transferfunction

Projekt zur Veranstaltung „Algorithmen und Datenstrukturen

Dennis Dette (43511) Christian Kobiela (47571)

Inhalt

[Aufgabenstellung 1](#_Toc486271669)

[Datenstruktur 2](#_Toc486271670)

[Parallelschaltung - Is\_parallel() 4](#_Toc486271671)

[Reihenschaltung - Is\_serial() 5](#_Toc486271672)

[Stern zu Dreieck Wandlung 6](#_Toc486271673)

[Dreieck zu Stern Wandlung 8](#_Toc486271674)

Aufgabenstellung

## PROBLEMBESCHREIBUNG

Die Komplexe Übertragungsfunktion von RCL-Netzwerken kann beim Vorliegen bestimmter Strukturen ohne die Lösung von Gleichungssystemen ermittelt werden. Dazu soll ein solches Netzwerk als Graph abgebildet werden und durch sukzessive Zusammenfassung von Bauelementen entsprechend der Reihenschaltungs- bzw. der Parallelschaltungsregeln zusammengefasst werden. Blöcke, die nicht zusammengefasst werden können, sollen in einem eigenständigen Teilgraphen zusammengefasst werden, der dann als eigenständige Einheit in einem weiteren Reduzierungsschritt aufgefasst werden kann. Hauptanliegen ist jedoch die Zusammenfassung der Gesamtschaltung in Form einer rationalen Funktion. [1]

## EINGABEDATEN

Nets: a:IN; b:Out; c: CMN; d,e: Internal; //CMN entspricht Masse

R1:R ( a, d); C2:C ( d, b); L4:L ( b, c );

## AUSGABE

Es sollen die einzelnen Schritte der Zusammenfassungsoperationen und eine textuelle Form der Resultatsfunktion ausgegeben werden. Zur besseren Darstellung der Transferfunktion soll auch eine Bruchrepräsentation mit der grafischen Oberfläche GDE erzeugt werden. [1]

[1] Quelle: Prof Schäfer - [ADS\_Aufgws1617.pdf]

Datenstruktur

Die Daten werden wie in der Aufgabenstellung gefordert eingelesen und beschreiben die Bauteile in der Form Name:Art(Pin1, Pin2). Es wurde eine Klasse Bauteil erstellt die diese Informationen bündelt und die Bauteile werden in einem Vektor dieser Klasse gespeichert und verarbeitet.

public:

string Name;

string Art;

string Pin1;

string Pin2;

Bauteil(string Name, string Art, string Pin1, string Pin2);

Bauteil();

};

Bauteil::Bauteil(string a, string b, string c, string d) {

Name = a;

Art = b;

Pin1 = c;

Pin2 = d;

}

Ebenso wurde eine Klasse erstellt die das Netzwerk als Ganzes beschreibt. Dies ist erforderlich um festzulegen und später festzustellen welche Knoten Ein-, Ausgang, GND oder Intern sind.

class Network {

public:

string INPUT; string OUTPUT; string CMN;

vector<string> INTERNALS;

Network();

Network(string INPUT, string OUTPUT, string CMN, vector<string> INTERNALS);

};

Network::Network() {}

Network::Network(string a, string b, string c, vector<string> d) {

INPUT = a;

OUTPUT = b;

CMN = c;

INTERNALS = d;

Zur Betrachtung von Stern- oder Dreieckspunkten welche nicht durch eine Reihen-, Parallelschaltung vereinfacht werden können, wurde eine Adjazenzmatrix erstellt. Dazu später mehr.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PINS | A | B | C | D | E |
| A |  |  |  | R1 |  |
| B |  |  |  | R3 |  |
| C |  |  |  | Z2||Z1 |  |
| D | R1 | R3 | Z2||Z1 |  |  |
| E |  |  |  |  |  |

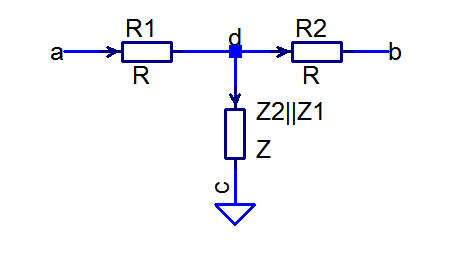


Abbildung 1: Mögliche Endform mit 3 Impedazen

\*Anmerkung, hierbei handelt es sich zwar um einen Stern, da es nur noch 3 Bauteile sind werden diese nicht weiter vereinfacht. An diesem Punkt wird die Übertragungsfunktion ausgegeben.

Parallelschaltung - Is\_parallel()

Der Algorithmus untersucht den vector<Bauteil\*> Bauteile zu aller erst nach Bauteilen die parallel zueinander sind. Dies geschieht iterativ in dem jedes Bauteil mit jedem verglichen und überprüft wird ob diese an beiden Pins zusammen hängen.

Anschließend wird aus den beiden Bauteilen ein neues erstellt und hinten an den Vector geschrieben.

Bauteile.push\_back(new Bauteil(

Bauteile.at(i)->Name

+ "||" +

Bauteile.at(ii)->Name,

Die verwendeten Bauteile werden anschließend gelöscht. Die Ausführdauer dieser Funktion steigt exponentiell mit der Länge des Bauteilevectors.

|  |  |
| --- | --- |
| (A.Pin2 == B.Pin1)&&(A.Pin1 == B.Pin2))||  ((A.Pin2 == B.Pin2)&&(A.Pin1 == B.Pin1) ? | |
| Ja | Nein |
| Bauteile Zusammenfassen | Return; |
| Verwendete Bauteile löschen |  |

Reihenschaltung - Is\_serial()

So einfach wie die Suche nach parallelen Bauteilen war die nach Seriellen nicht. Hierbei mussten einige Dinge beachtet werden. Was auf dem Papier einfacher ist, ist hier eindeutig komplexer.

Da bei einer seriellen Zusammenfassung, im Gegensatz zur Parallelen, Pins wegrationalisiert werden, musste darauf geachtet werden, dass es sich dabei nur um interne Pins handelte. Der Verlust eines Eingangs würde es unmöglich machen eine Übertragungsfunktion aufzustellen.

Das Netzwerk wird aus Sicht der Pins betrachtet. Es werden alle Pins untersucht ob an ihnen 2 und auch wirklich nur 2 Bauteile hängen. Wenn dieser Pins dann noch ein Internal war so konnte der nächste Schritt der Zusammenfassung beginnen.

Es wird überprüft welche Bauteile an diesem Pin angeschlossen sind und diese werden dann zwischengespeichert. Anschließend wird festgestellt welche Pins die Äußeren sind um bei der Zusammenfassung das neue Bauteil richtig zu beschreiben.

|  |  |
| --- | --- |
| Is\_serial\_PIN()? | |
| Ja | Nein |
| Find\_serial\_Bauteile() | Return; |
| find\_outer\_Pins() |  |
| Create\_new\_Bauteil() |  |

Das neue Bauteil wird dem vector<Bauteil\*> Bauteile angehängt und die verwendeten Bauteile werden gelöscht.

Stern zu Dreieck Wandlung

Ist nun kein Bauteil mehr zusammenzufassen und es existieren noch mehr als 3 Bauteile muss ein Stern vorliegen. Die Detektion ob ein Stern vorliegt und an welchem Pin dies der Fall ist wird über eine Adjazenzmatrix bestimmt.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PINS | A | B | C | D |
| A |  |  |  | R1 |
| B |  |  | R4 | R3 |
| C |  | R4 |  | Z2||Z1 |
| D | R1 | R3 | Z2||Z1 |  |

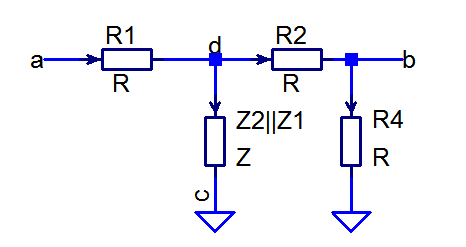


Abbildung 2: Sternstruktur

Liegt nun ein Stern vor so beinhaltet die Adjazenzmatrix genau 3 Einträge in einer Spalte/Zeile und zwar genau am betreffenden Sternpin der im Vereinfachungsschritt gelöscht werden wird.

Es werden die Impedanzen nach den Formeln der Sternwandlung zusammengefasst um weitere Reihen- und Parallelvereinfachungen zu ermöglichen.

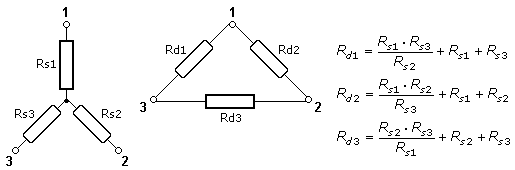


Abbildung 3: Stern Dreieck Wandlung [Quelle: elektroniktutor.de]

Dreieck zu Stern Wandlung

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PINS | A | B | C | D |
| A |  | R2 | R1 |  |
| B | R2 |  | Z2||Z1 |  |
| C | R1 | Z2||Z1 |  |  |
| D |  |  |  |  |

Interessanterweise kann man immer Sternpunkte finden und es würde ausreichen die Stern zu Dreieck Wandlung zu implementieren. Bis auf einen Sonderfall wenn ausschließlich 3 Bauteile in einem Dreieck zwischen dem Ein-, Ausgang und der Masse angeordnet sind.

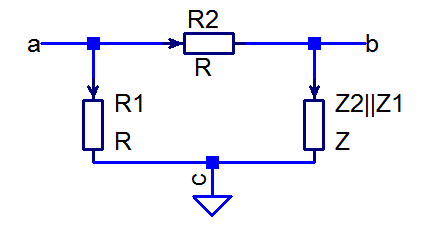


Abbildung 4: Dreieckstruktur

Diesen Fall erkennt man daran, dass eben die besagten 3 Bauteile an genau 3 Pins hängen. Die Dimension der Matrix gleicht also der Größe des Bauteile Vektors.

In diesem Fall werden ebenfalls die bekannten Formeln angewandt

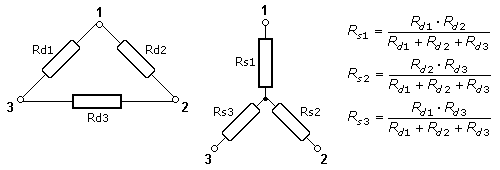


Abbildung 5:Dreieck Stern Wandlung [Quellel: elektroniktutor.de]

Hier muss das einzige mal ein Pin dem Netzwerk hinzugefügt werden, der Sternpin \* .