Versuchsbeschreibung Nr. 6

<u>Verwendete Geräte:</u> 1 Oszilloskop
1 Multimeter
1 Frequenzgenerator

<u>Literatur:</u> 1. L.Borucki: Digitaltechnik (Kap. 7,8)

<u>Teil 1:</u> - Kennenlernen von Flip-Flops

<u>Anmerkung:</u> Verwenden Sie für alle Meßaufbauten den IC-Steckkasten und achten Sie darauf, daß Sie unbenutzte Gatter auf ein definiertes Potential legen.
Die Spannungsversorgung aller IC's in diesem Versuch ist: VCC = +5V GND = 0V.
Es stehen die im Anhang dargestellten Bausteine für den Meßaufbau zur Verfügung.

<u>Aufgaben:</u>

1.
- Ermitteln Sie von untenstehender Schaltung durch Messung die Verknüpfungstabelle.
- Diskutieren Sie das Messergebnis.
- Wie sieht die reduzierte Wahrheitstabelle bei Verwendung von NAND-Gattern aus ?
- Welche Bedeutung haben R = S =0 bzw. R = S = 1 ?
- Wo finden Flip Flops Anwendungen ?

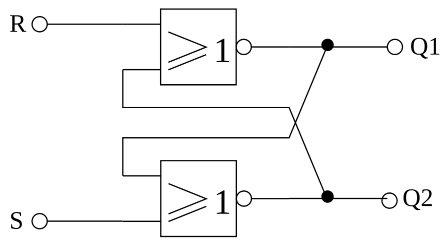


Abb.14: RS-Flip-Flop

2. - Bauen Sie untenstehende Schaltung auf.
- Messen Sie die Wahrheitstabelle eines D-Flip-Flops (R=S=0).
- Wie wirken R und S ?
- Obwohl für alle Kombinationen von R, S, D und Clock (C) definierte Ausgangszustände des D-FFs zu finden sind, sind einige logisch verboten. Bestimmen Sie diese Zustände.

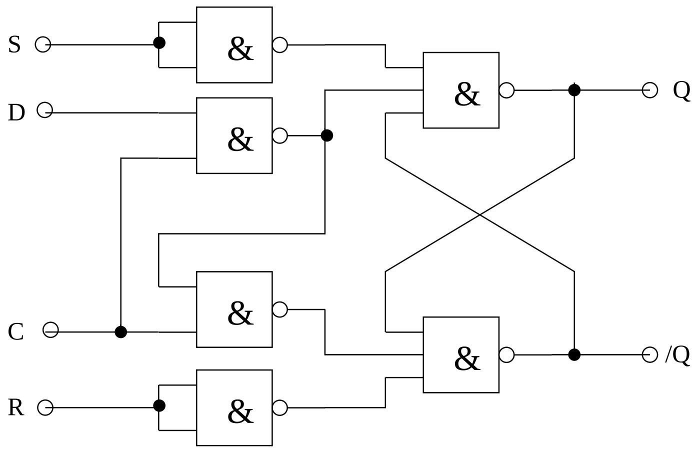


Abb.15: D-Flip-Flop

Teil 2: - Logische Simulation eines sequentiellen Schaltkreises

Anmerkung: Die Simulation der Schaltung wird mit der Demoversion von SILOS III durchgeführt. Die Demo-Version ist frei kopierbar und bei E. Mackensen im Raum 01..31 erhältlich.

Im Anhang ist das Programm SILOS III erläutert.

Aufgabe:

Auf der folgenden Seite ist die Schaltung des käuflich erhältlichen Bausteines MC14163B von Motorola zu sehen. Es handelt sich hierbei um einen 4-Bit-Zähler mit verschiedenen Funktionen. Auf der darauffolgenden Seite ist das aus dem Datenblatt des MC14163B angegebene Timingdiagramm zu sehen.

- Programmieren Sie die Schaltung als ASCII-File, wie im Anhang beschrieben ist.
  Zeichnen Sie dazu die Namen jedes Signales in das Schaltbild ein.
  Es soll das Verhalten der Schaltung für folgende Fälle nacheinander analysiert werden:
  * Löschen (Clear)
  * Ladevorgang (Load, Preset)
  * Zählen (Count)
  * Festhalten (Inhibit)
  Speichern Sie das ASCII-File unter c:\labor\vers6\mc14163b.dat ab.

- Simulieren Sie die Schaltung und betrachten Sie sich das entstandene Timingdiagramm.
  Vergleichen Sie Ihr Ergebnis mit dem aus dem Datenblatt mitgelieferten Timingdiagramm.

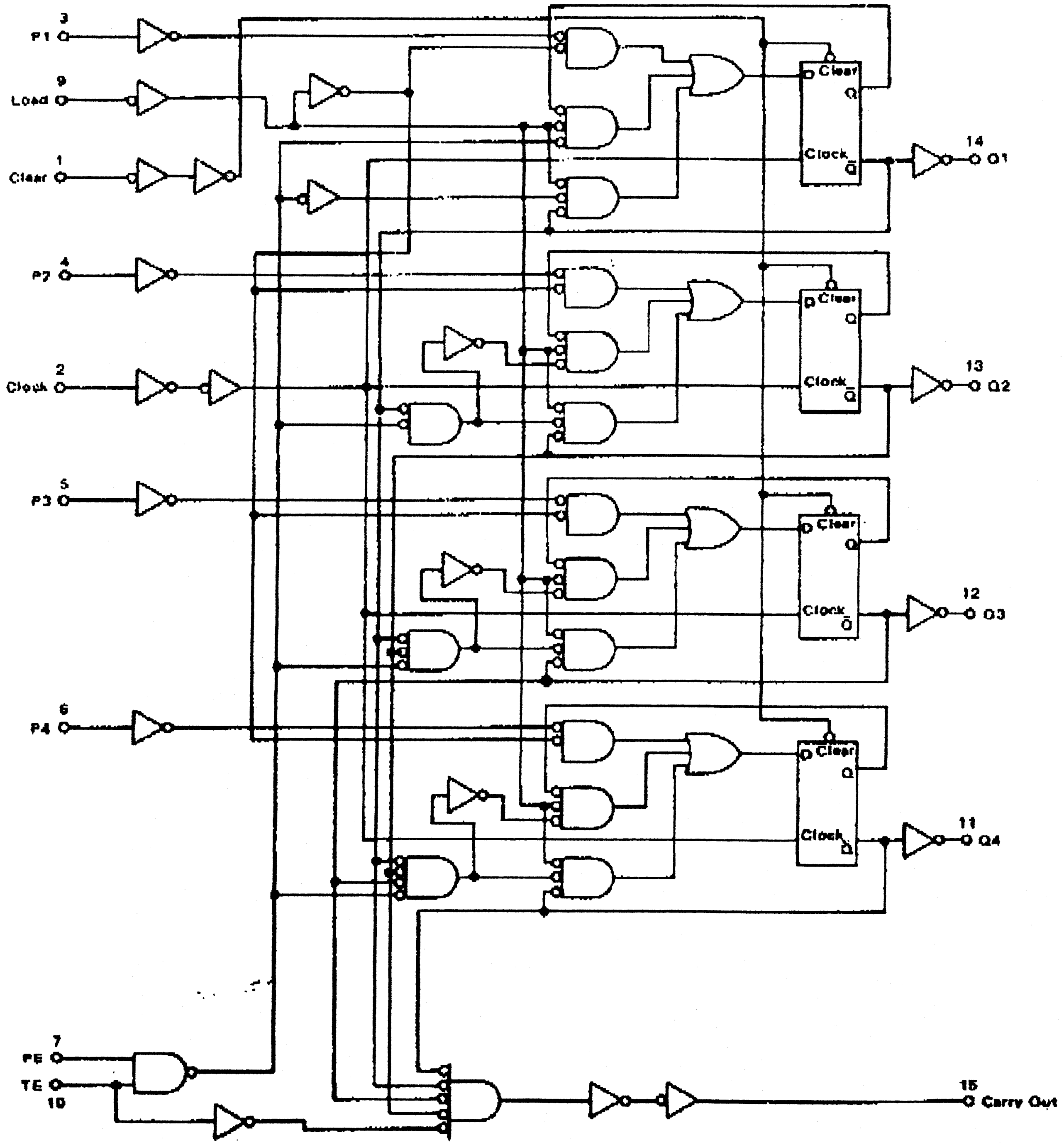
- Drucken Sie Ihr ASCII-Programmierfile und das entstandene Timingdiagramm aus.

Anmerkungen:

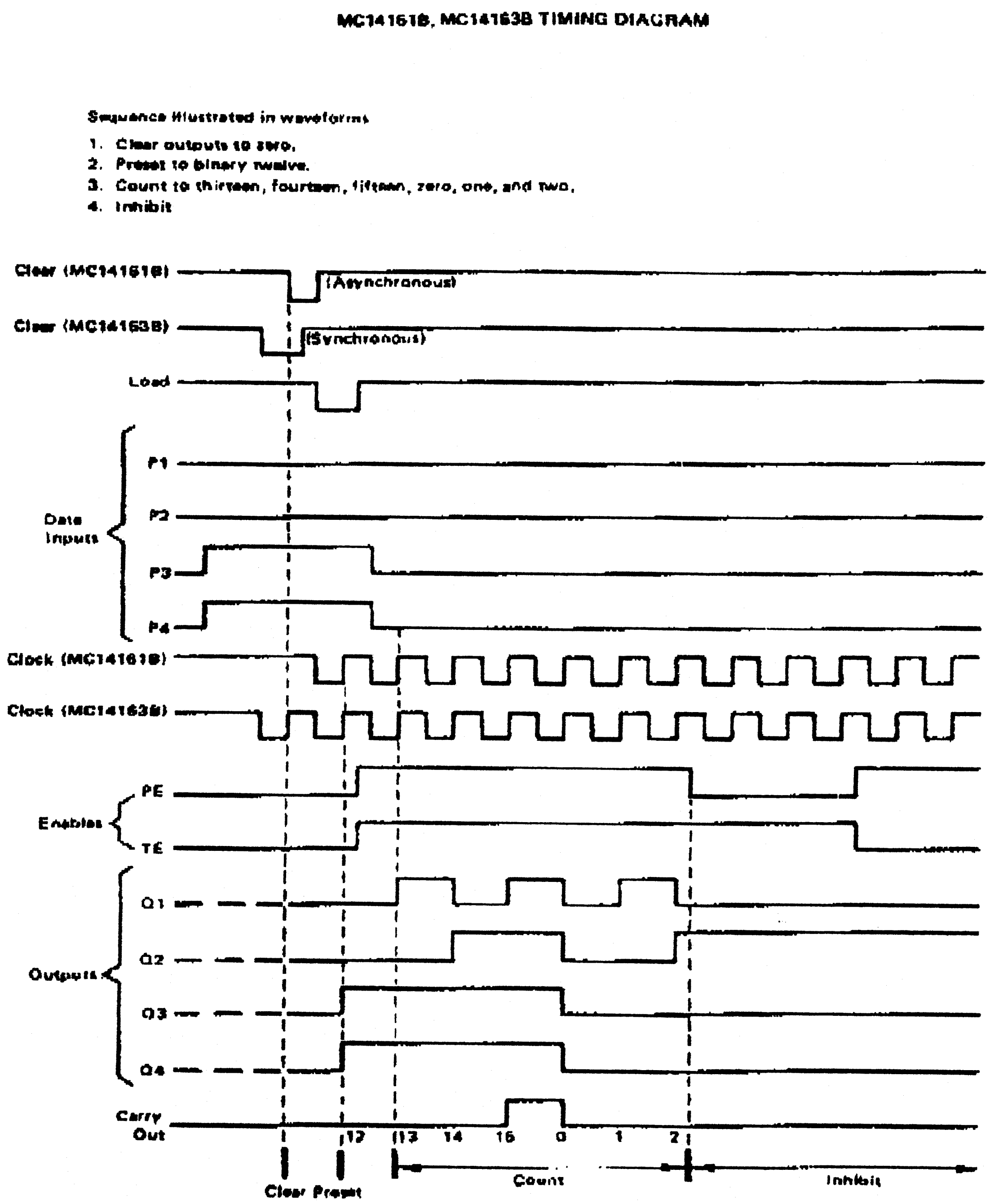
1. Verwenden Sie für Ihre Schaltungsprogrammierung D-Flip Flops, die positiv flankengetriggert sind (DPEFF).
2. Ein Kreis in der Schaltungsdarstellung entspricht einer Negation, wobei sowohl ein Dreieck gefolgt von einem Kreis wie auch der umgekehrte Fall einem Inverter entsprechen.

Schaltung des MC14163B aus dem Datenblatt:

MC14161B, MC14163B LOGIC DIAGRAM
(Clear is Synchronous for MC14163B)



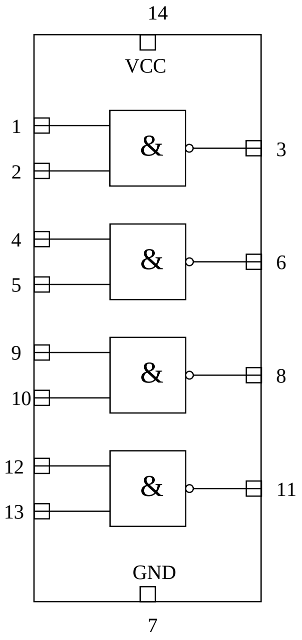
Timingdiagramm des MC14163B aus dem Datenblatt:



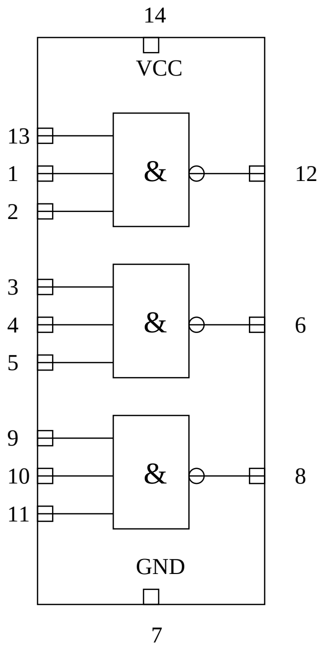
Anhang zu Versuchsbeschreibung Nr. 6

<u>Die zu verwendeten IC's in Teil 1</u>

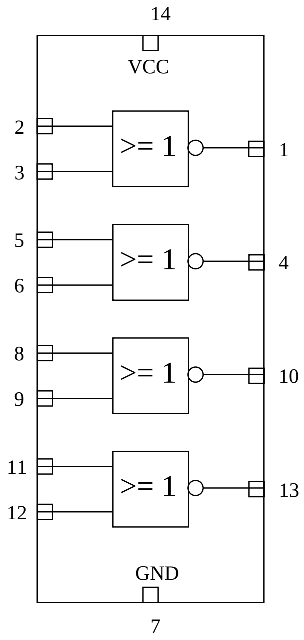
7400



7410



7402



Allgemeines zum Logikanalyseprogramm SILOS

SILOS ist ein Programm zur Logikanalyse von digitalen Schaltungen. Die zu simulierende Schaltung wird in Form einer beschreibenden Sprache als ASCII-File eingegeben, es kann dazu jeder Texteditor verwendet werden.

Ein SILOS-Eingabe-File besteht aus folgender Struktur:

* Titelzeile: .title Titelzeile
* Netzwerkbeschreibung
* Beschreibung der Stimulisignale
* Beliebige Kommentare

Alle Steueranweisungen beginnen mit einem "." in der ersten Spalte. Kommentare beginnen mit dem "$", sie können auch innerhalb einer Zeile angefügt werden. Leerzeilen gelten als Kommentarzeilen, sie können beliebig zur besseren Lesbarkeit eingefügt werden.

Die Netzwerkbeschreibung erfolgt durch die Beschreibung der Ein- und Ausgänge eines Gatters, wobei Netze durch beliebig gewählte Namen eindeutig zu kennzeichnen sind. Jedes Gatter belegt eine Zeile des Files z.B.

Nand23	.NAND	Eingang1	Eingang2 ......
1.)	2.)	3.)	4.)

Es bedeuten:

1.) Name des NANDs, zugleich Name des Ausgangsnetzes
2.) Gattertyp, hier NAND-Gatter
3.), 4.) beliebige Anzahl von Eingängen

Die Netzbezeichnung muß immer eindeutig sein, auch darf die Bezeichnung nur einmal verwendet werden. Bei einigen Elementen, z.B. beim positiv edge triggered Flip-Flop .JKPEFF sind Elementbezeichnung und Bezeichnung der Ausgänge verschieden:

FFLOP1	.JKPEFF	Clk	J1	K1	Set	Reset	/ Q1	Q1bar
1.)	2.)	3.)	4.)	5.)	6.)	7.)	8.)	9.)

Es bedeuten:

1.) Eindeutiger Name des Flip-Flops
2.) Aufgerufener Flip-Flop-Typ
3.) Takteingang
4.) Eingang J
5.) Eingang K
6.) asynchroner Setzeingang
7.) asynchroner Rücksetzeingang
8.) Ausgang Q
9.) Ausgang Q - nicht

Eine ähnliche Beschreibung kann für alle Elemente der Schaltung niedergeschrieben werden, wobei die jeweiligen Kurzzeichen dem SILOS-Handbuch-Auszug zu entnehmen sind. Für jedes Netz muß ein Name vergeben werden, der später auch bei der Spezifikation der Ausgangssignale anzugeben ist.

Die Stimuli, d.h. die Eingangssignale werden durch die folgende Anweisung beschrieben:

Takt	.CLK	0	S0	50	S1	70 S0 90 S1	.REP	50
1.)	2.)	3.)	4.)	5.)	6.)	7.)	8.)	9.)

Es bedeuten:

1.) Name des mit dem Stimuli zu belegenden Netzes
2.) Stimulus-Anweisung
3.) Zeitpunkt, an dem das Signal beginnt, in internen Zeiteinheiten (nsec Voreinstellung)
4.) Logischer Pegel des Signals, S steht für "Strong", d.h. Pegel wird erzwungen
5.) Zeitpunkt für neuen Pegel
6.) Logischer Pegel für neuen Zeitpunkt
7.) Zeitpunkt für nächsten Pegelwechsel u.s.w.
8.) Wiederholungsanweisung
9.) Wiederholungsperiode

Diese Anweisung ist für alle Eingangssignale zu wiederholen. Komplizierte Muster können mit der .Pattern-Anweisung vereinfacht eingegeben werden. Das Programm prüft vor Beginn jeder Simulation nach, ob alle Eingangssignale definiert sind.

Die Reihenfolge der Anweisungen bei SILOS ist unbedeutend.

In dieser einfachen Form haben alle Gatter nur eine minimale (default) Verzögerungszeit. In realen Systemen müssen die Verzögerungszeiten ebenfalls angegeben werden, was durch die optionale Einfügung der Anstiegszeit trise, der Anstiegsverzg̈erung mrise, der Abfallverzögerungszeit mfall bei jedem Gatter erfolgen kann.

Ferner kann durch eine .Delay-Anweisung für einen bestimmten Gattertyp eine generelle Spezifikation der Verzg̈erungszeiten erfolgen.

In ähnlicher Weise kann auch das Fan-Out durch den generellen .Load-Befehl definiert werden.

SILOS verfügt zudem über Befehle, mit denen man Bibliotheken und Unterprogramme im Sinne eines hierachischen Entwurfs einbinden kann, Sonderbefehle zur Beschreibung von RAM's und ROM's und Befehle zum Prf̈en auf Spikes.

Die wichtigsten Befehle für das ASCII-Eingabe-File

Befehle für die Netzwerkbeschreibung

Gate Level Primitives

Level and Strength Delay Functions

name $\left\{\begin{array}{l}\text{.DLA}\\ \text{.SDLA}\end{array}\right\}$ $[t_r,m_r,\ t_f,m_f]$ [load*] input

Combinatorial Gate Functions

name $\left\{\begin{array}{l}\text{.INV strg}\\ \text{.AND strg}\\ \text{.NAND strg}\\ \text{.OR strg}\\ \text{.NOR strg}\\ \text{.XOR strg}\\ \text{.XNOR strg}\end{array}\right\}$ $[t_r,m_r,\ t_f,m_f]$ [load*] $i_1\ i_2\ i_3 \ldots$

Tri-State Gate Functions

name $\left\{\begin{array}{l}\text{.TINV strg}\\ \text{.TAND strg}\\ \text{.TNAND strg}\\ \text{.TOR strg}\\ \text{.TNOR strg}\end{array}\right\}$ $[t_r,m_r,\ t_f,m_f]$ [load*] $i_1\ i_2\ i_3 \ldots$

And-Or-Invert, Or-And-Invert Functions

name $\left\{\begin{array}{l}\text{.AOI strg}\\ \text{.OAI strg}\end{array}\right\}$ $[t_r,m_r,\ t_f,m_f]$ [load*] $i_1 \ldots i_n\ \ j_1 \ldots j_k\ \ldots$

Unidirectional Wired Connection

name .UCONN $[t_r,m_r,\ t_f,m_f]$ [load*] $i_1\ i_2\ i_3 \ldots i_n$

User-Defined Unidirectional Gate

name .GATn $[t_r,m_r,\ t_f,m_f]$ [load*] $i_1\ i_2 \ldots i_4$

Flip-Flop Models

Setup-and-Hold Detectors

name $\left\{\begin{array}{l}\text{.SHNE}\\ \text{.SHPE}\end{array}\right\}$ $[t_s\ t_h]$ [load*] clock data

Flip-Flop Models

name $\left\{\begin{array}{l}\text{.DNEFF strg}\\ \text{.DPEFF strg}\end{array}\right\}$ $[t_r,m_r,\ t_f,m_f]$ [load*] clk D set clear q qb

name $\left\{\begin{array}{l}\text{.JKNEFF strg}\\ \text{.JKPEFF strg}\end{array}\right\}$ $[t_r,m_r,\ t_f,m_f]$ [load*] clk J K set clear q qb

name $\left\{\begin{array}{l}\text{.SRNEFF strg}\\ \text{.SRPEFF strg}\end{array}\right\}$ $[t_r,m_r,\ t_f,m_f]$ [load*] clk S R set clear q qb

name $\left\{\begin{array}{l}\text{.TNEFF strg}\\ \text{.TPEFF strg}\end{array}\right\}$ $[t_r,m_r,\ t_f,m_f]$ [load*] clk T set clear q qb

Zycad-Compatible Components

name .DFF strg $[t_r\ t_f]$ [load*] data clock reset
name .LAT strg $[t_r\ t_f]$ data enable reset

Switch Level Primitives

Capacitance and Resistor Specification

name **.CAP** value

name **.RES** [t_r,m_r t_f,m_f] [load*] input

MOS Logical Transistors

name $\left\{\begin{array}{l}\textbf{.NMOS}\\ \textbf{.PMOS}\end{array}\right\}$ [t_{on} t_{off}] [load*] ng nd ns

name **.CMOS** [t_{on} t_{off}] [load*] ngn ngp nd ns

Unidirectional Transfer Gates

name $\left\{\begin{array}{l}\textbf{.NXFR}\\ \textbf{.NRXFR}\end{array}\right\}$ [t_{on},m_{on} t_{off},m_{off}] [load*] en input

name $\left\{\begin{array}{l}\textbf{.PXFR}\\ \textbf{.PRXFR}\end{array}\right\}$ [t_{on},m_{on} t_{off},m_{off}] [load*] ep input

name $\left\{\begin{array}{l}\textbf{.CXFR}\\ \textbf{.CRXFR}\end{array}\right\}$ [t_{on},m_{on} t_{off},m_{off}] [load*] en ep input

Memory Models

Random-Access Memory

name **.RAM** [t_r t_w] [load*] er ew a_1 a_2 a_3 . . . a_n
\+ d_1 d_2 d_3 . . . d_k
\+ [iii. . . ooo. . . iii. . . ooo. . .]

Read-Only Memory

name **.ROM** [t_r] [load*] a_1 a_2 . . . a_n
\+ o_1 o_2 . . . o_k
\+ iii. . . ooo. . . iii. . . ooo. . . iii. . . ooo. . .

Unidirectional Random-Access Memory

name **.URAM** [t_r t_w] er ew [load*] a_1 a_2 a_3 . . . a_n
\+ di_1 di_2 di_3 . . . di_k
\+ do_1 do_2 do_3 . . . do_k
\+ [iii. . . ooo. . . iii. . . ooo. . .]

Programmable Logic Array

.PLA .itype strg [t_r t_f] .otype strg [t_r t_f]
\+ [load*] iname iname . . .
\+ oname oname. . .
\+ $xname_0$ s s s s s s s s s . . . s
\+ $xname_1$ s s s s s s s s s . . . s
. . .
.EOP

Befehle für die Stimulibeschreibung

Stimulus Specification

Clock Specification

name **.CLK** t_0 sc_0 t_1 sc_1 . . . t_n sc_n [**.REP** t_r] . . .

Pattern Input With Pattern-Macro Expansion

.PATTERN [d_t,t_0] name name . . .
+ name . . . name. . .
[t_0] bh bh bh bh . . . bh
[t_1] bh bh bh bh . . . bh
. . . .
pmname/
[t_2] bh bh bh bh . . . bh
. . . .
pmname/
. . . .
[t_n] bh bh bh bh . . . bh
.EOP

Note: sc = the below state-type codes:

S1	Supply High	**D1**	Driving High
S*	Supply Unknown	**D***	Driving Unknown
S0	Supply Low	**D0**	Driving Low
R1	Resistive High	**Z1**	High-Z High
R*	Resistive Unknown	**Z***	High-Z Unknown
R0	Resistive Low	**Z0**	High-Z Low
1D	Decay High	***S**	Spike State
***D**	Decay Unknown		
0D	Decay Low		

Diverse Befehle und Macro-Definitionen

Timing and Gate Instructions

Fanout Loading

.LOAD name = value name = value . . . [**.DEF** = value] . . .

Default Delays and Delay Override

.DELAY .type = d_1,m_1 d_2,m_2 . . . [**.DEF** = d_1,m_1 d_2,m_2]

Delay Scaling Factors

.DMAX .type = pct .type = pct . . . [**.DEF** = pct] . . .
.DMIN .type = pct .type = pct . . . [**.DEF** = pct] . . .

Type Specification for Gates

.TYPE .type strg .type strg . . .

User Defined Gate Definition

.GDEF n iiii = sc iiii = sc iiii = sc . . .

Macro Capabilities

Macro Definition

.MACRO mname [#d_1 #d_2 . . . #d_n] e_1 e_2 e_3 . . . e_k
xxxxxxxxxxxx
.EOM [mname]

Macro Expansions

(name mname v_1 v_2 . . . v_n r_1 r_2 r_3 . . . r_k
<name mname v_1 v_2 . . . v_n r_1 r_2 r_3 . . . r_k

Pattern-Macro Definition

.PMAC pmname snum [**.BIN**] [**.HEX**]
bh bh bh bh bh . . . bh
.EOP

Global Nodes

GLOBAL name name . . .

Library File Specification

.LIBRARY **.MACLIB** = filename **.SYSLIB** = filename

Beispiel für ein Simulationsfile

```
$
$ Beispieldatei fuer eine 4-Bit Synchronzaehler
$
.TITLE 4-Bit Zaehler
.GLOBAL RESET
.GLOBAL SET
$
$ Netzwerkbeschreibung
$
FF1 .JKPEFF CLK HI HI RESET /Q1 NQ1
FF2 .JKPEFF CLK Q1 Q1 SET RESET /Q2 NQ2
FF3 .JKPEFF CLK J3K3 J3K3 SET RESET /Q3 NQ3
FF4 .JKPEFF CLK J4K4 J4K4 SET RESET /Q4 NQ4
J3K3 .AND Q1 Q2
J4K4 .AND Q1 Q2 Q3
$
$ Stimulibeschreibung
$
HI .CLK 0 S1
RESET .CLK 0 S1 10 S0 20 S1
SET .CLK 0 S1
CLK .CLK 0 S0 50 S1 70 S0 90 S1 .REP 50
```

Beschreibung zur Bedienung der Software SILOS III

SILOS III läuft unter Windows 3.1. Wenn Sie sich im Programm-Manager befinden doppelklicken Sie auf das ICON Silos. Hierin befinden sich drei ICONS: SILOS III, SDA und Editor. SILOS III ist das eigentliche Programm für die Simulation. SDA (= Silos Data Analyser) ist der Graph, mit dem man sich die Simulationsergebnisse anschauen kann. SDA kann jedoch auch direkt aus SILOS III aufgerufen werden. Mit dem Editor kann das ASCII-Programmierfile erstellt werden.

Gehen Sie nun wie folgt vor:

Erstellen Sie Ihr ASCII-Programmierfile mit dem Editor, in dem Sie auf das Editor-Icon doppelklicken. Der Editor ist schon auf das Directory c:\labor\vers6\ eingestellt.

Wenn Sie Ihr ASCII-File abgespeichert haben, dann starten Sie SILOS III durch doppelklicken. Zunächst sollten Sie sich das Verzeichnis c:\labor\vers6\ als Arbeitsverzeichnis einstellen:

Menü File → New (input files) → Working directory

Für jedes seperate Simualtions-Projekt sollte ein eigenes Verzeichnis angelegt sein und das Workingdirectory sollte dann immer auf das entsprechende Projekt-Verzeichnis zeigen !

Lesen Sie als nächstes Ihr ASCII-Programmierfile ein:

Menü File → New (input files) → Input ...
Danach wählen Sie ein File aus und drücken den Button *Input.*

Falls das ASCII-File Fehler enthält kommt eine BOX mit Fehlermeldung nach dem Einlesen hoch. Nach dem ersten Start von SILOS III gibt es leider noch keine Error-Datei im Workingdirectory. Deshalb müssen Sie SILOS III noch mal ganz verlassen (*Menü File → Exit*). Im Working-Directory wird eine zentrale save.err - Datei angelegt, in die jeder Fehler eines x-beliebigen ASCII-Programmierfiles abgelegt wird. Nach nochmaligem Aufruf kann man sich nach dem Einlesen mit dem *Menü Reports → Errors* die Fehler anschauen.

Falls das ASCII-File keine Fehler enthält schliessen Sie die Inputbox mit dem *Close-Button.*

Starten Sie nun die Simulation wie folgt:

Menü Logic Sim → New Run Logic Simulation oder *Taste F5*

Hier müssen Sie einen Zeitbereich eingeben von z.B. 0 bis 200.
Ist die Simulatin beendet, so kann der Graph aufgerufen werden. Dies geschieht mit:

Menü Analyzer → Start Analyzer

Wird der Analyzer das erste Mal in einem Working-Direktory gestartet, so kommt ein Box hoch, in der die vorhandenen Signale der Simualtion vorhanden in der rechten Spalte vorhanden sind. Signale, die man sich im Graph anschauen will muss man sich nun selektieren und mit *ADD* in die linke Spalte hinzufügen. Mit dem *OK-Button* kann das Fenster verlassen werden, und die Signale werden im Graph angezeigt.

Wird der Analyzer zum zweiten Mal in einem Working-Directory aufgerufen, so kann es sein, daß sich im linken Fenster noch alte Signalnamen oder nicht mehr vorhandene Signalnamen befinden, die man mit Selektieren und *Delete* entfernen kann.

Befindet man sich nun im Graphen-Fenster, so kann man neue Signale hinzufügen, in dem man das *Menü Select → Display List* aufruft. Im Graphen-Fenster kann man Signale auch mit der linken Maustaste selektieren (am linken Rand) und dahin schieben, wo man sie haben will, um eine vernünftige Anordnung der Signale zu haben.

Das Programm SILOS III ist weitestgehend selbsterklärend, so daß an dieser Stelle auf eine weitere Erläuterung der einzelnen Programmpunkte verzichtet werden soll. Spielen Sie ruhig etwas mit den einzelnen Menüs herum, um sich mit den verschiedenen Möglichkeiten vertraut zu machen.