Über die Integration von analytischen und tabellarischen S-Parametermodellen in Transientensimulationen

Oliver Soffke, Thomas Hollstein, Ralf Kügler
und Manfred Glesner

5. Juli 2006

Übersicht

Motivation: Simulation von RFID-Tags im Gesamtsystem

- Die bidirektionale Übertragung zwischen Lesegerät und Tag muß akkurat erfasst werden
- Einfache Modelle: gekoppelte Spulen, Koppelkondensator, etc. . .
- Komplexere analytische Modelle lassen sich entweder als klassische Zweitore (z. B. RLC-Netzwerke) oder als Streumatrizen (z. B. Wellenleiter, UHF-Antennen) angeben
- Messtechnisch: Reale Antennenpaare liegen als tabellarische S-Parameterbeschreibungen vor
- Simulation: Geometrisch beschriebene Antennenpaare liegen als tabellarische S-Parameterbeschreibungen vor

$\Rightarrow$ Eine gemischte Simulation ist wünschenswert

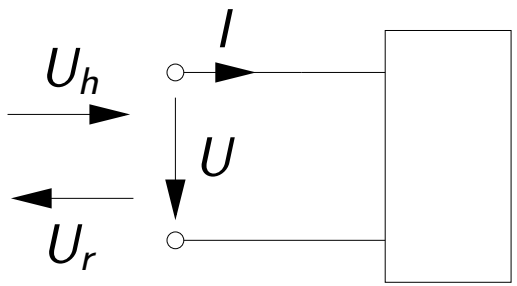
Lösungsansatz

- Integration von S-Parameterbeschreibungen in Schaltungssimulatoren
- S-Parameter liegen entweder analytisch ...
  - Implementierung mit Verilog-A
    - Verilog ähnliche Syntax
    - Erlaubt die Beschreibung analoger Größen
    - Verilog + Verilog-A = Verilog-AMS
- ... oder tabellarisch vor
  - Beschreibung durch gebrochen rationale Funktion
  - Bestimmung der Koeffizienten mittels Regression
  - In Spectre bereits integriert

Kurze Auffrischung: Streumatrix/S-Parameter

Mathematisch: Lineare Abbildung von Spannung und Strom auf hin- und rücklaufende Spannungswelle:

$$U = U_h + U_r$$
$$IZ_0 = U_h - U_r$$



Anschaulich: Auf einer Leitung mit Wellenwiderstand Z_0 läuft eine Welle U_h auf das Tor zu und eine Welle U_r vom Tor weg.

Die klassischen Zweitorgleichungen setzen die Spannungen und Ströme an den Toren zueinander ins Verhältnis (Z-, Y-, H- oder G-Matrix). Die Streumatrix setzt die hin- und rücklaufenden Wellen an den Toren zueinander ins Verhältnis:

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \quad \text{mit} \quad a = \frac{U_h}{\sqrt{Z_0}}, \quad b = \frac{U_r}{\sqrt{Z_0}}$$

Verilog-A (I)

- Verilog-ähnliche Syntax
- Erlaubt die Beschreibung „analoger“ (zeit- und wertkontinuierlicher) Vorgänge
- Modellierung durch Netzwerk aus gesteuerten Quellen
- Ein Modell besitzt Knoten, Zweige und Tore
  - Ein Zweig verbindet zwei Knoten
  - Mehrere parallele Zweige möglich
  - Ein Tor ist ein Verbindungsknoten nach außen
- Einem Zweig kann entweder der Fluß durch den Zweig oder das Potential über dem Zweig zugewiesen werden
- In beiden Fällen kann aber sowohl das Potential als auch der Fluß gelesen werden
- Wechsel der Zuweisung zwischen Potential und Fluß möglich $\longrightarrow$ Modellierung idealer Schalter
- Sonden (Probes):
  - Weder Fluß noch Potential werden zugewiesen
  - Entweder Potential oder Fluß kann gelesen werden
  - Die nicht gelesene Größe wird implizit zu Null gesetzt.

Verilog-A (II)

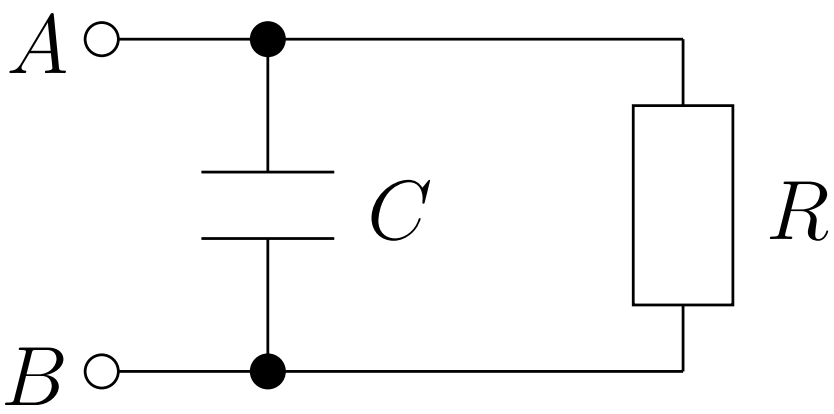
- Zuweisungen sind akkumulativ
- Verilog-A erlaubt multidisziplinäre Simulationen
  - Beispiel: Mechanische belasteter E-Motor und zugehörige Steuerelektronik
- Den Knoten werden entsprechende Disziplinen zugeordnet
- In jeder Disziplin wird eine Größe als Fluß und eine entsprechend zugehörige Größe als Potential modelliert

Beispiele:

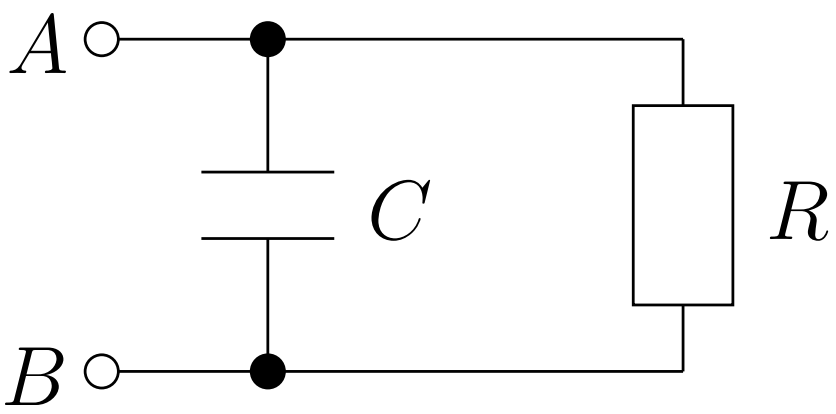
Disziplin	Fluß	Potential
elektrisch	Strom	Spannung
Kinematik	Kraft	Ort
Rotation	Drehmoment	Winkel
Wellen	**einlaufend**	**auslaufend**

- Die Disziplin „Wellen“ wurde hinzugefügt

Verilog-A (Beispiele)



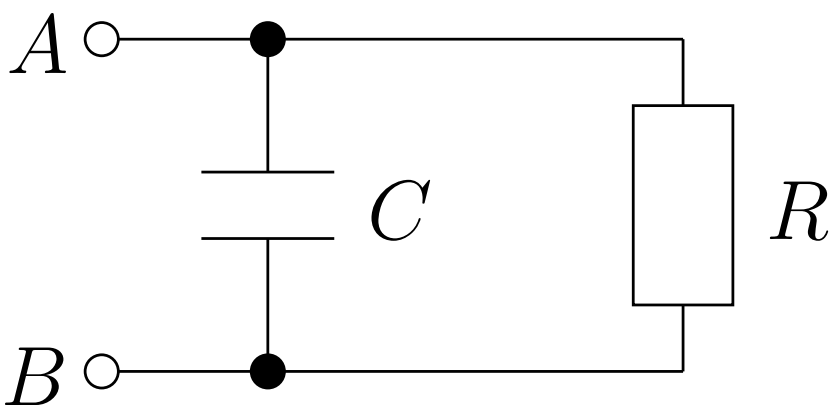
Verilog-A (Beispiele)



Möglichkeit I

```
branch (A,B) Res;
branch (A,B) Cap;
...
V(Res) <+ R*I(Res);
I(Cap) <+ C*ddt(V(Cap));
```

Verilog-A (Beispiele)



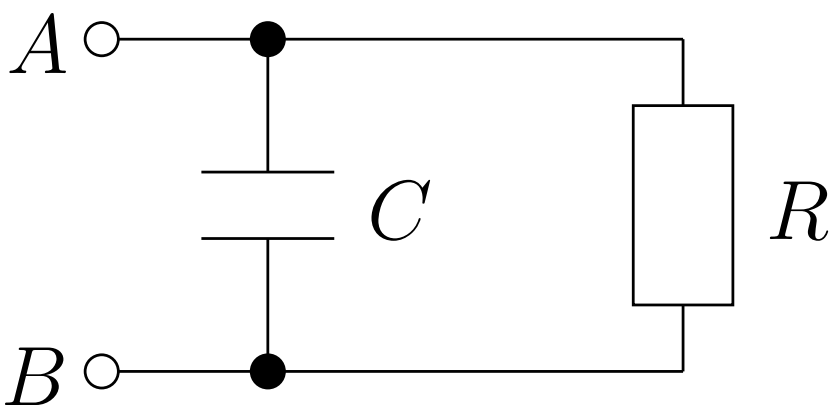
Möglichkeit I

```
branch (A,B) Res;
branch (A,B) Cap;
...
V(Res) <+ R*I(Res);
I(Cap) <+ C*ddt(V(Cap));
```

Möglichkeit II

```
branch (A,B) Res;
branch (A,B) Cap;
...
V(Res) <+ R*I(Res);
V(Cap) <+ idt(I(Cap))/C;
```

Verilog-A (Beispiele)



Möglichkeit I

```
branch (A,B) Res;
branch (A,B) Cap;
...
V(Res) <+ R*I(Res);
I(Cap) <+ C*ddt(V(Cap));
```

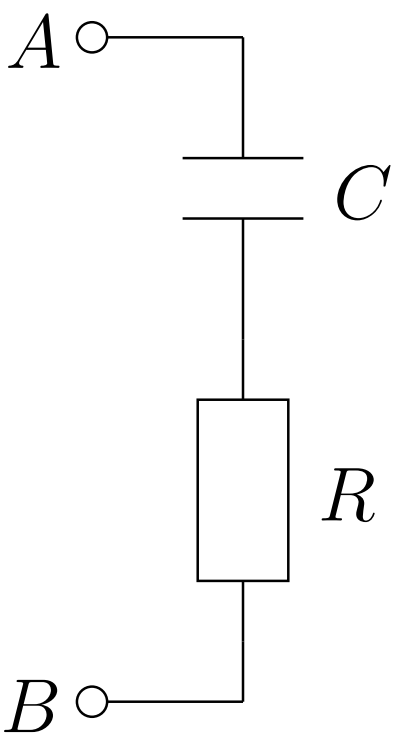
Möglichkeit II

```
branch (A,B) Res;
branch (A,B) Cap;
...
V(Res) <+ R*I(Res);
V(Cap) <+ idt(I(Cap))/C;
```

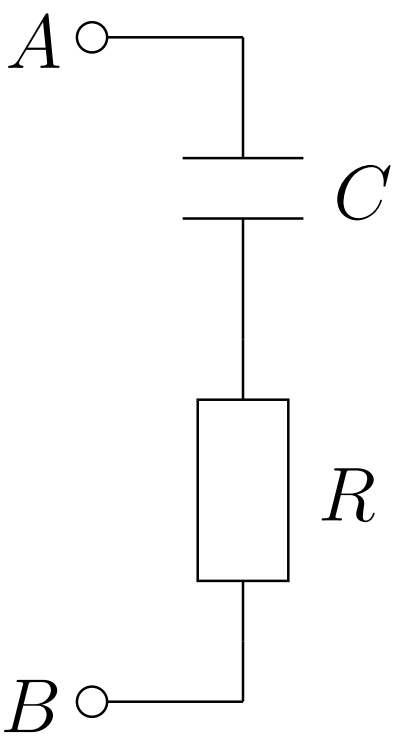
Möglichkeit III

```
branch (A,B) RC;
...
I(RC) <+ V(RC)/R;
I(RC) <+ C*ddt(V(RC));
```

Verilog-A (Beispiele)



Verilog-A (Beispiele)



Möglichkeit I

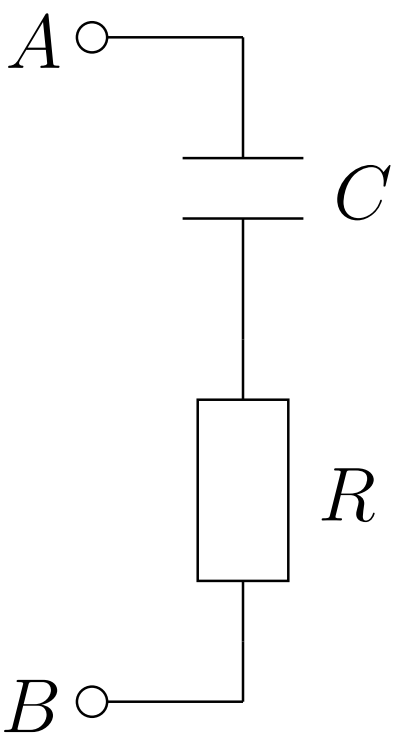
branch (A,B) RC;

...

V(RC) <+ R*I(RC);

V(RC) <+ idt(I(RC))/C;

Verilog-A (Beispiele)



Möglichkeit I

```
branch (A,B) RC;
...
V(RC) <+ R*I(RC);
V(RC) <+ idt(I(RC))/C;
```

Möglichkeit II

```
branch (A,B) RC;
...
I(RC) <+ C*ddt(V(RC)-R*I(RC));
```

Verilog-A: Implementierung der Z-Matrix

$$\begin{pmatrix} U_1 \\ U_2 \end{pmatrix} = \underbrace{\begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}}_{\mathbf{Z}} \begin{pmatrix} I_1 \\ I_2 \end{pmatrix}$$

Verilog-A: Implementierung der Z-Matrix

$$\begin{pmatrix} U_1 \\ U_2 \end{pmatrix} = \underbrace{\begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}}_{\mathbf{Z}} \begin{pmatrix} I_1 \\ I_2 \end{pmatrix}$$

Reelle Impedanzen

```
branch (A1, A0) PortA;
branch (B1, B0) PortB;
...
V(PortA) <+ Z11*I(PortA);
V(PortA) <+ Z12*I(PortB);
V(PortB) <+ Z21*I(PortA);
V(PortB) <+ Z22*I(PortB);
```

Verilog-A: Implementierung der Z-Matrix

$$\begin{pmatrix} U_1 \\ U_2 \end{pmatrix} = \underbrace{\begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}}_{\mathbf{Z}} \begin{pmatrix} I_1 \\ I_2 \end{pmatrix}$$

Reelle Impedanzen

branch (A1, A0) PortA;
branch (B1, B0) PortB;

...

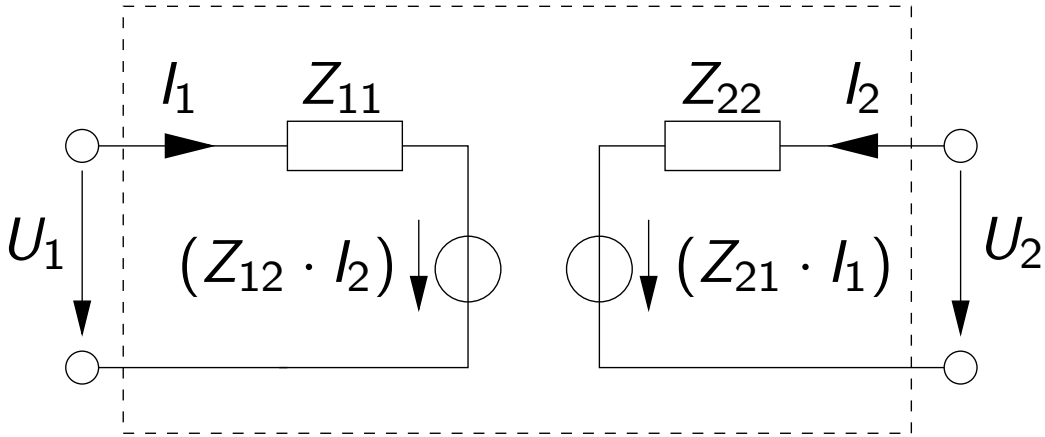
V(PortA) <+ Z_{11}*I(PortA);
V(PortA) <+ Z_{12}*I(PortB);
V(PortB) <+ Z_{21}*I(PortA);
V(PortB) <+ Z_{22}*I(PortB);

Komplexe Impedanzen

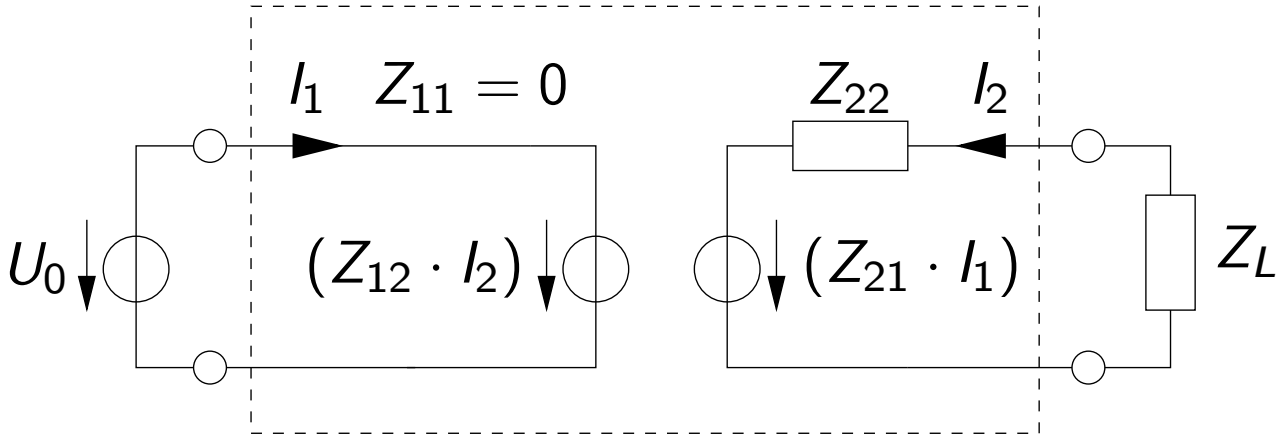
...

V(PortA) <+ laplace_nd(I(PortA), Num11, Denom11);
V(PortA) <+ laplace_nd(I(PortB), Num12, Denom12);
V(PortB) <+ laplace_nd(I(PortA), Num21, Denom21);
V(PortB) <+ laplace_nd(I(PortB), Num22, Denom22);

Verilog-A: Implementierung der Z-Matrix

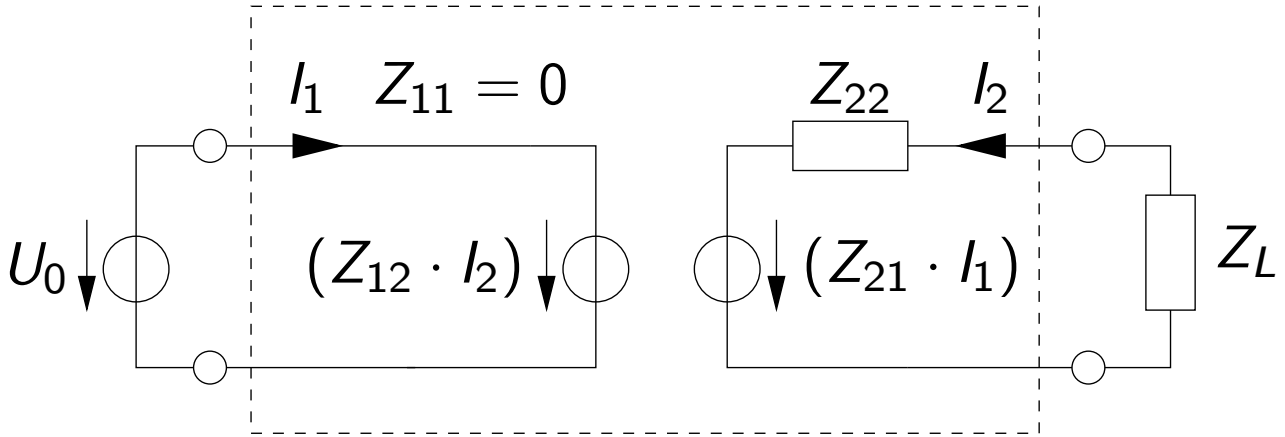


Verilog-A: Implementierung der Z-Matrix



- Ideale Spannungsquelle am Eingang
- Eingangswiderstand $Z_{11} = 0$

Verilog-A: Implementierung der Z-Matrix



- Das Gleichungssystem $\begin{pmatrix} U_0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & Z_{12} \\ Z_{21} & (Z_{22} + Z_L) \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \end{pmatrix}$

  ist eindeutig lösbar:

$$\boxed{I_2 = \frac{U_0}{Z_{12}},\ I_1 = -\frac{(Z_{22} + Z_L)\, U_0}{Z_{12} Z_{21}}}$$

- Dennoch läßt sich die Schaltung nicht simulieren
  - Topologieprüfung des Simulators meldet Fehler
  - Parallelschaltung zweier idealer Spannungsquellen
  - Die Quelle $Z_{12} \cdot I_2$ wird ausschließlich von den Größen anderer Zweige gesteuert

Erweiterung auf Wellengrößen

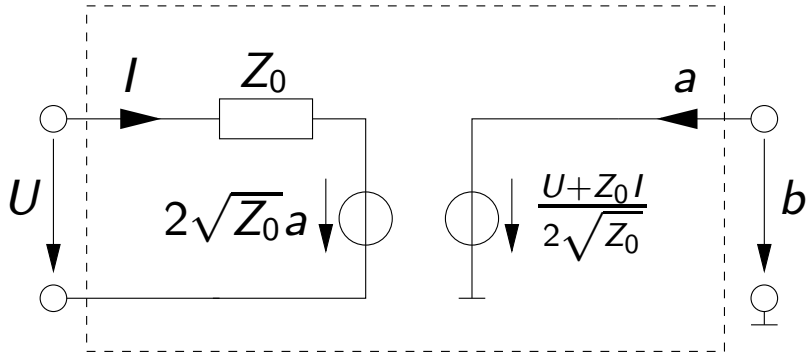
- Zuordnung der Wellengrößen
  - Fluß: Einlaufende Welle
  - Potential: Auslaufende Welle

mydisciplines.vams

```
nature IncidentWave
units = "V/sqrt(Ohm)";
access = A;
endnature

nature ReflectedWave
units = "V/sqrt(Ohm)";
access = B;
endnature

discipline waves
potential ReflectedWave;
flow IncidentWave;
enddiscipline
```



- Konverter von U/I auf a/b
- Potential oder Fluß eines Zweiges kann zugewiesen werden:

$$b = \frac{U + Z_0 \cdot I}{2\sqrt{Z_0}}$$

$$U = 2\sqrt{Z_0} \cdot a + Z_0 \cdot I$$

- Zwei gesteuerte Potentialquellen
- Durch den Serienwiderstand Z_0 gibt es keine Probleme mit der Topologieprüfung

Fluß-Potential Konverter

- Auslaufende Welle von Modul A ist einlaufende Welle von Modul B und umgekehrt
- Dies ist durch einfaches Zusammenschalten nicht gewährleistet
- Ein spezielle „Verbindungsmodul“ wird benötigt

  → Fluß-Potential Konverter

Fluß-Potential Konverter

```
module FPX (W1, W2);
waves W1, W2;
branch (W1) W1port;
branch (W2) W2port;

analog begin
A(W1port) <+ −B(W2port);
A(W2port) <+ −B(W1port);
end
endmodule
```

- Setzt auslaufende Welle von Modul A auf einlaufende Welle von Modul B um
- Besteht aus zwei gesteuerten Flußquellen
- Zusammenschaltung mit den gesteuerten Potentialquellen der „regulären“ Module problemlos möglich

Das eigentliche Modell

- Das Modell selbst wird im Falle eines Zweitors folgendermaßen beschrieben:

$$b_1 = S_{11}a_1 + S_{12}a_2$$
$$b_2 = S_{21}a_1 + S_{22}a_2$$

- Dies läßt sich unmittelbar in Verilog-A umsetzen:

S-Matrix Implementierung

```
...
B(W1Port) <+ laplace_nd(A(W1Port), Num11, Denom11);
B(W1Port) <+ laplace_nd(A(W2Port), Num12, Denom12);
B(W2Port) <+ laplace_nd(A(W1Port), Num21, Denom21);
B(W2Port) <+ laplace_nd(A(W2Port), Num22, Denom22);
```

- Durch den Fluß-Potential-Konverter ist das Zusammenschalten problemlos möglich, auch wenn $S_{11} = 0$ und/oder $S_{22} = 0$ (Topologieprüfung!)
- Dieses läßt sich direkt auf N-Tore erweitern

S-Parameter: Beispiele

- Zusätzlich können die analogen Operatoren, wie beipielsweise delay() zum Einsatz kommen

RC-Tiefpass

$$S = \begin{pmatrix} \frac{R+CZ_0(R-Z_0)s}{R+2Z_0+CZ_0(R+Z_0)s} & \frac{2Z_0(R+Z_0)}{Z_0(3R+2Z_0)+R^2+CZ_0(R+Z_0)^2s} \\ \frac{2Z_0(R+Z_0)}{Z_0(3R+2Z_0)+R^2+CZ_0(R+Z_0)^2s} & \frac{R-CZ_0(R+Z_0)s}{R+2Z_0+CZ_0(R+Z_0)s} \end{pmatrix}$$

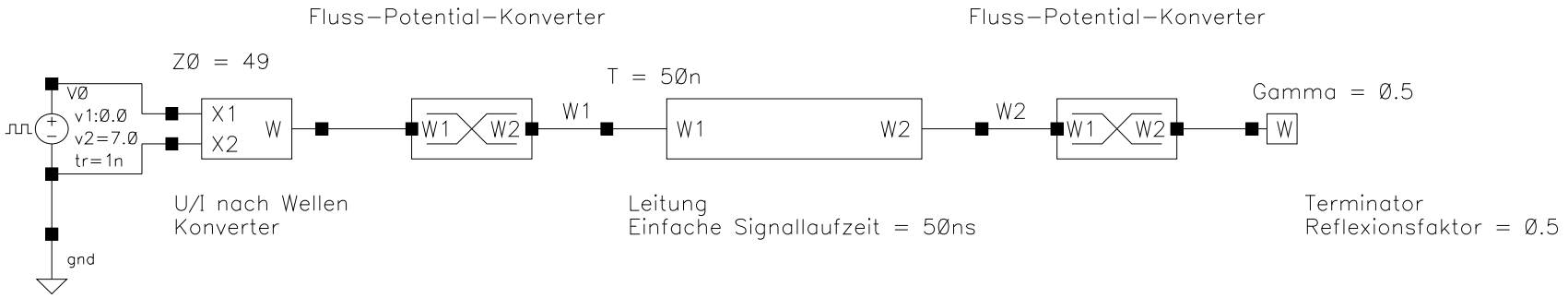
```
B(W1port) <+ laplace_nd(A(W1port), {R, C*Z0*(R-Z0)}, {R+2*Z0, C*Z0*(R+Z0)});
B(W1port) <+ laplace_nd(A(W2port), {2*Z0*(R+Z0)}, {Z0*(3*R+2*Z0)+R*R, C*Z0*(R+Z0)*(R+Z0)});
B(W2port) <+ laplace_nd(A(W1port), {2*Z0*(R+Z0)}, {Z0*(3*R+2*Z0)+R*R, C*Z0*(R+Z0)*(R+Z0)});
B(W2port) <+ laplace_nd(A(W2port), {R, -C*Z0*(R+Z0)}, {R+2*Z0, C*Z0*(R+Z0)});
```

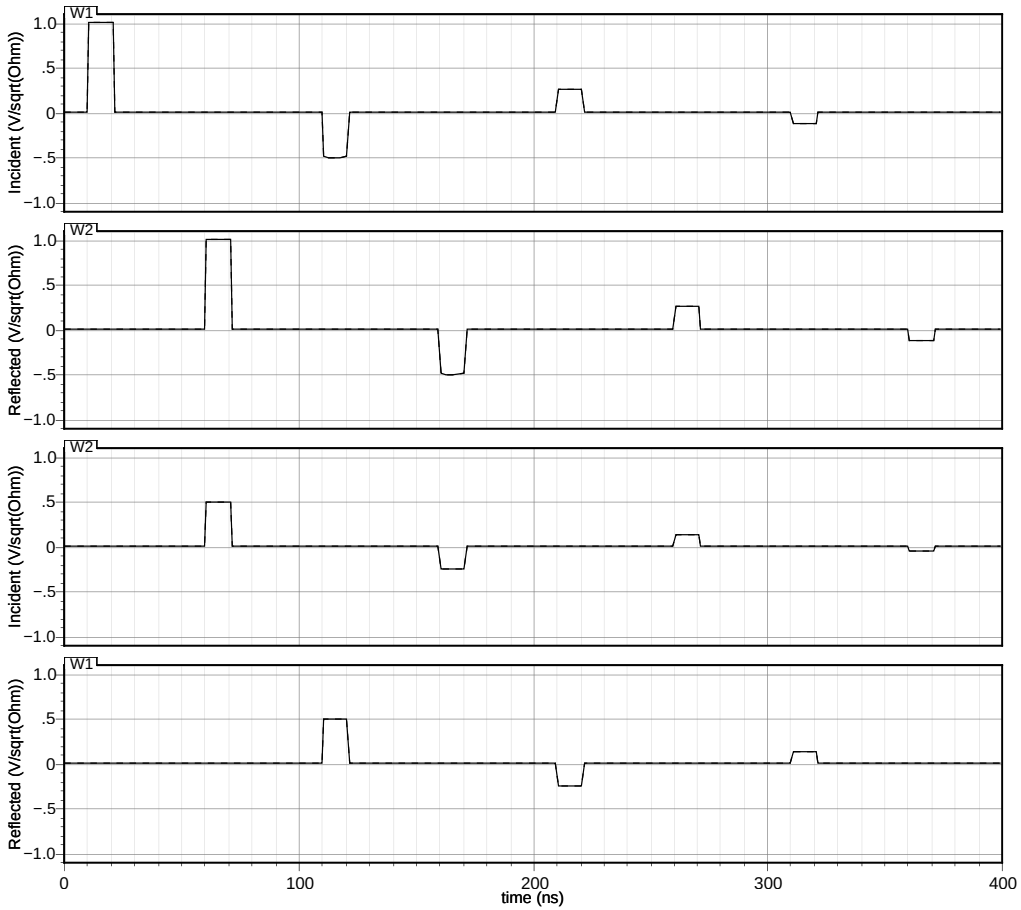
Verlustlose Leitung

$$S = \begin{pmatrix} 0 & e^{-sT} \\ e^{-sT} & 0 \end{pmatrix} \quad \bullet\!\!-\!\!\circ \quad \begin{aligned} b_1(t) &= a_2(t-T) \\ b_2(t) &= a_1(t-T) \end{aligned}$$

```
B(W1port) <+ delay(A(W2port), T);
B(W2port) <+ delay(A(W1port), T);
```

Beispiel für eine Wellensimulation





Terminator

B(W) <+ Gamma*A(W);

Tabellarische S-Parameter

- Häufig liegt die Streumatrix nicht in analytischer sondern in tabellarischer Form vor
  - Messung mit Vektoranalysator
  - EM Simulationen
  - ...
- S-Parameter lassen sich häufig als gebrochen rationale Funktion ausdrücken:

$$S_{nm} = \frac{\sum_{i=0}^{\infty} b_i s^i}{\sum_{i=0}^{\infty} a_i s^i} \approx \frac{\sum_{i=0}^{M} b_i s^i}{\sum_{i=0}^{N} a_i s^i}.$$

- Die Koeffizienten lassen sich durch nichtlineare Regression bestimmen
- Ein entsprechener Algorithmus ist in Cadence Spectre bereits implementiert

Zusammenfassung (I)

- Verilog-A ist für die Verhaltensbeschreibung und Modellierung mit zeit- und wertkontinuierlichen Größen hervorragend geeignet (aber auch Abtastung kann modelliert werden)
- Viele „analoge“ Operatoren stehen zur Verfügung:
  - delay() für Verzögerungen
  - transition() für steigende und fallende Flanken
  - Zeitkontinuierliche Filter (laplace_xy())
  - Zeitdiskrete Filter (zi_xy())
  - ddt() und idt() für Ableitung und Integration nach der Zeit
- Zur Beschreibung komplexer Größen im Frequenzbereich kann der laplace_xy()-Operator verwendet werden
- Allerdings muß auf die resultierende Schaltungstopologie geachtet werden
- Modellierung ist multidisziplinär
  - Elektrisch, mechanisch, thermisch, etc. . .
  - Eine Disziplin „Wellen“ wurde hinzugefügt

Zusammenfassung (II)

- „Supportmodule“ wurden entwickelt
  - Transformation der U/I-Ebene in die A/B-Ebene und umgekehrt
  - Verbindung der einzelnen Modelle
- Beispielsimulation mit einer einfachen Leitung
  - Läßt sich durch Streuparameter sehr einfach beschreiben
  - Gemischte U/I und A/B Simulation möglich: Jedes System lässt sich in der jeweils einfachsten Weise modellieren
- Tabellarische S-Parameter lassen sich durch nichtlineare Regression auf gebrochen rationale Funktionen abbilden
- Auf diese Weise ist es möglich messtechnisch oder durch numerische Simulationen erfasste Systeme in die Gesamtsimulation einzubeziehen

? !

? Vielen Dank !

? !