte zugreifen kann.

Zeit**diskrete** LTI-Systeme (2)

Das Verhalten zeitdiskreter Systeme wird durch *Differenzengleichungen* (DZGL) beschrieben, z.B.



$$y[n] = x[n] + \alpha y[n-1]$$

$$\Rightarrow \quad y[n] - \alpha y[n-1] = x[n]$$

$$\Rightarrow \quad y[n] = ? \qquad h[n] = ?$$

Auch hier: nur LTI-Systeme mit konstanten Koeffizienten lassen sich mit leistungsfähigen mathematischen Methoden wie Fourier- und Laplace- behandeln.

Für zeitdiskrete Systeme gibt es die ***z*-Transformation** als Spezialfall der Laplace-Transformation für äquidistant abgetastete Systeme.

Signal-Transformationen (1)

Transformationen **vereinfachen** die Analyse und Manipulation von Signalen und (linearen) Systemen in Nachrichtentechnik und DSP:

- Praktische Analysen im Zeitbereich* sind oft kompliziert (Differential- und Integralgleichungen)
- Im Frequenzbereich lassen sich Analysen und Manipulationen einfacher (z.B. Multiplikation statt Integral / Faltung) durchführen

Zeitbereich
(kontinuierlich oder diskret)

Transformation →

← Inverse Transformation

Frequenzbereich
(Fourier / Laplace / z - Transformierte)

* Das können verallgemeinert auch räumliche Koordinaten (Bildverarbeitung) sein

Signal-Transformationen (2)

Fourier-Transformation: Darstellung eines Signals / Systems als Summe von Sinusschwingungen → Information über die spektrale Zusammensetzung / Gewichtung (physikalische Frequenz ***f***)

Laplace-Transformation: Darstellung eines Signals / Systems als Summe von exponentiell gedämpften Sinusschwingungen → Information über Einschwingverhalten und Stabilität

***z*-Transformation:** Laplace Transformation für zeitdiskrete Signale – *lineare Differenzengleichungen mit konst. Koeffizienten* statt *Differenzial*gleichungen

Generell: Differential, Integral- und vor allem Faltungs-Operationen werden auf Multiplikationen und Divisionen abgebildet! Vgl. logarithmische Berechnung der Multiplikation.

Signal-Transformationen (3)

Fourier $F\{\bullet\}$: $$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \Leftrightarrow x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} d\omega$$

Laplace $L\{\bullet\}$: $$X(s) = \int_{-\infty}^{\infty} x(t) e^{-st} dt \Leftrightarrow x(t) = \frac{1}{j2\pi} \int_{\sigma - j\infty}^{\sigma + j\infty} X(s) e^{st} ds$$

z-Transf. $\mathcal{Z}\{\bullet\}$: $$X(z) = \sum_{k=0}^{\infty} x[k] z^{-k} \Leftrightarrow x[k] = \frac{1}{j2\pi} \oint_C \frac{X(z) z^k}{z} dz$$

mit $\omega = 2\pi f$, $s = \sigma + j\omega$ und $z = e^{sT_S}$

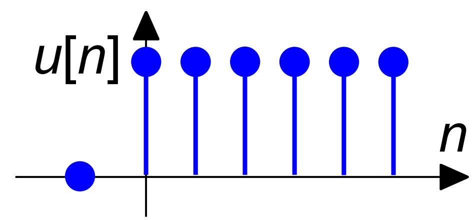
Aus $X(s)$ und $X(z)$ lässt sich leicht Frequenzgang des Systems bei physikalischen Frequenzen ermitteln für $s = j\omega$ und $z = e^{j\omega T_S}$

Beispiel: *z*-Transformierte von *u*[*n*]

Die einseitige *z*-Transformierte eines kausalen Signals *x*[*n*] ist

$$X(z) = \mathcal{Z}\{x[n]\} = \sum_{k=0}^{\infty} x[k] z^{-k} \qquad \overset{\shortparallel}{x[n] = 0 \text{ für } n < 0}$$

- *z*-Transformierte des Einheitssprungs



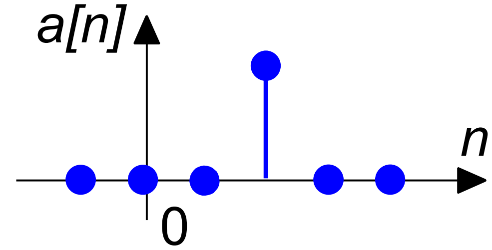
$$U(z) = \mathcal{Z}\{u[n]\} = \sum_{n=0}^{\infty} u[n] z^{-n} = \sum_{n=0}^{\infty} z^{-n} = 1 + z^{-1} + z^{-2} + \ldots = \frac{1}{1 - z^{-1}} \text{ für } |z| < 1$$

(über Formel für unendliche geometrische Reihe)

- Welches einfache System hat *u*[*n*] als Impulsantwort? Zeichnen Sie den SFG auf!

Beispiel: *z*-Transformierte von δ[*n*-2]

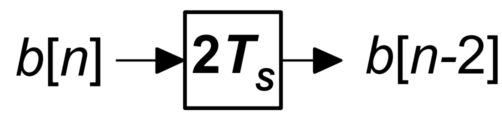
z-Transformierte eines um zwei Samples verzögerten Einheitsimpulses, *a*[*n*] = δ[*n*-2]:



$$A(z) = \mathcal{Z}\{a[n]\} = \sum_{n=0}^{\infty} a[n] z^{-n} = \sum_{n=0}^{\infty} \delta[n-2] z^{-n}$$

$$= 0 + 0 + z^{-2} + 0 + \ldots = \boldsymbol{z^{-2}}$$

z-Transformierte eines um zwei Samples verzögerten beliebigen Signals *b*[*n*-2]:



$$\mathcal{Z}\{b[n]\} = \sum_{n=0}^{\infty} b[n] z^{-n} \equiv B(z)$$

$$\Rightarrow \mathcal{Z}\{b[n-2]\} = \sum_{n=0}^{\infty} b[n-2] z^{-n} = \sum_{n=0}^{\infty} b[n] z^{-n} z^{-2} = \boxed{B(z) z^{-2}}$$

Beispiel: *z*-Transformierte von δ[*n* - *k*]

Ähnlich wie auf der letzten Folie kann man leicht allgemein zeigen:

$$b[n] \circ\!\!-\!\!\bullet B(z) \quad \Rightarrow \quad b[n-k] \circ\!\!-\!\!\bullet B(z)z^{-k}$$

Damit erhält man die *z*-Transformierte allgemeiner Signale und Impulsantworten, z.B.

$$x[n] = 4\delta[n] + 3\delta[n-1] - 2\delta[n-2] \circ\!\!-\!\!\bullet 4 + 3z^{-1} - 2z^{-2} = X(z)$$

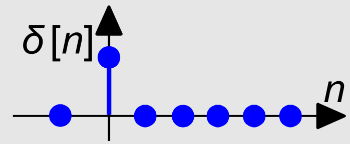
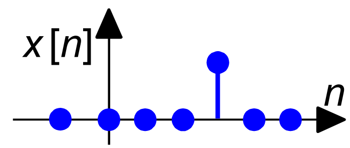
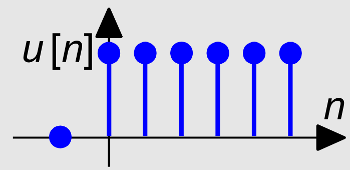
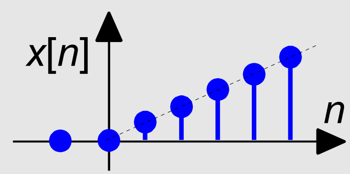
oder allgemein:

$$a[n] = \sum_{k=0}^{\infty} a_k \delta[n-k] \circ\!\!-\!\!\bullet A(z) = \sum_{k=0}^{\infty} a_k z^{-k}$$

Eigenschaften der z-Transformation

Eigenschaft	Zeitbereich $x[n]$	z-Transformierte $X(z)$
Linearität	$x[n] + y[n]$	$X(z) + Y(z)$
Zeitverschiebung	$x[n-k],\ k \in \mathbb{Z}$	$X(z)\ z^{-k}$
Frequenzverschiebung (Modulation)	$x[n]\ e^{-j\omega n},\ \omega \in \mathbb{R}$	$X(z\ e^{j\omega})$
= Skalierung in der z-Ebene	$x[n]\ \alpha^{n},\ \alpha \in \mathbb{R}$	$X(z / \alpha)$
Zeitumkehr	$x[-n]$	$X(z^{-1})$
Konjugiert komplexe Folge	$x^*[n]$	$X^*(z^*)$
Faltung im Zeitbereich	$x[n] \ast y[n]$	$X(z) \cdot Y(z)$

z-Transformierte wichtiger Signale / Systeme

Signal bzw. System	Zeitbereich $x[n]$ bzw. $h[n]$		z-Transformierte $X(z)$ bzw. $H(z)$
Dirac-Stoß	$\delta[n]$		1
Verzögerter Dirac-Stoß / Verzögerung	$\delta[n-m]$		z^{-m}
Einheitssprung / Integrator	$u[n]$		$\frac{z}{z-1} = \frac{1}{1-z^{-1}}$ für $\lvert z\rvert>1$
Exponenzialfunktion / Verlustbehafteter Integrator	$a^{n-1}u[n], \lvert a\rvert < 1 \in \mathbb{R}$		$\frac{z}{z-a} = \frac{1}{1-az^{-1}}$ für $\lvert z\rvert>\lvert a\rvert$
Rampe / Zwei kaskadierte Integratoren	$nu[n]$		$\frac{z}{(z-1)^2} = \frac{z^{-1}}{(1-z^{-1})^2}$

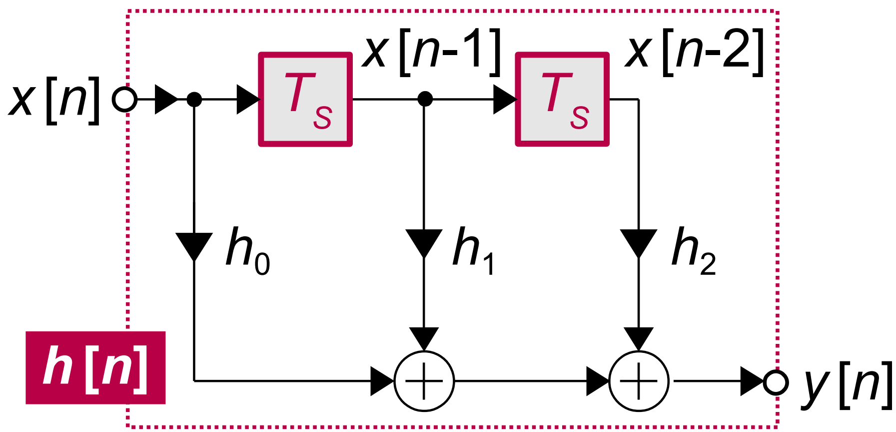
Zeichnen Sie den SFG der Systeme auf!

Signalflussgraph (SFG) in der *z*-Ebene

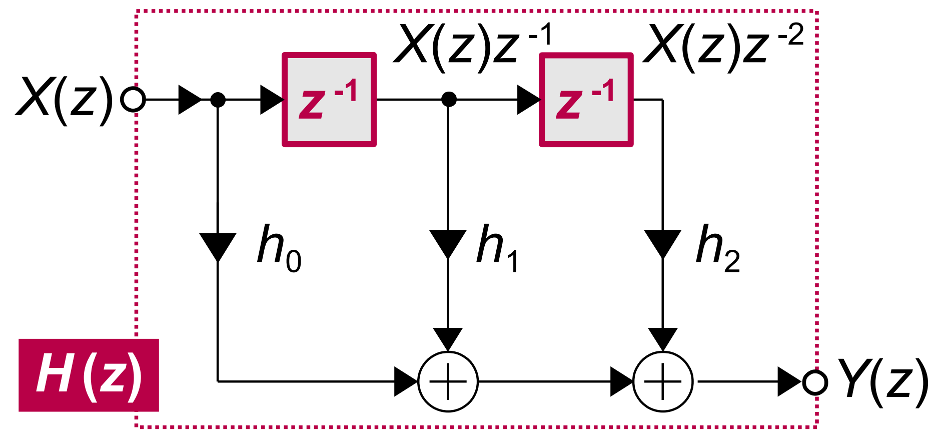
Transformiere Differenzengleichung (DZGL) in die *z*-Ebene:

$$\boldsymbol{y[n]} = h_0\, x[n] + h_1\, x[n-1] + h_2\, x[n-2] \quad \circ\!\!-\!\!\bullet \quad \boldsymbol{Y(z)} = h_0 X(z) + h_1 X(z) z^{-1} + h_2 X(z) z^{-2}$$

Zeichne SFG aus DZGL
bzw. aus *z*-Transformierter:



$\circ\!\!-\!\!\bullet$



SFGs von **LTI-Systemen** haben in der Zeitebene und in der *z*-Ebene **gleiche Topologie:**
Verzögerungen um T_S entsprechen Multiplikation mit z^{-1}!

SFG in der z-Ebene - Systemfunktion

Aus der z-Transformierten des Systems auf der letzten Folie

$$Y(z) = h_0 X(z) + h_1 X(z) z^{-1} + h_2 X(z) z^{-2} \Rightarrow \boldsymbol{H(z) = \frac{Y(z)}{X(z)}} = h_0 + h_1 z^{-1} + h_2 z^{-2}$$

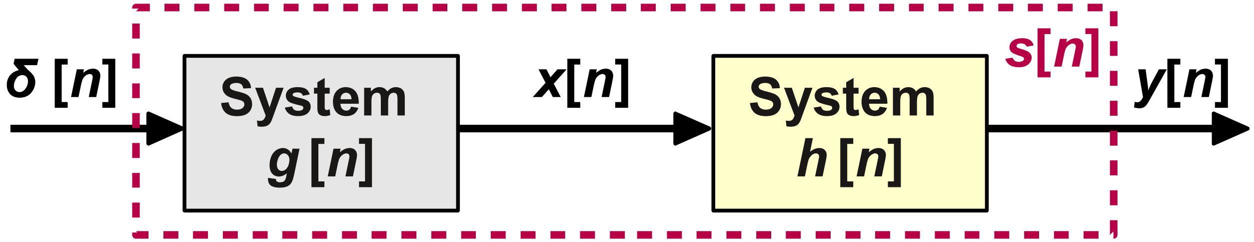
kann man die *Übertragungs*- oder *Systemfunktion* $H(z)$ bestimmen. $H(z)$ erlaubt u.a. die Ermittlung

- der Impulsantwort $h[n]$ durch inverse z-Transformation, wenn direkte Bestimmung wie bei rekursiven Systemen nicht möglich ist,
- des Frequenzgangs $H(e^{j\Omega})$ (→ Kap. 2),
- und der Stabilität des Systems (→ Kap. 2).

> Zeitdiskrete LTI-Systeme werden daher meist über die Übertragungsfunktion $H(z)$ angegeben!

Kaskadierte Systeme in der *z*-Ebene

Zeitbereich:
Gesamtimpulsantwort
s[*n*] über Faltung

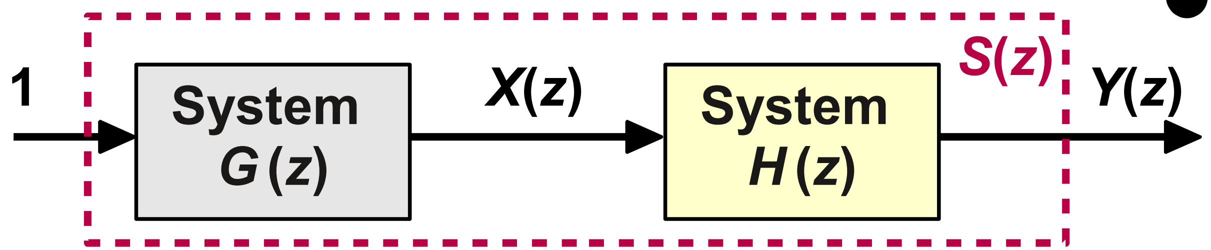


$\mathbf{g[n]} = \delta[n] - \delta[n-1]$

$\mathbf{h[n]} = 0.25\,\delta[n] + 0.5\,\delta[n-1] + 0.25\,\delta[n-2]$

$$\mathbf{s[n]} = y[n] = x[n] * h[n] = g[n] * h[n]$$
$$= \mathbf{0.25\,\delta[n] + 0.25\,\delta[n-1] - 0.25\,\delta[n-2] - 0.25\,\delta[n-3]}$$

***z*-Ebene:**
Gesamtsystemfunktion
S(*z*) über Multiplikation



$\mathbf{G(z)} = 1 - z^{-1}$

$\mathbf{H(z)} = 0.25 + 0.5\,z^{-1} + 0.25\,z^{-2}$

$$\mathbf{S(z)} = Y(z) = X(z)\,H(z) = G(z)\,H(z) = \mathbf{0.25 + 0.25\,z^{-1} - 0.25\,z^{-2} - 0.25\,z^{-3}}$$

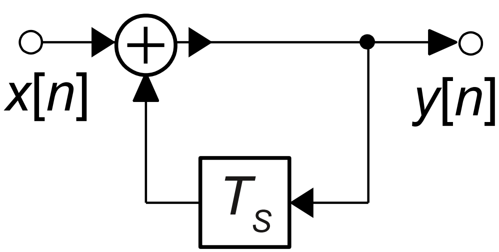
Integrator / Akkumulator reloaded

Können Sie aus der Systemfunktion $H_I(z)$ den Signalflussgraphen des Integrators ableiten?

$$H_I(z) = \frac{Y(z)}{X(z)} = \frac{1}{1-z^{-1}}$$

$$\Leftrightarrow \quad Y(z)\left(1-z^{-1}\right) = X(z) \quad \bullet\!\!-\!\!\circ \quad y[n]-y[n-1] = x[n]$$

$$\Leftrightarrow \quad \boldsymbol{y[n] = x[n] + y[n-1]}$$

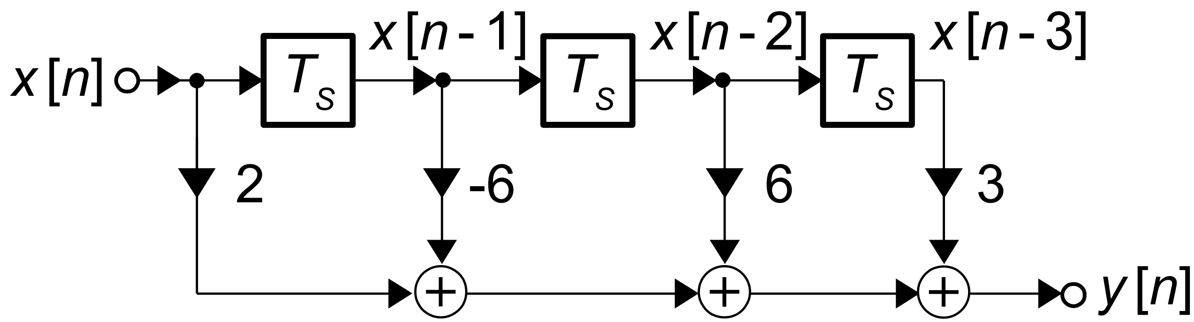


Darstellungen zeitdiskreter Systeme

Zeitdiskrete Systeme lassen sich auf verschiedene Arten beschreiben (s.u.). Bevorzugt wird meist die *Systemfunktion* $H(z)$, da sie leicht manipuliert (z.B. Multiplikation statt Faltung) und analysiert werden kann.

Bei transversalen Systemen kann man die Formen leicht in einander überführen:

Grafisch / SFG:



Differenzengleichung: $y[n] = 2\,x[n] \;-6\,x[n-1] \;+6\,x[n-2] \;+3\,x[n-3]$

Systemfunktion: $Y(z) = 2\,X(z) \;-6\,X(z)\,z^{-1} \;+6\,X(z)\,z^{-2} \;+3\,X(z)\,z^{-3}$

$$\Rightarrow \boldsymbol{H(z) = Y(z)\,/\,X(z) = 2 \quad -6\,z^{-1} \quad +6\,z^{-2} \quad +3\,z^{-3}}$$

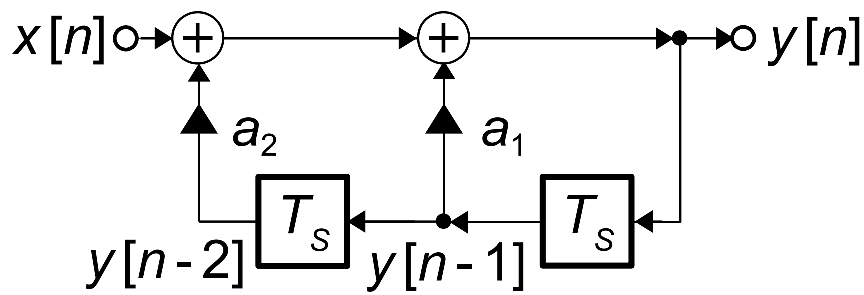
Impulsantwort: $\boldsymbol{h[n]} = 2\,\delta[n] \;-6\,\delta[n-1] \;+6\,\delta[n-2] \;+3\,\delta[n-3]$

$$= \{2; \quad -6; \quad +6; \quad +3\}$$

Zeitdiskrete **rekursive** Systeme

SFG, Differenzengleichung und Systemfunktion $H(z)$ lassen sich auch für rekursive Systeme (relativ) leicht in einander überführen:

Grafisch / SFG:



Differenzengleichung: $y[n] = x[n] \quad + a_1 y[n-1] \quad + a_2 y[n-2]$

Systemfunktion: $Y(z) = X(z) \quad + a_1 Y(z) z^{-1} \quad + a_2 Y(z) z^{-2}$

$$\Rightarrow H(z) = Y(z) / X(z) = 1 / (1 - a_1 z^{-1} \quad - a_2 z^{-2})$$

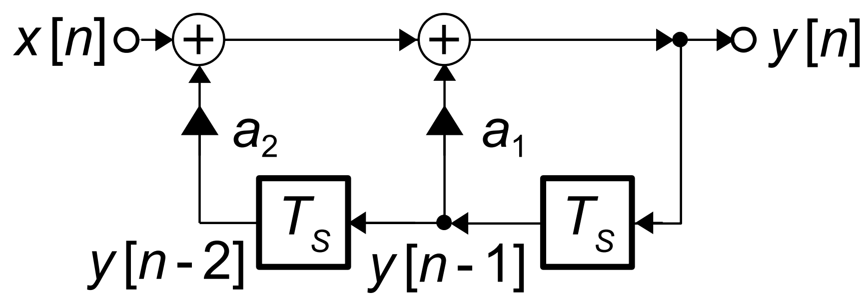
Impulsantwort: $h[n] = \delta[n] + a_1 \delta[n-1] + (a_1 a_2 + a_1^2) \delta[n-2] \;+ \ldots$

$= ?$

h[*n*] lässt sich bei rek. Systemen meist nicht in geschlossener Form ablesen!

Impulsantwort zeitdiskreter rekursiver Systeme

Um die Impulsantwort eines rekursiven Systems zu ermitteln, kann man:



- Eine geschlossene Darstellung für die entstehende endliche Reihe finden (nur für einfache Systeme sinnvoll)
- Aus der Systemfunktion $H(z)$ die inverse z-Transformation berechnen (aber wer mag schon Kontourintegrale ...)
- Mit Hilfe von Tabellen die inverse z-Transformation zu $H(z)$ finden, u.U. nach einer Partialbruchzerlegung
- Computer Algebra Systems (Mathematica, Pythons simpy, …) verwenden, z.B. http://www.wolframalpha.com/input/?i=inverse+Z+transform+calculator

$$H(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} \quad ○\!\!-\!\!\bullet \quad h[n] = \frac{2^{-(n+1)}\left(\left(a_1 + \sqrt{a_1^2 + 4a_2}\right)^{n+1} - \left(a_1 - \sqrt{a_1^2 + 4a_2}\right)^{n+1}\right)}{\sqrt{\left(a_1^2 + 4a_2\right)}}$$

- Die algebraische Gleichung für $h[n]$ ist oft unübersichtlich (s.o.), daher gleich **Numeric** Analysis Systems (Pythons NumPy / SciPy, Matlab, …) verwenden

Mehr?

- Ein sehr guter, praxisnaher Text zur *z*-Transformation mit vielen Beispielen ist web.eecs.umich.edu/~aey/eecs206/lectures/zfer.pdf
- Wir werden zunächst nur Systeme betrachten, für die die einfachen Umformungen der letzten Folien genügen.

Zusammenfassung *z*-Transformation

- *z*-Transformation ist sehr nützliches Werkzeug zur einfacheren Analyse von zeitdiskreten Signalen und Systemen
- Verzögerung um kT_S ○–● z^{-k}
- Faltung ○–● Multiplikation
- Vor allem Rücktransformation (Impulsantwort) im Allgemeinen nicht trivial, dank MATLAB o.ä. heutzutage kein Problem mehr

Nächstes Kapitel:

- Stabilität, Pole und Nullstellen aus *z*-Transformation
- Berechnung der Frequenzantwort aus *z*-Transformation

Digitale Signalverarbeitung auf FPGAs

Kap. 1 Zeitdiskrete Signale und LTI-Systeme im Zeitbereich

Teil 4 *Implementierungen zeitdiskreter Systeme*

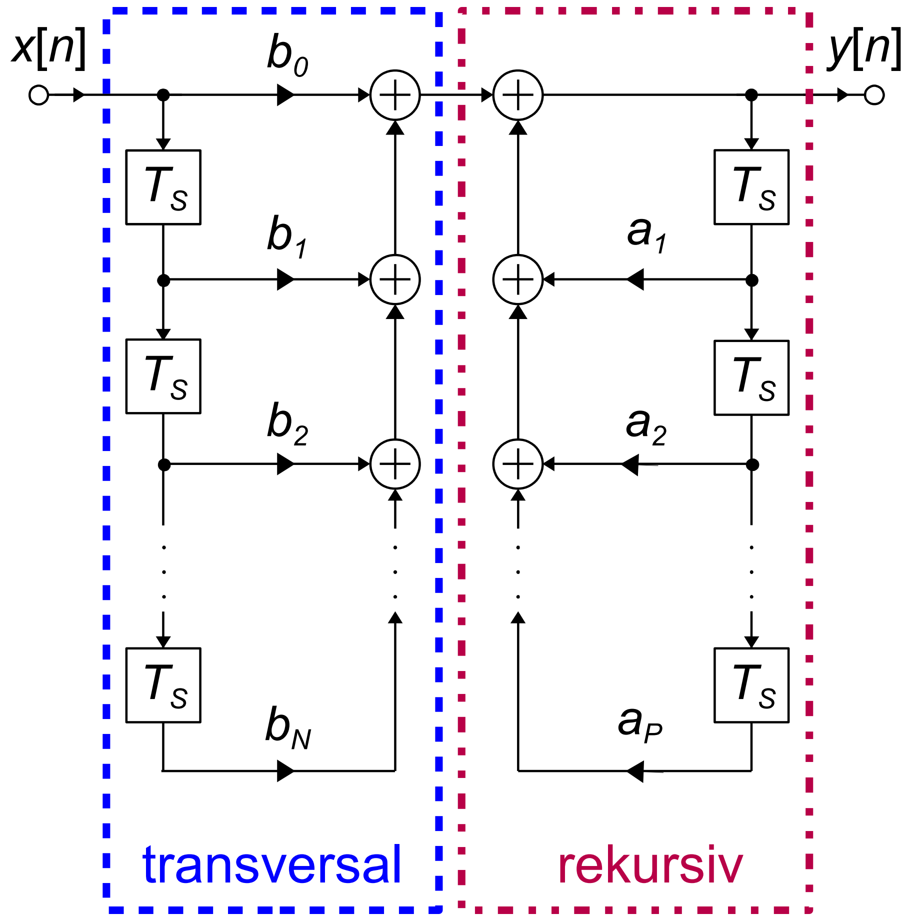
2016

Dr. Christian Münker

Direktform eines beliebigen LTI-Systems (1)

- **Jedes zeitdiskrete LTI-System** läßt sich durch folgende Differenzengleichung und folgenden SFG beschreiben:

$$y[n] = \boxed{\sum_{k=0}^{N} b_k x[n-k]} + \boxed{\sum_{i=1}^{P} a_i y[n-i]}$$

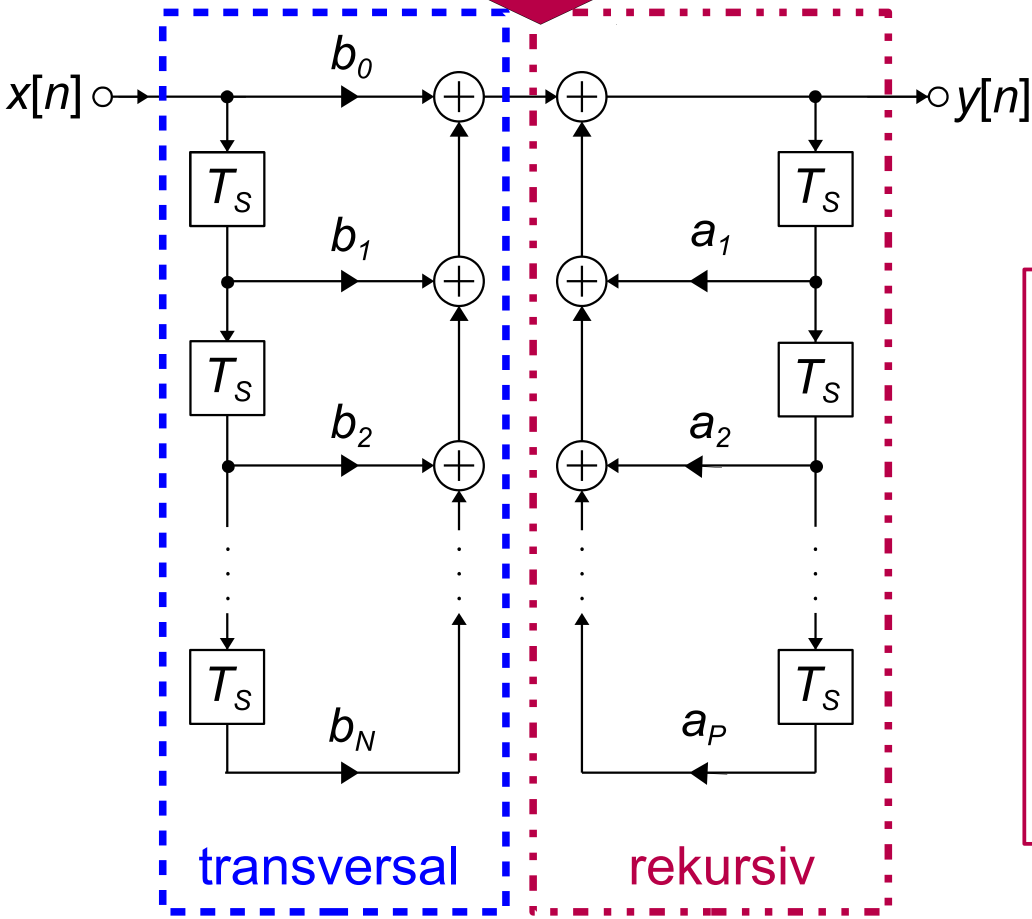


SFG, Differenzengleichung und $H(z)$ können *direkt* ineinander überführt werden: „*Direktform Typ* 1"

- Viele andere Topologien (SFGs) mit identischem $H(z)$ möglich
- Direktform ist aus numerischen Gründen nicht immer optimal (→ Kap. 5)

Direktform eines beliebigen LTI-Systems (2)

$$y[n] = \sum_{k=0}^{N} b_k x[n-k] + \sum_{i=1}^{P} a_i y[n-i] \quad \Leftrightarrow \quad Y(z) = \sum_{k=0}^{N} b_k X(z) z^{-k} + \sum_{i=1}^{P} a_i Y(z) z^{-i}$$



$$\Rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 - \sum_{i=1}^{P} a_i z^{-i}}$$

Differenzengleichung

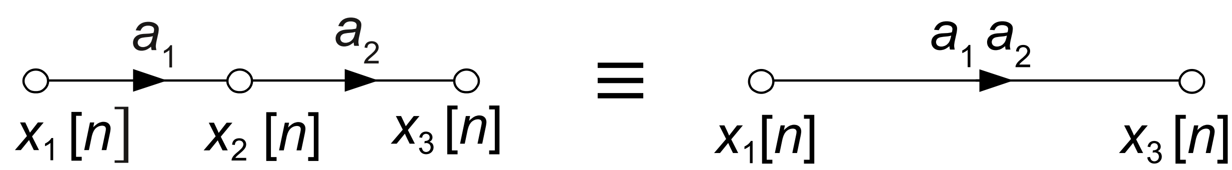
⇕

Übertragungsfunktion $H(z)$

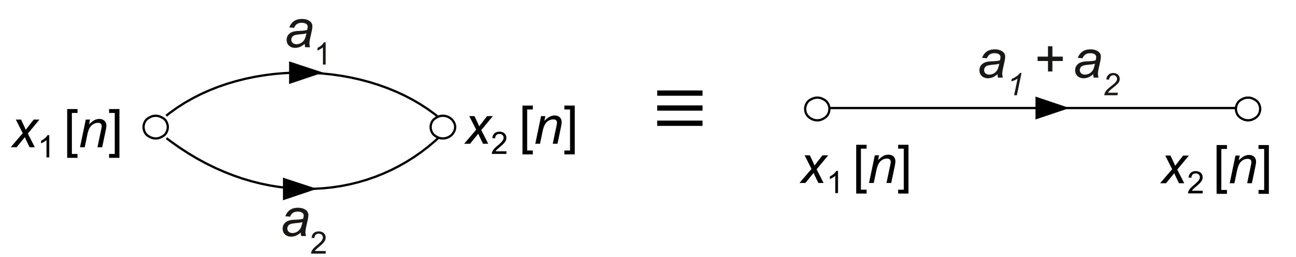
⇕

Direkte Hardware-Konstruktion

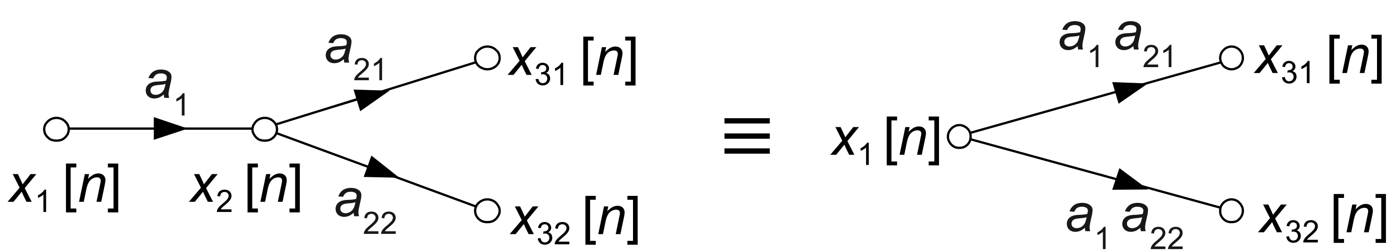
Umformen von LTI - Netzwerken (1)

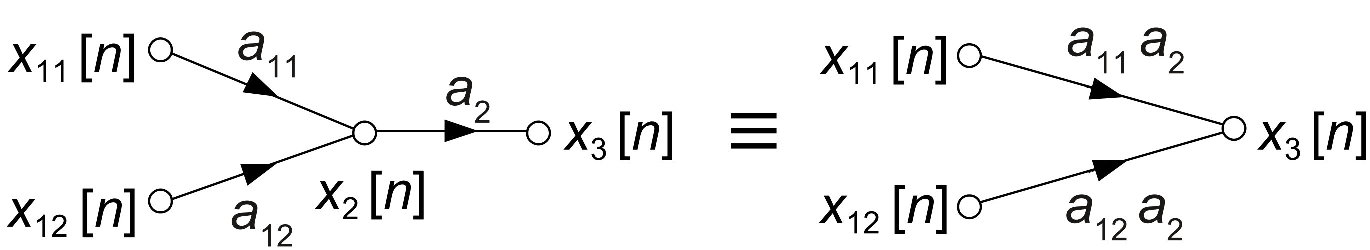


Zusammenfassen serieller Zweige



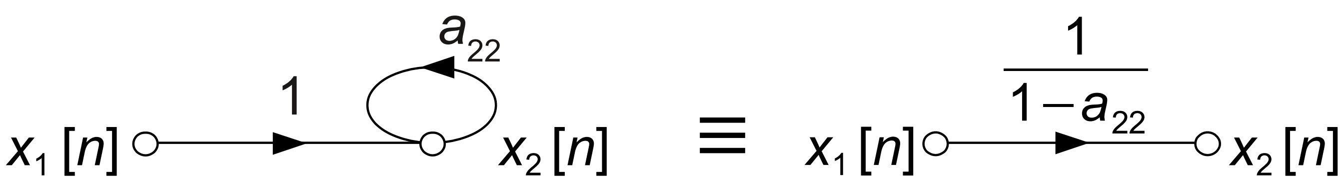
Zusammenfassen paralleler Zweige



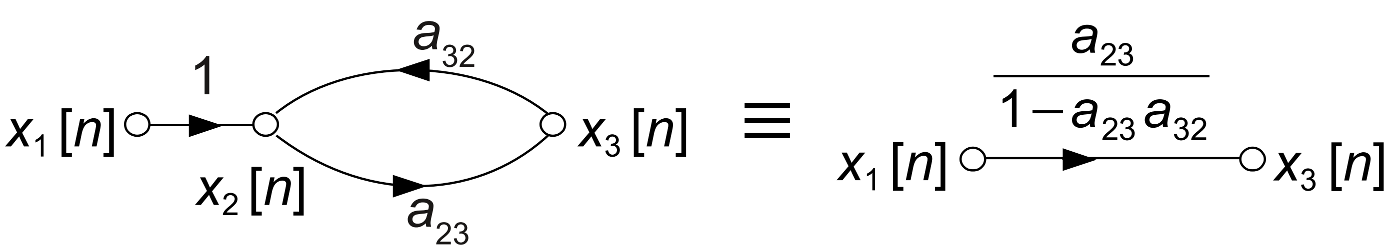


Distributivität („Schieben über Knoten“)

Umformen von LTI - Netzwerken (2)



Zusammenfassen einer Selbstschleife



Allg. Schleife

Transponieren von LTI – Netzwerken (1)

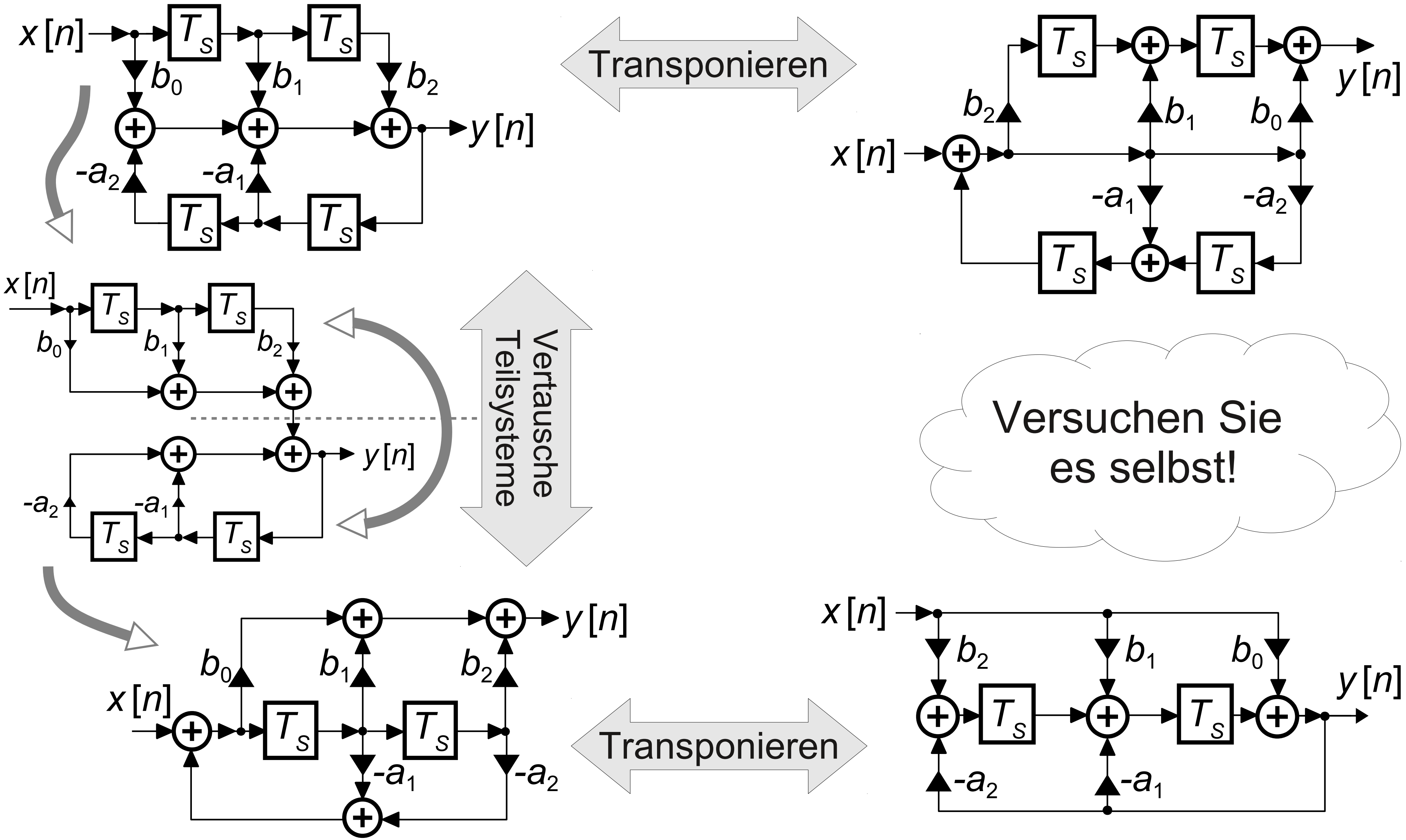
Transponieren (Umkehrung des Signalfluss-Graphen):

- Vertausche alle Signalflussrichtungen
- Ersetze Summierer durch Verzweigungsknoten und umgekehrt
- Vertausche Eingang und Ausgang

Beispiel auf nächster Folie selbst rechnen!

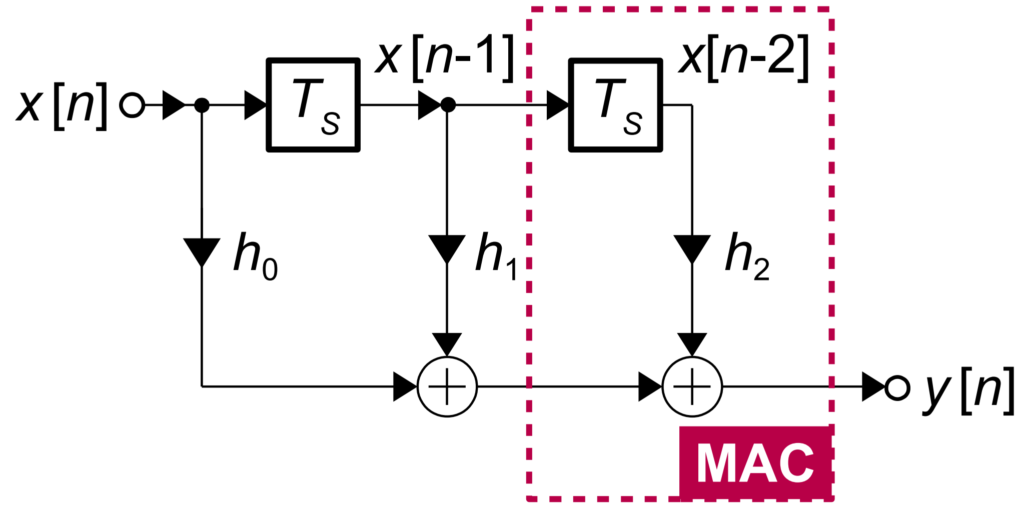
Umgeformte Systeme haben gleiche Impulsantwort und System-funktion, aber u.U. unterschiedliche Vor / Nachteile bei Implementierung!

Transponieren von LTI – Netzwerken (2)



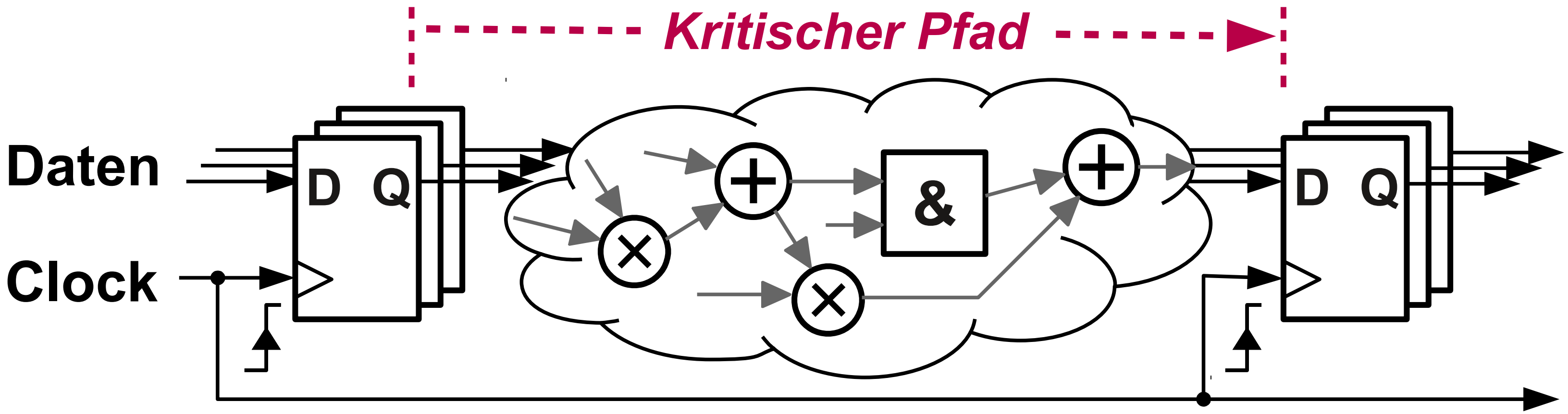
Multiply-Accumulate (MAC)

- Multiply-Add oder Multiply-Accumulate (MAC) ist Grundfunktion der digitalen Signalverarbeitung und kleinste „Währungseinheit“
- Auf vielen FPGAs und Prozessoren als optimierte Recheneinheit implementiert
- Angabe von MAC/Sample oder MAC/s ermöglicht Vergleich des Rechenaufwands für verschiedene Algorithmen oder Architekturen



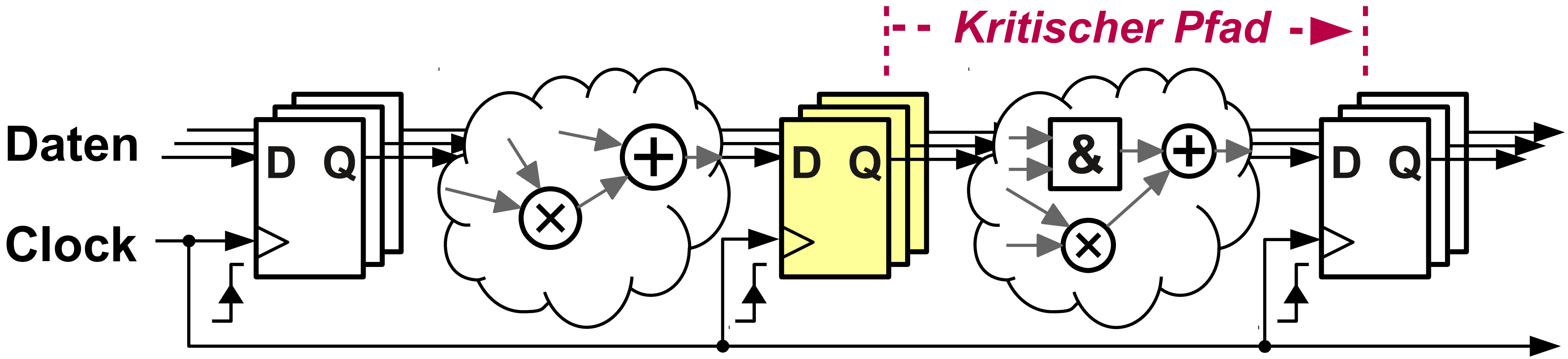
- Abschätzung ob Filter auf einem bestimmten uC / DSP realisierbar ist
- Hier: 3 MACs pro Ausgangs- bzw. Eingangssample
- Einfache Umformungen wie auf letzten Folien ändern nicht die Anzahl der benötigten Rechenoperationen

Kritischer Pfad



- *Alle* Flip-Flops übernehmen Daten mit steigender Flanke des Clock-Netz-werks (synchrone Logik): Daten müssen zu diesem Zeitpunkt stabil sein!
- Laufzeiten in Gattern / Look-Up Tables / Verdrahtung (Kombinatorik) begrenzen maximal mögliche Taktrate f_{Smax}
- Der Datenpfad mit der längsten Laufzeit τ_{krit} in einem Chip / Rechenwerk wird **kritischer Pfad** genannt, da er die maximale Taktfrequenz bestimmt
- Bei FPGA-Synthese wird abhängig von der gewünschten maximalen Taktrate die Implementierung ausgewählt (kompakt & langsam ↔ groß & schnell)

Hohe Taktraten durch Pipelining



Durch zusätzliche Register verkürzt man den krit. Pfad (**Pipelining**):

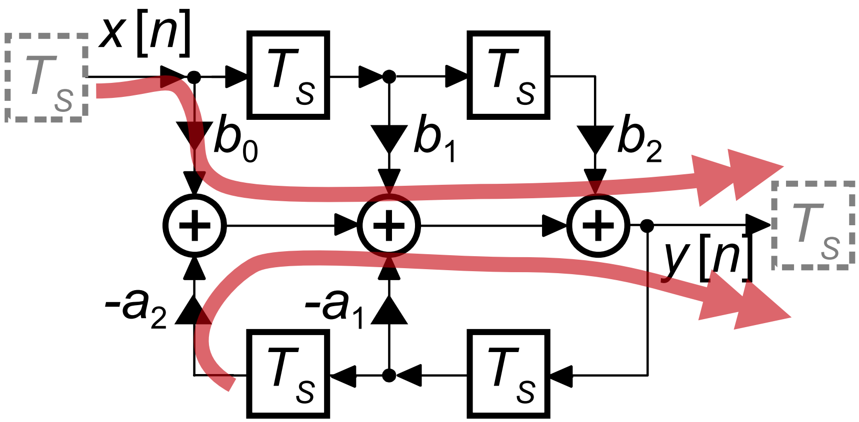
kurzer kritischer Pfad → hohe maximale Taktrate!

Nachteile:

- Zusätzliche Register
- Höhere Latenz (Durchlaufzeit)
- Vor allem bei rekursiven Systemen nicht immer möglich

Beispiele für kritischen Pfad

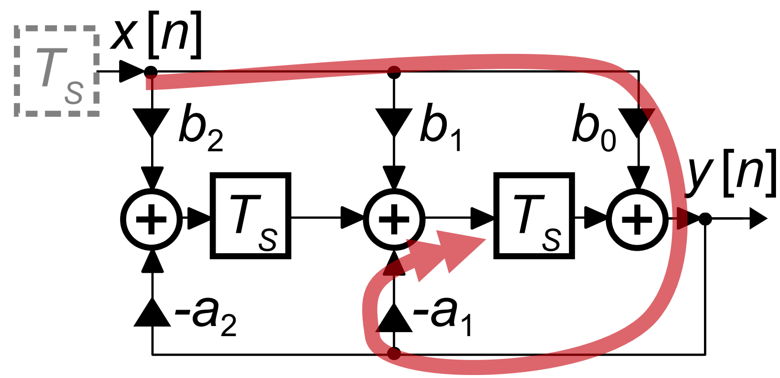
Beispiel: Durchlaufzeit pro Addierer 2 ns, pro Multiplizierer 10 ns:



„Direktform 1“

$$\tau_{krit} = 4\,\tau_{add} + \tau_{mul} = \mathbf{18\ ns}$$

$$\Rightarrow f_{S,max} = 1/\tau_{krit} = \mathbf{55\ MHz}$$

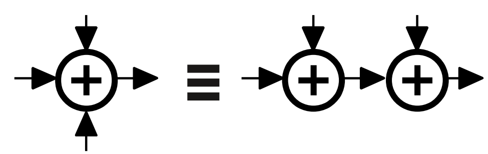


„Transponierte Form“

$$\tau_{krit} = 3\,\tau_{add} + 2\,\tau_{mul} = \mathbf{26\ ns}$$

$$\Rightarrow f_{S,max} = 1/\tau_{krit} = \mathbf{38\ MHz}$$

- Pfade vom Eingang und zum Ausgang nicht vergessen
- Oft mehrere Pfade mit gleicher Laufzeit – entscheidend ist der langsamste
- Addierer mit 3 Eingängen sind eigentlich 2 Addierer mit je 2 Eingängen!



Zusammenfassung Implementierungen

- Eine Systemfunktion kann mit verschiedenen Filtertopologien implementiert werden
- Je nach Implementierung ergeben sich unterschiedliche Laufzeiten, die ineinander umgeformt werden können
- Unterschiedliche Implementierungen können verschiedene maximale Laufzeiten zwischen synchron getakteten Registern haben (kritischer Pfad)
- Durch Pipelining-Register kann der kritische Pfad verkürzt und so die maximale Taktrate erhöht werden