

4 Systembus- und Speicherorganisation

4.1 Datenübertragung mit den Systemkomponenten

4.1.1 Adressierung prozessorexterner Speicherzellen und Register

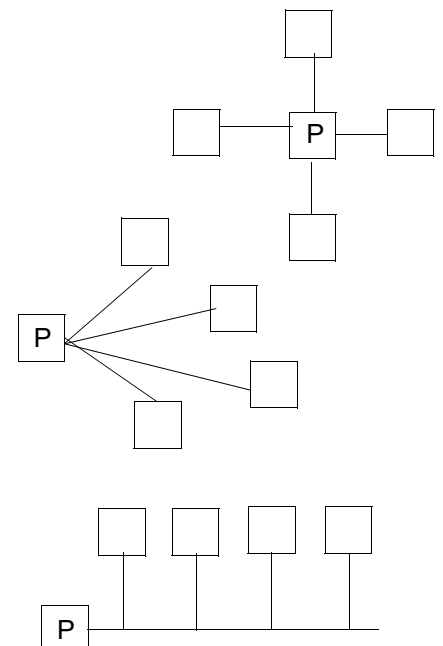
In technischer Terminologie bezeichnet man alle sog. aktiven Systemkomponenten, wie Prozessoren und Ein-/Ausgabecontroller, als Master und alle passiven Systemkomponenten, wie Speichereinheiten oder Ein-/Ausgabeeinheiten (soweit sie nicht Steuerungsfunktion haben), als Slaves. Alle diese Systemkomponenten besitzen Speicherzellen oder Register, auf die ein ausgezeichnete Master im System, das ist in der Regel der (zentrale) Prozessor, zugreifen können muß. Die Speicherzellen bzw. Register dienen zwei Funktionen:

Master
Slave

1. zur längerfristigen oder kurzfristigen Speicherung von Daten (Datenspeicher, Datenregister), dann werden sie als „Lager“ bzw. „Puffer“ vom ausgezeichneten Master im *Verlauf* einer Datenübertragung angesprochen;
2. zur Speicherung von Steuerungs- oder Statusinformation (Controlregister, Statusregister), dann werden sie vom ausgezeichneten Master in erster Linie zur *Initialisierung*, aber auch zur *Überwachung* einer Datenübertragung angesprochen.

Busstruktur

Ein einfacher Prozessor ist so konstruiert, daß er zu einer Zeit nur mit einer einzigen der Systemkomponenten kommunizieren kann, genauer, mit einem einzigen ihrer Speicherplätze. Deshalb ist es sinnvoll, alle Systemkomponenten an den Prozessor unmittelbar anzuschließen, genauer, ihn mit den Systemkomponenten an einem einzigen Punkt zu verbinden. In Wirklichkeit ist dieser „Sammelpunkt“ eine „Sammelschiene“, ein sog. Bus, der Systembus, mit prozessorseitig monodirektionalen Adreßleitungen, dem Adreßbus A, und bidirektionalen Datenleitungen, dem Datenbus D. Des weiteren gibt es monodirektionale und bidirektionale Steuerleitungen, die zusammengefaßt als Steuerbus bezeichnet und mit C (control) abgekürzt werden. Auch sie sind Teil des Systembusses. (Genau genommen



sind nicht die Leitungen monodirektional oder bidirektional, sondern die auf ihnen entweder nur in einer bzw. in beiden Richtungen „laufenden“ Signale).

Signale: $A_{n-1:0}$ (address)

$D_{m-1:0}$ (data)

Bild 4-1 zeigt die Konfiguration eines solchen einfachen Systems mit einem zentralen Prozessor (links) und weiteren Systemkomponenten (oben). Bei diesen handelt es sich um Speichereinheiten (mit sehr vielen Speicherzellen) und Ein-/Ausgabeeinheiten (mit einigen wenigen Registern); zu letzteren zählen Schnittstellen- oder Interface-Adapter (kurz Interfaces oder Adapter) sowie Steuereinheiten (Controller). – Über die Steuerleitungen des Systembusses ist im Moment nichts ausgesagt, sie sind im Bild auch nicht eingezeichnet; sie werden erst eingeführt, wenn sie im Laufe der folgenden Abschnitte benötigt werden.

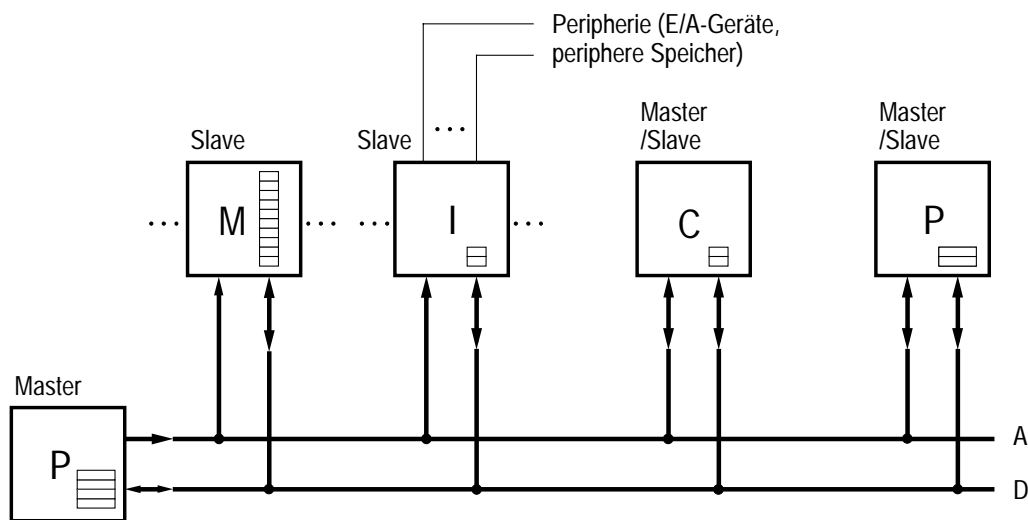


Bild 4-1. Prinzipstruktur eines einfachen Rechnersystems; P Prozessor, M Speicher (Memory), I Ein-/Ausgabeeinheit (Interface-Adapter), C Steuereinheit (Controller); A Adreßbus, D Datenbus als die beiden wichtigsten Teile des Systembusses. Die kleinen Kästchen symbolisieren Speicherzellen bzw. Register.

Anwahl der Systemkomponenten

Mit Ausnahme des zentralen Prozessors (der immer Master ist) werden alle am Bus angeschlossenen Systemkomponenten durch je ein Signal individuell ausgewählt. Diese individuelle Anwahl ist aus technischen Gründen notwendig, um z.B. unterschiedliche Reaktionszeiten der Systemkomponenten berücksichtigen zu können. Ebenfalls aus technischen Gründen sind die Systemkomponenten über Verstärker, sog. Treiber, an den Bus angeschlossen, die ihrerseits mit denselben Anwahlsignalen wie die zugehörigen Komponenten angesteuert werden. Diese Anwahlsignale (select, enable) werden entweder aus den Adreßsignalen allein oder aus den Adreßsignalen und speziellen Steuersignalen des Prozessors gewonnen. Wie das im einzelnen geschieht, ist in 4.2.1 beschrieben (siehe die Bilder 4-10 ff.).

Neben der Auswahl muß auch die Richtung für die Datenübertragung vom Master, in unserem Fall vom Prozessor, vorgegeben werden. Während aber zur *Anwahl* für *jede* andere Systemkomponente (ein Slave oder ein als Slave arbeitender Master¹) genau ein Signal bereitgestellt werden muß, genügt es, für die *Richtung* der Datenübertragung vom Master zum Slave hin nur ein einziges, für *alle* Komponenten gemeinsames Signal vorzusehen (read/write). Das ist deshalb erlaubt, weil funktionell gesehen neben einem Prozessorregister als dem einen Verbindungspunkt sowieso immer nur ein einziger Speicherplatz als anderer Verbindungspunkt bei einer Datenübertragung über den Systembus aktiviert ist.

Signale: S_i (select)

R/\overline{W} (read/write)

Bild 4-2 zeigt den Anschluß von einer Speicher- und zwei Ein-/Ausgabeeinheiten an den Systembus unter Verwendung einer bestimmten Symbolik. Darin ist für die Systemkomponenten bewußt nicht die übliche technisch orientierte Kästchendarstellung, sondern eine andere, funktionell betonte Darstellung gewählt worden, in der jede Einheit durch ihre Speicherzellen bzw. Register und die zugehörige Auswahllogik (ihre Decodierer) repräsentiert ist. – Diese Interpretation des Bildes mit den drei gezeichneten Einheiten ist aber nicht die einzig mögliche; vielmehr können sich z.B. hinter einem Speichersymbol mehrere Speichereinheiten verbergen, oder umgekehrt können mehrere Speichersymbole als eine einzige Speichereinheit aufgefaßt werden. – Der Prozessor ist in diesem Bild wie auch in den folgenden Blockbildern dieses Kapitels nur durch seine Schnittstelle und die im jeweils betrachteten Zusammenhang wesentlichen Signale dargestellt.

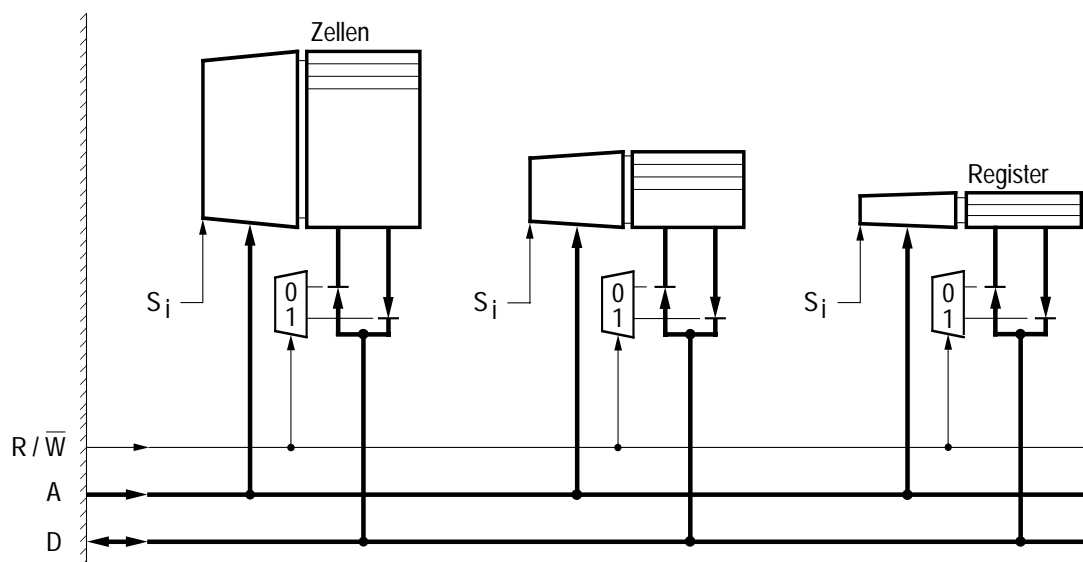


Bild 4-2. Auswahl von Systemkomponenten (physisch gesehen) oder Speicherbereichen (logisch gesehen) einschließlich Vorgabe der Übertragungsrichtung durch das Read/Write-Signal.

1. Wir bezeichnen im folgenden den Master in diesem Betriebsfall auch als Slave.

4.1.2 Abläufe für den Datentransport zwischen Prozessor und Speicher

Buszyklus
Lesen
Schreiben

In technischer Terminologie bezeichnet man den Transport eines Datums auf dem Bus als Buszyklus, den Transport Slave → Master als Lesezyklus bzw. Lesen und den Transport Master → Slave als Schreibzyklus bzw. Schreiben. Zum Lesen und Schreiben prozessorexterner Speicherplätze ist es notwendig, die Abläufe in den beteiligten Systemkomponenten in korrekte zeitliche Beziehungen zueinander zu setzen. Zur Beschreibung dieser Synchronisationsaufgabe benutzen wir beispielhaft

- den *Prozessor* als Master und den *Speicher* als Slave.

Die Herstellung der gewünschten zeitlichen Ordnung erfolgt über mehrere spezielle monodirektionale Steuerleitungen.

Abstimmung der Handlungen

Zugriffszeit
access time

Die Adressierung prozessorexterner Speicherplätze sowie deren Bereitschaft zum Übernehmen (Schreiben) bzw. Übergeben (Lesen) eines Datums an den Prozessor benötigt eine gewisse Zeit (Zugriffszeit, access time). Sofern diese Zeit für alle am Bus angeschlossenen Komponenten annähernd gleich ist und sich während der Installation des Systems nicht ändert, genügt es, nach der Bereitstellung der Adresse im Mikroprogramm des Prozessors eine feste Wartezeit von einer Dauer größer/gleich der Zugriffszeit vorzusehen. Dies erübrigt sich natürlich bei schnellen Speichern, nämlich dann, wenn der Prozessortakt länger als die Zugriffszeit ist. Sofern die Zugriffszeit jedoch für die einzelnen Systemkomponenten unterschiedlich groß ist oder irgendwann Einheiten mit zum Zeitpunkt des Prozessorentwurfs unbekannter Zugriffszeit angeschlossen werden sollen, muß der Prozessor nach einer bestimmten Anzahl von Takten, sog. Wartetakten (wait states), mit der Einheit zur Datenübernahme (beim Schreiben) bzw. Datenübergabe (beim Lesen) synchronisiert werden.

Petri-Netz

Bild 4-3 zeigt die beschriebenen beiden Fälle in einer als Petri-Netz bezeichneten Darstellung hoher Abstraktion, und zwar beispielhaft

- für das *Lesen* eines Datums aus dem *Speicher*.

Darin sind die beiden im Prozessor und im Speicher ablaufenden Prozesse durch zwei in bestimmter Weise miteinander verbundene zyklische Graphen dargestellt. Die Graphen bestehen wie gewöhnlich aus Kreisen für die Prozeßzustände („Plätze“, „Stellen“) und aus Strichen oder Kästchen für die Prozeßaktionen („Balken“). In jedem der Graphen befindet sich genau eine Marke, deren Positionen den jeweiligen Stand der Prozesse widerspiegeln und beim Übergang über einen Balken auf den jeweils nächsten Platz bzw. die nächste Stelle die dazwischenstehenden Aktionen auslösen.

Beide Prozesse sind zur Abstimmung ihrer Handlungen durch die Balken miteinander verbunden, so daß aus den Einzelprozessen ein Prozeßnetz, das Petri-Netz,

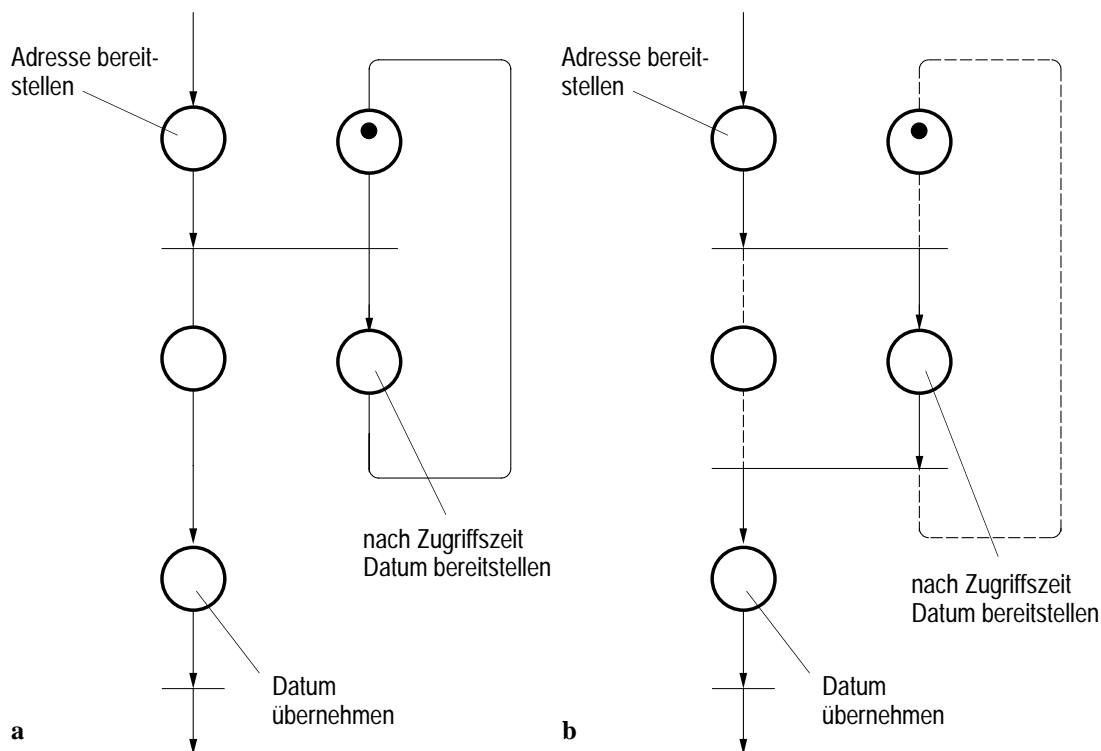


Bild 4-3. Festlegung einer zeitlichen Ordnung der Aktionen von Prozessorsteuerung und Speichersteuerung für das Lesen eines Datums (Transport Speicherplatz → Prozessor; links Prozessor, rechts Speicher); **a** Speicherzugriffszeit konstant, **b** Speicherzugriffszeit variabel. Die ablaufbestimmenden Prozesse in b sind ausgezogen gezeichnet, um ihr Wechselspiel zu verdeutlichen.

entsteht. In einem Petri-Netz gilt die Regel, daß ein (gemeinsamer) Balken nur dann von einer Marke überschritten werden darf, wenn alle Plätze vor dem Balken mit Marken besetzt sind (und alle Plätze hinter dem Balken leer sind). Eine einen Balken überschreitende Marke nimmt dabei alle anderen Marken mit, wodurch die für ein Petri-Netz typische absolute Gleichzeitigkeit von Zustandsübergängen mehrerer Prozesse als „unteilbare Handlung“ ausgedrückt wird. – Jede einzelne Gesamtkonstellation von Marken bildet einen Gesamtzustand des Netzes. Alle Gesamtzustände zusammengekommen beschreiben alle erreichbaren Markenkonstellationen im Netz. Deren Verbindungen bildet den Erreichbarkeitsgraphen (der in unserem Zusammenhang keine Rolle spielt).

Die drei in Bild 4-3 vorkommenden Aktionen werden in den folgenden Bildern durch $\dots \rightarrow A$, $D \leftarrow M[A]$ und $\dots \leftarrow D$ abgekürzt, nämlich unter der Vorstellung der Datenübertragung mit einem Speicher der Bezeichnung M (memory) mit einer Adresse auf A und dem Datum auf D. – Die Pünktchen stehen für „Adresse von irgend wo her im Prozessor“ bzw. „Datum irgend wo hin im Prozessor“.

Die beiden in Bild 4-3 dargestellten Netze bringen anschaulich zum Ausdruck, wie die Einhaltung der geforderten Reihenfolge der Prozeßaktionen erreicht wird, nämlich in Teilbild a durch die Beachtung der Zeitbedingung bzw. in Teilbild b durch den zweiten Synchronisationsbalken. – Ein vollständiger Markendurchlauf bildet einen Buszyklus, im hier beschriebenen Fall einen Lesezyklus.

Synchronisation über Signale

Aus Bild 4-3 geht zwar die durch die Synchronisation festgelegte Reihenfolge der Aktionen beider Prozesse hervor, es ist durch die Strichelung aber nur angedeutet, in welcher Situation welcher Prozeß eine aktive (tätige) und welcher eine passive (untätige, d.h. wartende) Rolle spielt. Für eine theoretische Beschreibung der Funktionsweise genügt das; für den praktischen Entwurf einer Logikschaltung jedoch nicht.

Um diesem Mangel hinsichtlich einer Realisierung abzuhelpfen, wird für jeden Prozeß ein Signal eingeführt, das vom jeweils aktiven Prozeß aktiviert wird und den jeweils passiven Prozeß aus seinem Wartezustand befreit. Das heißt, der Prozessor als diejenige Systemkomponente, die die Datenübertragung einleitet (Master), weckt den Speicher als die darauf wartende Komponente (Slave) mit einem Alarm-/Wecksignal (alert), Richtung Prozessor → Speicher. Daraufhin startet die Speichersteuerung ein Zeitglied oder einen Zähler oder durchläuft einige zusätzliche Zustände (Wartezustände) und liefert nach einer Dauer t_a dem Prozessor ein Bereitsignal (ready), Richtung Prozessor ← Speicher. – Man bezeichnet die beiden Signale als Handshake-Signale: sie laufen sozusagen auf den Steuerleitungen zwischen den Datenübertragungspartnern „hin“ und „her“.

Signale: AL (alert)

RY (ready)

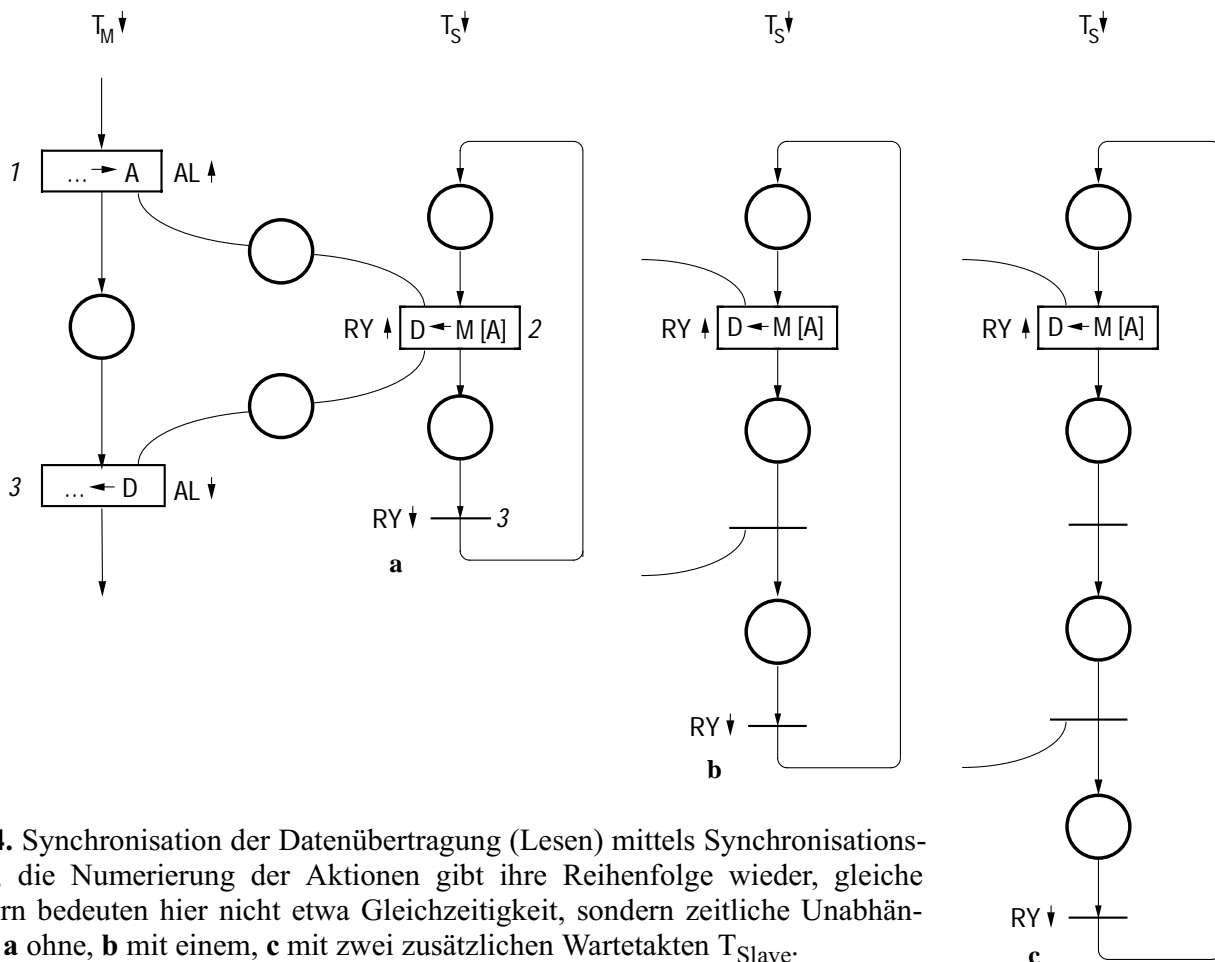


Bild 4-4. Synchronisation der Datenübertragung (Lesen) mittels Synchronisationsflipflop; die Numerierung der Aktionen gibt ihre Reihenfolge wieder, gleiche Nummern bedeuten hier nicht etwa Gleichzeitigkeit, sondern zeitliche Unabhängigkeit; **a** ohne, **b** mit einem, **c** mit zwei zusätzlichen Wartetakten T_{Slave} .

Ein erster Entwurf. Bild 4-4 zeigt korrespondierend zu Bild 4-3b den geschilderten Vorgang für das Lesen als detaillierteres Petri-Netz unter Berücksichtigung der Steuersignale. Darin ist davon ausgegangen, daß die Prozessorsteuerung (das Mikroprogramm) dem Prozessortakt T_{Master} folgt und die Speichersteuerung einen eigenständigen, von T_{Master} völlig unabhängigen Takt T_{Slave} hat.

Würde man nun aus dem gezeigten Petri-Netz unmittelbar eine Logikschaltung entwickeln, so entstünde eine Realisierung mit einem Synchronisationsflipflop, d.h., die Steuersignale würden *indirekt* von den Steuerwerken ausgewertet; wir bezeichnen diese Art der Implementierung als explizite Synchronisation. – Dieser Weg wird jedoch beim Buszyklus üblicherweise nicht beschritten; stattdessen wird das Kommunikationsprotokoll neu definiert.

explizite
Synchroni-
sation

Ein zweiter Entwurf. Bild 4-5 zeigt das gegenüber Bild 4-4 geänderte Kommunikationsprotokoll, entstanden durch willkürliches, weiteres „Legen einer Spur“ mit dem Ziel, eine bestimmte Reihenfolge der in Bild 4-4 noch „offenen“ Aktionen zu erzwingen. Dabei handelt es sich um die dort in ihrer Reihenfolge unbestimmten Aktionen 3, die hier nun – wie eingetragen – in die Reihenfolge 3, 4 gebracht werden.

Wird für dieses neue Petri-Netz ein Logikschaltungsentwurf durchgeführt, so entsteht eine Realisierung ohne ein Synchronisationsflipflop, d.h., die Steuersi-

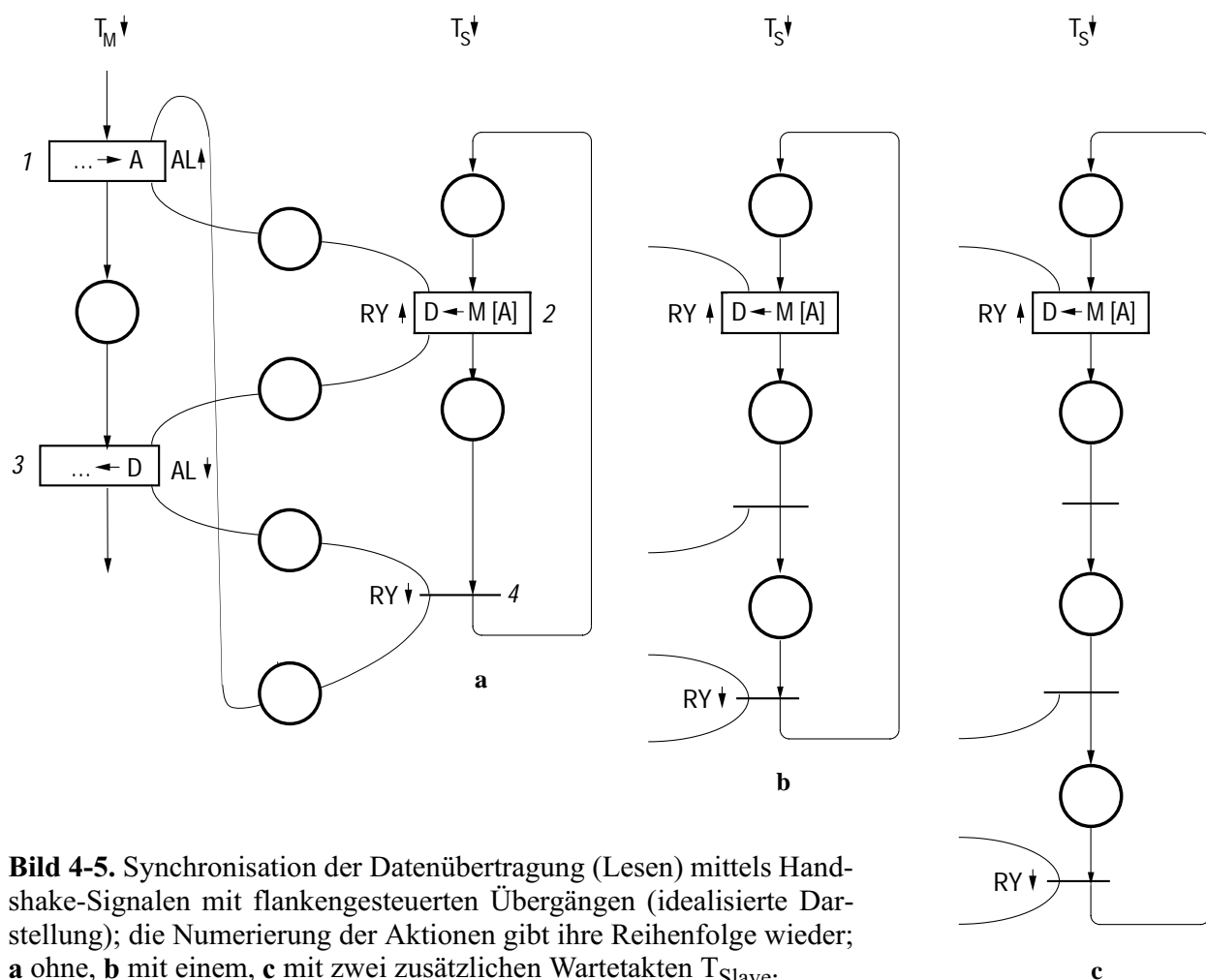


Bild 4-5. Synchronisation der Datenübertragung (Lesen) mittels Handshake-Signalen mit flankengesteuerten Übergängen (idealisierte Darstellung); die Numerierung der Aktionen gibt ihre Reihenfolge wieder; **a** ohne, **b** mit einem, **c** mit zwei zusätzlichen Wartetakten T_{Slave} .

gnale werden von den Steuerwerken *unmittelbar* ausgewertet; wir bezeichnen diese Art der Implementierung als implizite Synchronisation. – Diese Entwurfsentscheidung impliziert natürlich neben geänderter Speichersteuerung auch eine geänderte Prozessorsteuerung und ist somit Teil des Prozessorentwurfs. Für einen im Handel befindlichen Prozessor, an den ein Speicher angeschlossen werden soll, besteht dieser Freiheitsgrad nicht. Für ihn muß sich der Entwicklungsingenieur an das vorgeschriebene Kommunikationsprotokoll halten.

Systemstruktur Prozessor/Speicher

Für die Beschreibung der asynchronen und der synchronen Datenübertragung (nachfolgend 4.1.3 bzw. 4.1.4) legen wir die in Bild 4-6 dargestellte Systemstruktur zugrunde. Am Systembus ist neben dem Prozessor eine Speichereinheit angeschlossen (stellvertretend auch für andere Systemkomponenten mit prozessorexternen Speicherplätzen). Das Alertsinal muß für jeden Slave individuell gebildet werden. Das geschieht durch Und-Verknüpfung des Signals S_i^1 mit einem vom Prozessor ausgehenden Steuersignal mit der über die Synchronisation hinausgehenden Funktion, die Gültigkeit der ausgegebenen Adreßbits auf dem Adreßbus anzuzeigen (address strobe). – Die Speichersteuerung ihrerseits liefert neben dem Ready-Signal für den Prozessor ein weiteres Steuersignal, und zwar für den Speicher(chip) zum Durchschalten des Datenbusses ((chip) enable).

Signale: AS (address strobe)

EN (enable)

In Bild 4-6a sind die Signale eingezeichnet. Die zeitliche Abstimmung zwischen dem Prozessor und dem Speicher erfolgt für die im nächsten Abschnitt beschriebene asynchrone Datenübertragung mittels der Handshake-Signale AS/AL und RY. Für die im übernächsten Abschnitt beschriebene synchrone Datenübertragung kommt noch eine in Bild 4-6a nicht eingezeichnete gemeinsame Taktleitung als Bestandteil des Busses hinzu. – Bild 4-6b zeigt zwei Kurzdarstellungen, wie sie künftig in diesem Buch als Abstraktion von Teilbild a verwendet werden, – wie man sieht – beide mit Unterdrückung der Speichersteuerung und der Anwahllogik sowie des Richtungs- und des Strobesignals. Die erste Darstellung *mit* dem Anwahlsignal S_i wird dann benutzt, wenn es auf die Anwahl der Systemkomponenten ankommt; die zweite Darstellung *ohne* S_i wird hingegen dann benutzt, wenn die Anwahl der Systemkomponenten als Implementierungsdetail nicht in Erscheinung treten soll. – Bild 4-6 gilt nicht nur für das Lesen, wie hier standardmäßig betrachtet, sondern auch für das Schreiben.

Realisierung des Buszyklus

Bild 4-7 zeigt den entworfenen Lesezyklus mit Handshaking nun in der Darstellung vernetzter Graphen. Diese Darstellung – im folgenden zur Unterscheidung

1. Woher dieses kommt, siehe wieder Bilder 4-10 ff.

von Petri-Netzen als Graphennetz bezeichnet – dient gewissermaßen als Basis für die später zu erörternden, praktisch vorkommenden Buszyklen. Sie enthält nicht mehr – theoretischer Darstellung folgend – die im Petri-Netz wirksamen *Signalflanken*, sondern nun – technischer Darstellung folgend – die im Graphennetz wirkenden *Signalpegel*.¹ Letztere Darstellung wird zur Verwirklichung durch eine elektronische Schaltung benötigt: Signalflanken gehen nämlich von Zustandsübergängen aus und bewirken – eben theoretisch – absolut *gleichzeitig* Zustandsübergänge im anderen Graphen; Signalpegel gehen von den Zuständen selbst aus und bewirken – nun technisch – unmittelbar *folgend* Zustandsübergänge im anderen Graphen. – Wir benutzen also Petri-Netze auf der höheren Ab-

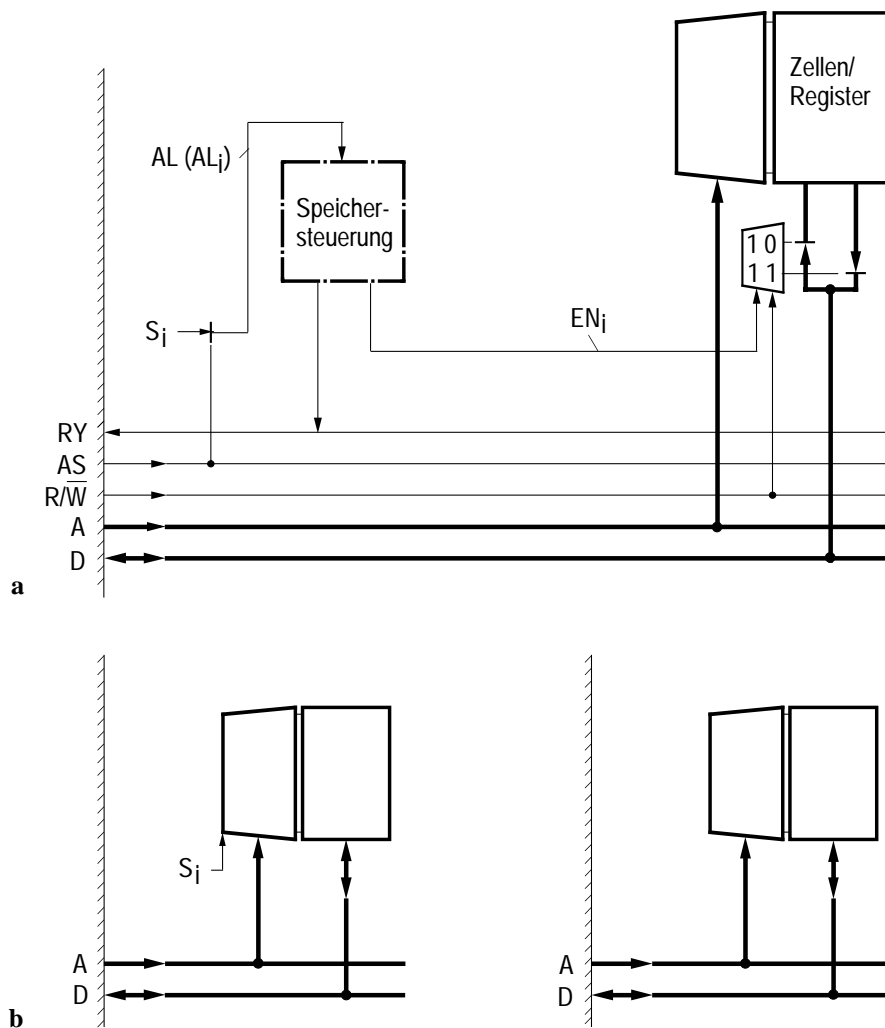


Bild 4-6. Anschluß einer Speichereinheit an den Systembus; **a** Blockbild, **b** abstrahierte Darstellungen mit bzw. ohne Anwahlleitung. Da immer mehrere Buskomponenten angeschlossen sind, bedarf es eines individuellen $AL_i = AS \cdot S_i$ (S_i wird mit dem Strobe-Signal „durchgeschaltet“). In den folgenden Graphen benutzen wir jedoch der Kürze halber weiter lediglich AL. RY wird als allen Buskomponenten gemeinsames Signal durch verdrahtetes Oder (wired or) gewonnen.

1. Zur besseren Unterscheidung kennzeichnen wir Signale mit Großbuchstaben, wenn sie als Ausgänge wirken (gesendet werden), und mit Kleinbuchstaben, wenn sie als Eingänge wirken (empfangen werden).

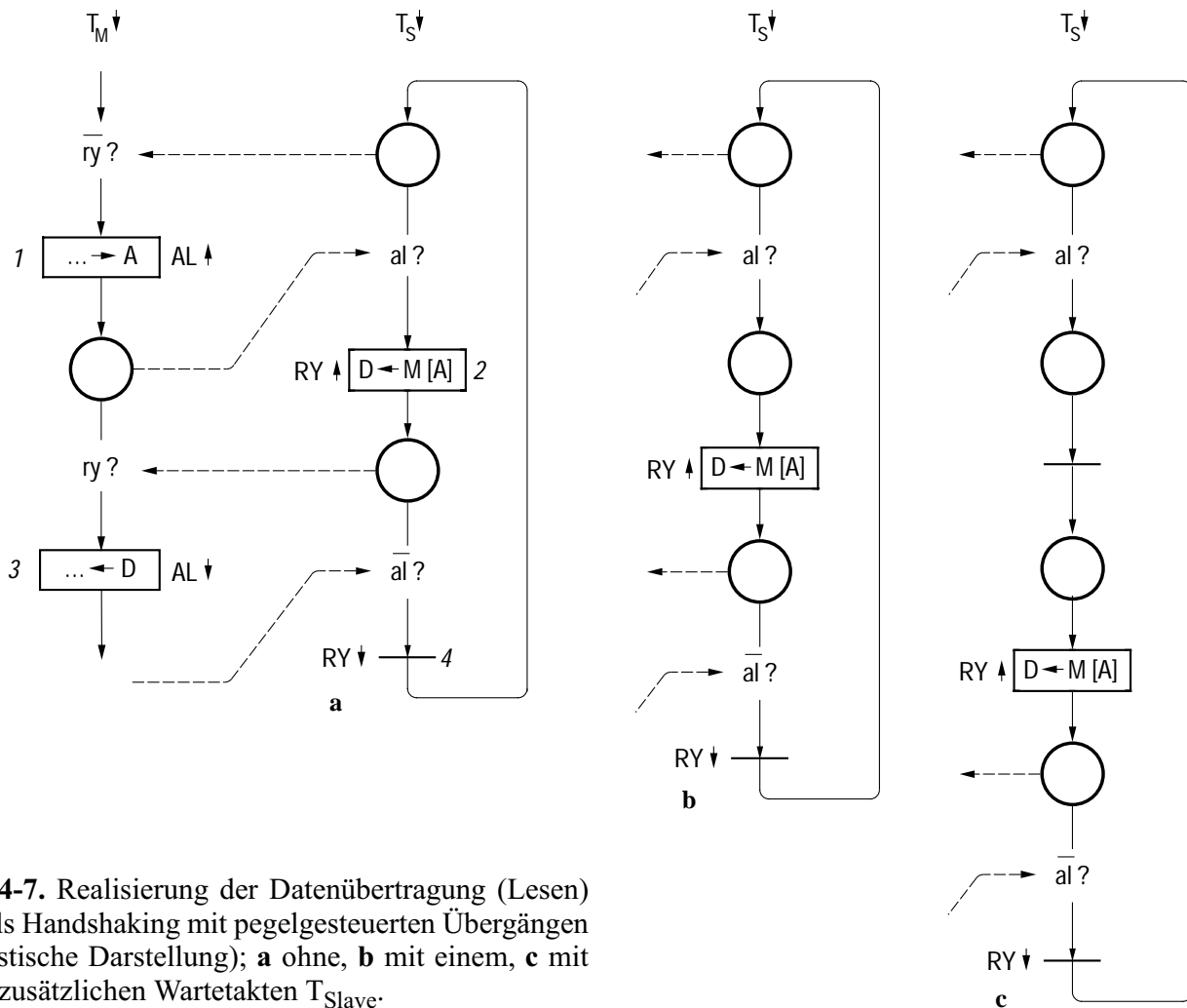


Bild 4-7. Realisierung der Datenübertragung (Lesen) mittels Handshaking mit pegelgesteuerten Übergängen (realistische Darstellung); **a** ohne, **b** mit einem, **c** mit zwei zusätzlichen Wartetakten T_{Slave} .

straktionsebene der Festlegung der Funktionsweise und Graphennetze auf der niedrigeren Abstraktionsebene der Entwicklung von Logikschaltungen.

Die ordnungsgemäße Funktionsweise des in Bild 4-7 wiedergegebenen Buszyklus ist durch das zahnradähnliche Ineinandergreifen der einzelnen Aktionen garantiert, gleichgültig wie schnell der eine und wie schnell der andere Prozeß fortschreitet, d.h., wie unterschiedlich schnell ihre Takte sind. Die einzuhaltende Abfolge der Aktionen kann aber auch anders, nämlich durch zusätzliche Bedingungen garantiert werden, wodurch die eine oder die andere Abfrage entfallen darf. Solche Bedingungen können auf zweierlei Weise eingebracht werden:

1. Vorausgesetzt, auslösende Flanken finden in Zuständen „über“ den ihnen zugeordneten Abfragen immer Marken vor, so daß sie tatsächlich auch immer mitgenommen werden (d.h., die Prozesse müssen sich in den jeweiligen Zuständen befinden, *bevor* die Signalflanken erscheinen), kann die \overline{ry} -Abfrage im Prozessor entfallen, es entsteht *asynchrone* Datenübertragung (siehe 4.1.3).
2. Vorausgesetzt, alle Marken werden von einem allen Systemkomponenten gemeinsamen Bustakt von Zustand zu Zustand gleichzeitig befördert (d.h., Marken werden auch ohne explizite Synchronisation garantiert weiterbewegt), kann neben der \overline{ry} -Abfrage im Prozessor auch die \overline{al} -Abfrage im Speicher entfallen, es entsteht *synchrone* Datenübertragung (siehe 4.1.4).

4.1.3 Asynchroner Buszyklus

Bei einem asynchronen Bus gibt es keine zentrale Taktleitung als Bestandteil des Busses. Die neben dem Prozessor am Bus angeschlossenen Systemkomponenten arbeiten entweder jeweils mit eigenem Takt (sofern sie in Sychronotechnik aufgebaut sind), oder sie kommen ganz ohne Takt aus (wenn sie in Asynchrontechnik aufgebaut sind). Für den ersten Fall des Buszyklus bei *getakteter* Speichersteuerung zeigt Bild 4-7 die Synchronisation. Für den zweiten, in der Praxis wichtigeren Fall des Buszyklus bei *ungetakteter* Speichersteuerung vereinfacht sich Bild 4-7, und es entsteht Bild 4-8.

asynchroner Buszyklus bei ungetakteter Speichersteuerung

Kennzeichnend für diese Art der Kommunikation der Systemkomponenten ist es, daß deren Handshake-Signale nur prozessorseitig von einem Takt abgetastet werden (Zustandsübergänge ausgelöst durch die Flanken des *Taktsignals*), während sie speicherseitig *unmittelbar* ausgewertet werden (Zustandsübergänge ausgelöst durch die Flanken des *Handshake-Signals*). – Die Synchronisation verläuft ordnungsgemäß, sofern die in Asynchrontechnik aufgebaute Speichersteuerung *unmittelbar* auf das Prozessorsignal AS/AL reagiert.

Bemerkung. Mit elektronischen Schaltungen lassen sich keine Flanken, sondern nur Pegel auswerten, so daß eine Flanke (als Übergang zwischen zwei Pegeln) praktisch immer nur mit ihrem neuen Pegel wirkt. In unserem Fall der Speichersteuerung wirken die Handshake-Signale also auch in Asynchrontechnik mit ihren Pegeln, aber im Gegensatz zu Sychronotechnik unmittelbar nach Erreichen ihres Aktivzustands. In dieser Technik wird immer vorausgesetzt, daß das steuernde Signal nach einer Änderung lange genug stabil bleibt, und zwar so lange, bis das gesteuerte Werk darauf definitiv reagiert, d.h. eindeutig einen stabilen Folgezustand erreicht hat. Erst dann darf sich das steuernde Signal erneut ändern und kann dadurch eine erneute Reaktion des gesteuerten Werkes auslösen. Die Änderung des Alertsignals zieht also unmittelbar die Reaktion der Speichersteuerung in der Art einer unteilbaren Handlung nach sich (anschaulich gesprochen: die

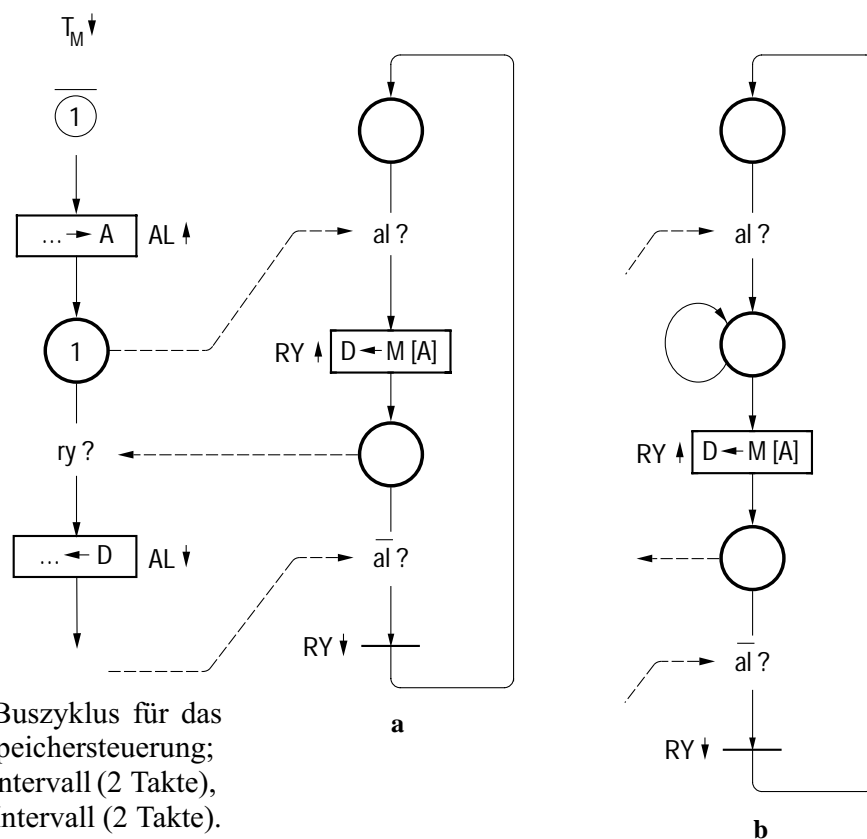


Bild 4-8. Asynchroner Buszyklus für das Lesen bei ungetakteter Speichersteuerung;
a Zugriffszeit $< T_{\text{Master-Intervall}}$ (2 Takte),
b Zugriffszeit $> T_{\text{Master-Intervall}}$ (2 Takte).

Speichersteuerung reagiert auf die *Flanke* des Alertsignals). In der Graphendarstellung können also zur besseren Veranschaulichung des Ablaufs die gestrichelten Synchronisationslinien zu den pegelgesteuerten asynchronen Zustandsübergängen auch von den auslösenden Flanken aus gezeichnet werden. – Zum Schaltungsentwurf in Asynchrontechnik siehe [Liebig/Thome].

Bild 4-8 ist gegenüber dem „Basis“-Bild 4-7 einfacher, da die Rücksynchronisation mit dem RY-Signal entfällt (oberer Rechts-links-Pfeil in Bild 4-7) und nur die Rücksynchronisation mit dem AL-Signal durchgeführt werden muß (unterer Links-rechts-Pfeil in Bild 4-7). Die Eingangsbedingung im Prozessorgraphen folgt unseren Darstellungskonventionen für Fließbandgraphen und bedeutet, daß nur bzw. erst eine neue Marke nach 1 gelangen kann, wenn in 1 keine Marke ist oder eine Marke 1 verlassen hat (Markenkonflikt). – Teilbild b zeigt mit dem zyklischen Pfeil im mittleren Zustand die Ausbildung der Zeitverzögerung in der Speichersteuerung durch ein Verzögerungsglied, das mit der positiven Flanke angestoßen wird und nach einer bestimmten Zeit auf 0 „zurückspringt“.

Bemerkung. In dieser Realisierung kann das Ausbleiben eines Signals zu einer weiteren Funktion, z.B. zur Systemüberwachung, genutzt werden. Meldet sich das System nämlich bei Adressierung eines nicht vorhandenen Speicherplatzes nicht mit $RY = 1$, so kann im Prozessor mit Hilfe eines auf einen Wert größer als die Zugriffszeit eingestellten Zeitgebers, eines sog. Watch-dog-Timers, ein Alarm in Gestalt eines Traps ausgelöst werden.

4.1.4 Synchroner Buszyklus

Bei einem synchronen Bus werden alle am Bus angeschlossenen Systemkomponenten, im hier betrachteten Fall der Prozessor und der Speicher, mit ein und demselben Taktsignal betrieben. Auch hier sind unterschiedliche Synchronisationstechniken möglich: Die Zustandsfortschaltung kann in beiden Komponenten wechselweise mit der positiven und der negativen Taktflanke erfolgen, oder sie kann gleichzeitig mit ein und derselben Taktflanke erfolgen. Zur Entwicklung entsprechender Buszyklen bildet wieder Bild 4-7 den Ausgangspunkt. Dadurch, daß die Handshake-Signalpegel prozessor- wie speicherseits von demselben Takt abgetastet werden, dürfen sowohl die Rücksynchronisation mit dem RY-Signal als auch die Rücksynchronisation mit dem AS-Signal entfallen.

Die Inspektion eines auf der Basis von Bild 4-7 konstruierten synchronen Buszyklus mit *gleichen* Taktflanken ergibt ein Minimum von 3 Takten. Um aber bei diesem wie beim asynchronen Buszyklus auf das Minimum von 2 Takten zu kommen, muß Bild 4-7 modifiziert werden, und zwar, indem die Abfrage $ry?$ in der Prozessorsteuerung sowie das Aktivieren und Inaktivieren von RY in der Speichersteuerung um jeweils einen Zustand „nach unten“ verschoben werden. Das heißt, daß in der Prozessorsteuerung die Ready-Abfrage unmittelbar vor der Datenübernahme erfolgt und ein weiterer Zustand eingeführt werden muß, siehe Bild 4-9. Diese Maßnahmen allein führen aber noch nicht auf das Minimum von 2 Takten für einen Buszyklus (gemäß den kürzest möglichen Flankenwechseln der Handshake-Signale). Dazu muß der Prozessor gleichzeitig mit der Datenübernahme des laufenden Buszyklus die Adresse des nächsten Buszyklus ausgeben bzw. – was auf dasselbe hinausläuft – gleichzeitig mit der Ausgabe der „neuen“ Adresse die Übernahme des „alten“ Datums abschließen; das wird

synchroner
Buszyklus
bei gleichen
Taktflanken

durch den Zusatz „und nicht ready“ im Markenkonflikt K im Prozessorgraphen gewährleistet, siehe Bild 4-9.

Der Prozessor arbeitet somit in den Fällen, in denen er diese Adresse noch nicht zu diesen Zeitpunkten zur Verfügung stellt, mit 3 Takten/Buszyklus (1 Marke im linken Graphen) und in den Fällen, in denen er diese Adresse bereits zu diesen Zeitpunkten vorrätig hat, mit 2 Takten/Buszyklus (2 Marken im linken Graphen). Teilbild b zeigt im Detail die Ausbildung der Zeitverzögerung in der Speichersteuerung durch einen oder mehrere zusätzliche Wartezustände; derselbe Effekt kann alternativ durch einen Zähler erreicht werden, der auf einen Wert $n < 0$ vor-eingestellt wird und mit jedem Taktimpuls um 1 heruntergezählt wird, bis er den Zählerstand 0 erreicht hat.

wait states
beim syn-
chronen
Buszyklus

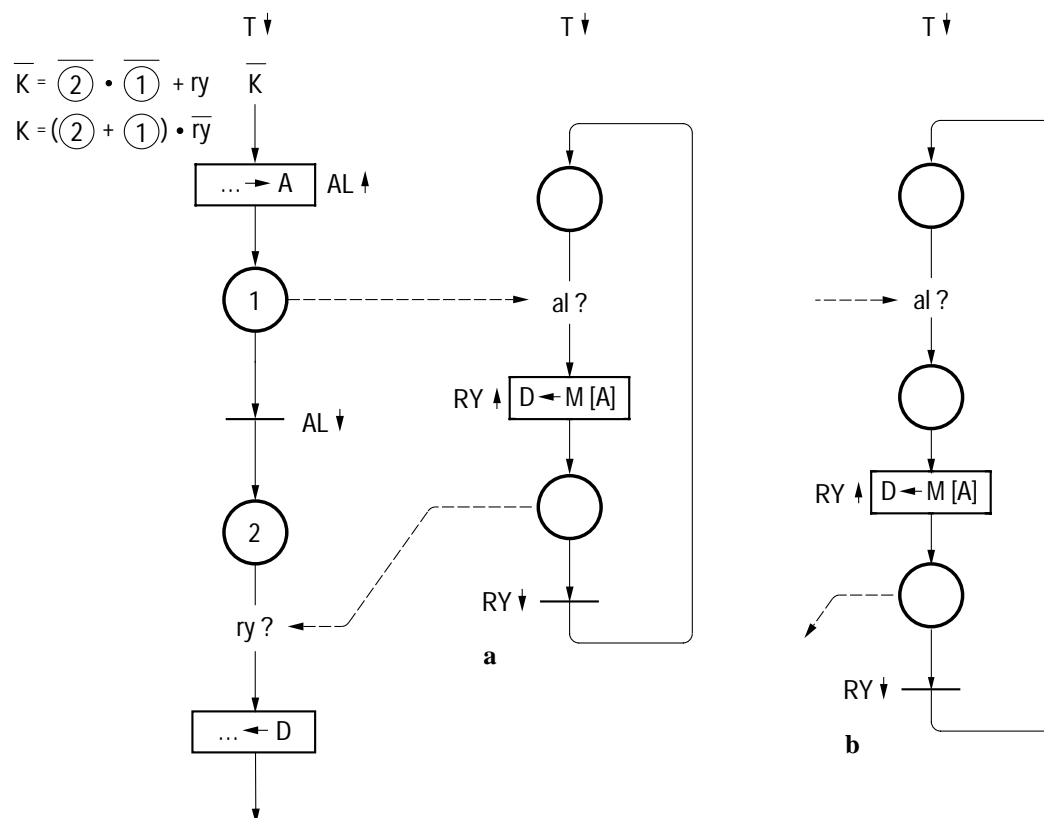


Bild 4-9. Synchroner Buszyklus für das Lesen bei gemeinsam getakteter Speichersteuerung (T Bustakt); **a** ohne Wartezustand (2 bzw. 3 Takte), **b** mit einem oder mehr Wartezuständen (3 bzw. 4 oder mehr Takte).

Aufgabe 4.1. Zeichnen Sie zwei Signaldiagramme für die in Bild 4-8b und Bild 4-9b dargestellten Lesezyklen mit jeweils zwei aufeinanderfolgenden Lesevorgängen. Gehen Sie dabei davon aus, daß die Adressen schnellstmöglich vom Prozessor bereitgestellt werden.

Aufgabe 4.2. Konstruieren Sie die zu Bild 4-8 und Bild 4-9 gehörenden asynchronen bzw. synchronen Schreibzyklen und zeichnen Sie für sie Signaldiagramme.

Aufgabe 4.3. In Bild 4-7 ist der allgemeine Fall eines (asynchronen) Buszyklus mit für beide Systemkomponenten eigenen, unterschiedlichen Takten T_{Master} und T_{Slave} dargestellt. In diesem Graphennetz ist natürlich der Fall eines synchronen Buszyklus mit für beide Systemkomponenten identischem Takt $T_{\text{Master}} = T_{\text{Slave}} = T$ enthalten. Ermitteln Sie die minimale Anzahl an Takten für einen Buszyklus, und zwar (a) bei Taktsteuerung mit gleichen und (b) bei Taktsteuerung mit wechselnden Taktflanken. Welcher der Buszyklen ist schneller? Und warum?



<http://www.springer.com/978-3-540-00027-3>

Rechnerorganisation

Die Prinzipien

Liebig, H.

2003, X, 494 S., Softcover

ISBN: 978-3-540-00027-3