Heft 04
Sequentielle Digitalschaltungen

Stand: 1.02

- Vorläufiges Exemplar. Nur zur Information -

1. Sequentielle Elementarschaltungen

1.1. Grundbegriffe

Kombinatorische Schaltungen liefern ihre Ergebnisse unmittelbar nachdem die Operanden-Werte an den Eingängen anliegen. Gültige Ergebnis-Belegungen sind nach einer Durchlaufzeit zu erwarten, die sich näherungsweise als Produkt von Gatterlaufzeit und Schaltungstiefe ergibt. Solche Schaltungen haben keine "Vorgeschichte"; jede Operanden-Kombination ergibt jeweils immer wieder dasselbe Ergebnis. Kombinatorische Zuordnungen lassen sich aber nur für beschränkte Operandenlängen und elementare Operationen technisch verwirklichen. Sind die beteiligten Informationsstrukturen zu lang bzw. die Informationswandlungen zu kompliziert, so ist die Ausführung in mehreren aufeinanderfolgenden Zeitschritten zu organisieren. Dafür sind elementare Speichermittel notwendig, die direkt mit kombinatorischen Netzwerken zusammengeschaltet sind. Derartige Einrichtungen wollen wir nachfolgend behandeln. Wir setzen dabei voraus, daß die elementaren Speicherfunktionen durch geeignete Zusammenschaltung *logischer* Funktionseinheiten (z. B. von Gattern) verwirklicht werden und nicht durch Ausnutzung *physikalischer* Effekte (Magnetismus, Ladungsspeicherung usw.).

Erklärung:
Aus Aufwands- bzw. Kostengründen hat man, von Beginn der Entwicklung an bis in die 70er Jahre hinein, kombinatorische Verknüpfungen und Speicherfunktionen zumeist auf unterschiedliche Weise realisiert. Um dies zu verstehen, müssen wir uns nur vorstellen, wir müßten eine elementare, auf "logischer" Grundlage beruhende Speicherzelle (ein Latch bzw. Flipflop) mit einzelnen Transistoren oder gar mit Elektronenröhren aufbauen. Verständlich, daß man sich alle möglichen Tricks hat einfallen lassen, um den Aufwand zu vermindern. So gab es Speicherelemente auf Grundlage von Ferritkernen, Umlaufspeicher mit Ultraschall-Verzögerungsleitungen, Ladungsspeicher mit Kondensatoren usw. Erst durch die Technologien der integrierten Schaltkreise sind die aus Gattern aufgebauten Speicherelemente zur Selbstverständlichkeit geworden. Allerdings finden wir auch in modernen Systemen sparsamere Lösungen. Insbesondere wird das Prinzip der Ladungsspeicherung ausgiebig genutzt. Das betrifft im besonderen die DRAMs, aber auch manche Prozessoren haben "dynamisch" wirkende Speichermittel.

Zustand (Maschinen- bzw. Automatenzustand)
Eine Schaltung befindet sich zu einer beliebigen Zeit t in einem einzigen bestimmten Zustand z_i, der durch die jeweilige Belegung der Speichermittel bestimmt ist. Die Schaltung verharrt solange in diesem Zustand, bis ein *Zustandswechsel* $z_i \rightarrow z_j$ bzw. $z(t) \rightarrow z(t+1)$ veranlaßt wird. Auslösende Ursachen sind entweder (1) externe *Ereignisse* oder (2) interne *Zeitvorgaben* (Takte).

Maschinenzyklus
Das Zeitintervall für den Übergang von einem Zustand in den anderen bezeichnen wir als Maschinenzyklus.

Speicherung
Speichern ist das Halten eines Zustandes über die Zeit.

Takt
Takte sind zeitbestimmende Signale. Zustandswechsel werden durch Takte ausgelöst. In den meisten Fällen ist ein Takt eine kontinuierliche Folge von Impulsen mit konstanter Folgefrequenz (es gibt aber auch Einzeltakte und Takte mit veränderlicher Folgefrequenz).

Synchroner Takt
Ein synchroner Takt gibt einen festen Zeitrahmen für die Zustandsübergänge im System vor; die zeitbestimmende Impulsfolge hat eine feste und konstante Impulsfrequenz (Taktfrequenz). Abweichungen sind nicht beabsichtigt, sondern lediglich Folge von Parameter-Toleranzen (Frequenzkonstanz, Verzögerungen usw.).

Asynchroner Takt
Beim asynchronen Takt ist der Zeitrahmen variabel; die Taktimpulse werden in Abhängigkeit vom jeweiligen Verarbeitungsablauf gebildet.

Achtung: Man kann Schaltwerke auch so auslegen, daß sie ohne jeglichen Takt arbeiten. Das gelingt aber nicht immer, und oft (außer in ziemlich einfachen Fällen) braucht man irgendeine Form der Taktierung. In der Literatur bezeichnet man zumeist alle Arten von Schaltwerken, die nicht von einem zentral gebildeten Takt gesteuert werden, als asynchron. "Asynchron" ist also nicht immer mit "völlig ungetaktet" gleichzusetzen!

Statische Arbeitsweise
Statisch arbeitende sequentielle Schaltungen sind bei beliebig langer Dauer des einzelnen Maschinenzyklus noch arbeitsfähig; man kann also den Takt "anhalten". Statisch arbeitende Schaltungen (z. B. Prozessoren) sind daran zu erkennen, daß im Datenblatt keine Mindest-Taktfrequenz angegeben ist.

Dynamische Arbeitsweise
Dynamisch arbeitende Schaltungen sind unterhalb einer gewissen Mindest-Taktfrequenz (im Datenblatt angegeben) nicht mehr funktionsfähig. Dies ist lediglich durch die Technologie bedingt: man hat, zwecks Aufwandsverringerung, für bestimmte Funktionen dynamische (zeitabhängige) Effekte, z. B. die Ladungsspeicherung, ausgenutzt. Für die Funktionsweise an sich ist dies bedeutungslos; es ist allerdings nicht möglich, zu Fehlersuchzwecken den Takt einfach "anzuhalten" und trotzdem den jeweiligen Zustand zu bewahren.

Diskrete Maschinenzeit
Für die grundsätzliche Funktionsweise (natürlich nicht für das Leistungsvermögen) ist die absolute Dauer des Maschinenzyklus bedeutungslos. Um die Funktionsweise einer Schaltung zu erklären, kann man also von kontinuierlichen Zeitangaben absehen. Vielmehr spricht man von *diskreten* Zeitpunkten, an denen die Zustandsübergänge stattfinden. Diese werden üblicherweise mit t (aktueller Zeitpunkt), t+1 (nächster Zeitpunkt), t-1 (vorhergehender Zeitpunkt) usw. bezeichnet.

1.2. Latches und Flipflops

1.2.1. Speicherung durch gesteuerte Selbsthaltung

Wie können wir mit "rein logischen" Mitteln, ohne direkte Ausnutzung physikalischer Effekte, Information speichern? Wir gehen systematisch vor und betrachten zunächst das einzelne Bit. Wenn wir den Ausgang eines ODER-Gatters auf einen seiner Eingänge zurückführen (Abbildung 1.1a), brauchen wir am anderen Eingang nur einmal eine Eins anzulegen. Daraufhin wird das Gatter die Ausgangsbelegung "Eins" über die Rückführung selbst halten.

Abbildung 1.1 Elementare Speicherschaltungen nach dem Prinzip der Selbsthaltung

Wir haben aber keine Möglichkeit, wieder eine Null zu speichern. Damit ist die Schaltung praktisch kaum brauchbar. Wie müssen wir sie verbessern? - Offensichtlich bekommen wir den Ausgang des ODER-Gatters wieder auf Null, wenn wir die Rückführung auftrennen. Das gelingt mit einem UND-Gatter (Abbildung 1.1b). Eine solche Schaltung bezeichnet man als D-Latch. Der Ausdruck (Latch = Klinke; sprich: Lätch) bringt das Selbsthalteprinzip bildhaft zum Ausdruck.

Wirkungsweise: Der Speichererlaubniseingang ist als aktiv High definiert, der Löscheingang als aktiv Low. Wollen wir ein Datenbit speichern, muß das Speichersignal auf High gesetzt werden. Um die einmal gespeicherte Information zu halten, muß "Speichern" mit Low und "Löschen" mit High belegt sein (es schaltet somit die Rückführung durch). Vor jedem erneuten Speichern ist die Rückführung zu trennen, indem "Löschen" auf Low gelegt wird.

Ähnliche Selbsthalteschaltungen lassen sich auch mit zwei gleichartigen Gattern (NAND oder NOR) aufbauen (Abbildung 1.2). Solche Schaltungen bezeichnet man als RS-Latch. "RS" steht für "Rücksetzen" (Reset) und "Setzen" (Set).

Abbildung 1.2 RS-Latch mit NAND- bzw. NOR-Gattern

Wollen wir eine Eins speichern, so müssen wir den S-Eingang aktivieren (Setzen), wollen wir eine Null speichern, den R-Eingang (*Rücksetzen*). Die Eingänge R und S sind hier jeweils gleichsinnig aktiv (NANDs: Low; NORs: High). Zudem können wir ausgangsseitig den gespeicherten Zustand sowohl in "wahrer" als auch in invertierter Form abnehmen.

Die Tatsache der Selbsthaltung wird meist durch eine besondere zeichnerische Darstellung mit überkreuzten Verbindungen kenntlich gemacht

Problem: der "verbotene" Zustand
Was geschieht, wenn R und S beide aktiv sind? Diese Belegung wird üblicherweise als "verboten" bezeichnet. So haben wir es auch in die Belegungstabelle der Abbildung 1.2 eingetragen. Tatsächlich kann man diese Belegung aber anlegen. Sie führt dazu, daß *beide* Ausgänge gleiche Pegel annehmen (NANDs: High; NORs: Low). Das wird in manchen Schaltungen auch ausgenutzt*). Was wirklich "verboten" ist, ist das gleichzeitige *Ab*schalten (Deaktivieren) von R und S: dann nämlich ist der Zustand der Schaltung nicht mehr definiert; es ergibt sich eine zufällige Belegung.

*): beispielsweise in einfachen Vermittlungsschaltungen (Arbitration Latches).

1.2.2. Erweiterungen des RS-Latches

Mehrere Setz- und Rücksetzeingänge
Die Gatter, die das RS-Latch bilden, können auf mehr als zwei Eingänge erweitert werden (Abbildung 1.3). Diese zusätzlichen R- und S-Eingänge wirken dann jeweils disjunktiv (als ODER-Verknüpfung).

Abbildung 1.3 RS-Latches mit mehreren Setz- und Rücksetzeingängen

Gesamt- oder Systemrücksetzen
Oft ist es notwendig, Latches oder Flipflops in einen Grundzustand zu versetzen. Dazu ist ein entsprechend (z. B. nach dem Einschalten) zu erregendes Rücksetzsignal vorzusehen. Dies kann in allen betreffenden Latches über jeweils einen zusätzlichen Eingang wirksam werden.

Herkömmliche Steuerflipflops
Nach einem seit langem bewährten Entwurfsprinzip werden die Maschinenzustände durch Flipflops (bzw. Latches) repräsentiert, die durch *Einschaltbedingungen* gesetzt und durch *Ausschaltbedingungen* zurückgesetzt werden. Diese Bedingungen sind jeweils kombinatorische Verknüpfungen der betreffenden Signale. (Es ist die - zumindest vom Prinzip her einfache - "naive" Umsetzung der Funktionsbeschreibung in eine Schaltung. Steht dort beispielsweise: "eine Datenübertragung ist einzuleiten, wenn...; sie ist zu beenden, wenn...", dann bildet der Hardware-Entwerfer die Beschaltung der Ein- und Ausschaltknoten unmittelbar gemäß den beschreibenden Angaben, die auf die "wenn's" folgen.)

Kombinatorische Verknüpfungen kann man in disjunktiver Normalform darstellen und mit UND-ODER-Netzwerken aufbauen. Dabei ist die ODER-Ebene bereits durch die Gatter gegeben, die das Latch selbst bilden (Abbildung 1.4).

Abbildung 1.4 Typisches Steuerflipflop

Auf diese Weise kann man mit elementaren Gattern recht komplizierte Steuerschaltungen aufbauen. Heutzutage sind solche Schaltungen zumeist in programmierbarer Logik (EPLDs, GALs usw.) verschwunden.

Taktsteuerung von Latches
Soll das Schalten eines Latches nur bei Anliegen eines besonderen Signals (Taktsignals) erlaubt sein, so ist dies mit den Eingängen R und S jeweils konjunktiv zu verknüpfen (Abbildung 1-5; dort ist auch dargestellt, wie wahlweise zusätzliche, unabhängig vom Takt wirkende Stelleingänge PRESET und CLEAR eingefügt werden können).

Abbildung 1.5 Taktgesteuertes (transparentes) RS-Latch

1.2.3. Latches als Datenspeicher (D-Latches)

Wenn wir ein Datenbit speichern wollen, müssen wir das Latch entsprechend umbauen bzw. erweitern. Wir brauchen einen Dateneingang (D) sowie eine Möglichkeit, die Übernahme des anliegenden Datenwertes zu erlauben bzw. zu sperren. Das Latch darf nur bei entsprechender Erlaubnis umschalten. Es ist bei anliegender Erlaubnis zu setzen, wenn eine Eins, und zu löschen, wenn eine Null zu speichern ist. Abbildung 1.6 zeigt verschiedene Ausführungsformen von D-Latches.

Abbildung 1.6 D-Latches

Latch-Register
Register sind Anordnungen aus mehreren gleichartigen Flipflops oder Latches, die mit gemeinsamen Takt- und Steuersignalen gesteuert werden. Abbildung 1.7 zeigt ein typisches Latch-Register.

Abbildung 1.7 8-Bit-Latch-Register '373 (Texas Instruments)

Schnelle Halteregister
Gelegentlich ist es notwendig, Latch-Register mit geringster Verzögerungszeit für die Daten vorzusehen, beispielsweise als Adressen-Halteregister an Speichern (da der Speicherzugriff erst dann beginnen kann, wenn die Adresse an den Speicherschaltkreisen anliegt, ist es von leistungsbestimmender Bedeutung, die Adresse auf dem Weg vom Prozessor zum Speicherschaltkreis nicht übermäßig zu verzögern). "Fertige" Latch-Register sind hier nicht immer schnell genug. Ein Ausweg besteht darin, Multiplexer-Schaltkreise einzusetzen (Abbildung 1.8; vergleichen Sie damit die Daten-Latches c), d) in Abbildung 1.6): die UND-ODER-Strukturen sind nichts anderes als die Auswahlschaltungen, wie wir sie in jedem 2-zu-1-Multiplexer vorfinden).

Abbildung 1.8 Schnelles Halteregister mit Multiplexer-Schaltkreis

Ein D-Latch aus Gattern hat zwischen D und Q die Schaltungstiefe 2. Betrachtet man fertige (off the shelf) Schaltkreise, ist die Multiplexer-Bauweise schneller (Tabelle 1.1).

Baureihe	**AS**	**ACT**
ein Gatter (zum Vergleich)	1...5 ns	1,5...7,4 ns
fertiges Register '373	3,5...6 ns	1,5...13 ns (typischerweise 8,5 ns)
MUX '157	1...6 ns	1,5...7,5 ns (typischerweise 4,5 ns)
MUX '158 (invertierend)	1...4,5 ns	1,5...8 ns (typischerweise 4,5 ns)

Tabelle 1.1 Geschwindigkeitsvergleich von D-Latches

Hinweise:

1. Multiplexer mit invertiertem Ausgang haben (in TTL-Technologie) die geringste Verzögerungszeit. Für einen Speicher ist es gleichgültig, ob Adressen "wahr" oder invertiert angeboten werden. Eine solche Ausführung erfordert aber zusätzliche Negatoren in der Rückführung.

2. Die Rückführung muß den Dateneingang verwenden, der bezüglich des Auswahlsignals die geringste Verzögerung hat (das ist bei den Typen '157; '158 der A-Eingang). Ansonsten würde die Information nach dem Umschalten von "Speichern" auf "Selbsthaltung" verlorengehen.

3. Altmodisch? - Keineswegs. Nur wird man heutzutage kaum noch mit einzelnen Multiplexer-Schaltkreisen entwerfen. Aber die Latches und Flipflops in hochintegrierten Schaltkreisen sind oft aus rückgekoppelten Multiplexer-Strukturen aufgebaut, und die elementaren Funktionsblöcke - die Makrozellen - moderner programmierbarer Schaltkreise (EPLDs, FPGAs) enthalten solche Strukturen, um damit Latches und Flipflops verwirklichen zu können.

1.2.4. Flipflops

Ein Flipflop ist ein taktgesteuerter 1-Bit-Speicher, der seine Ausgangsbelegung nur in Bezug auf eine jeweils spezifizierte Takt*flanke* ändert (d. h. auf die ansteigende bzw. abfallende Flanke des Taktimpulses). Oft haben Flipflops zusätzliche, unabhängig vom Takt wirkende Lösch- bzw. Einstelleingänge (Clear bzw. Preset). Flipflops werden heutzutage nicht mehr aus Gattern zusammengeschaltet, sondern praktisch ausschließlich in Form von "off the shelf"-Schaltkreisen oder innerhalb von programmierbarer Logik oder von LSI-Schaltkreisen eingesetzt.

Latch und Flipflop

Das taktgesteuerte Latch unterscheidet sich vom Flipflop dadurch, indem bei ersterem alle vom Takt gesteuerten Eingänge für die Dauer der Takt-Aktivierung an den Ausgängen wirksam sind. Ist in den Schaltungen von Abbildung 1.6 das Taktsignal aktiv, so folgen die Ausgänge allen Änderungen der Eingänge nach. Man sagt, das Latch ist *transparent* für die Eingangssignale (Sie finden in Datenblättern gelegentlich die Bezeichnung "transparent latches", und man bezeichnet das Taktsignal gelegentlich auch nicht als Takt [Clock], sondern als Erlaubnis- bzw. Tor-Signal [Enable bzw. Gate]). Hingegen wird beim "echten" Flipflop die Eingangsbelegung nur zur jeweils spezifizierten Takt*flanke* ausgewertet (Abbildung 1.9).

Abbildung 1.9 Schaltverhalten von "transparentem" Latch und "echtem" Flipflop im Vergleich

Latch und Flipflop in der Fachsprache

Leider wird nicht immer streng unterschieden. Manche Theoretiker bezeichnen jeden 1-Bit-Speicher als Flipflop, unabhängig davon, wie die Taktsteuerung aussieht. Hingegen finden Sie in der praxisbezogenen Literatur (auch in Datenblättern) die Bezeichnung "Latch" für jede Art von Halteregistern, auch wenn diese mit flankengesteuerten Flip-flops bestückt sind. Tip: Sehen Sie sich am Schaltsymbol und an der Innenschaltung die Takteingänge an. Aus der Symbolik ist ersichtlich, ob es sich um Flankensteuerung handelt oder nicht. Ein weiteres Indiz: Die Angabe der Vorhaltezeit (t_{SETUP}) mit Bezug auf die LO-HI-Flanke und Haltezeit (t_{HOLD}) Null deuten auf Flankensteuerung hin. Ist t_{SETUP} mit Bezug auf die HI-LO-Flanke spezifiziert und $t_{HOLD} > 0$, handelt es sich meist um ein transparentes Latch.

Master-Slave-Flipflops

Die Flankensteuerung erreicht man durch Hintereinanderschalten zweier taktgesteuerter Latches, wobei das zweite mit dem invertierten Takt beschaltet ist (Abbildung 1.10).

Abbildung 1.10 RS-Master-Slave-Flipflop

Flankengesteuerte Flipflops

Auch mit rein kombinatorischen Verknüpfungen läßt sich eine "echte" Flankensteuerung verwirklichen (Abbildung 1.11).

Abbildung 1.11 Flankengetriggertes D-Flipflop (Typ 7474; Texas Instruments)

Prinzip:

Bei aktivem Takt (C = HI) hält der jeweils beim LO-HI-Übergang von C übernommene Wert über die Rückführungen sich selbst, auch wenn sich D ändert.

Flipflop-Arten

Wieviele Flipflop-Arten gibt es? - Diese Frage läßt sich exakt beantworten: Das Verhalten eines Flipflops ist abhängig von der jeweiligen Belegung der Eingänge und von seinem aktuellen Zustand. Ein Flipflop hat als einfachstes binäres Speicherglied insgesamt nur zwei Zustände (0 oder 1). Wenn wir uns mit höchstens zwei Eingängen begnügen (wir kennen das RS-Flipflop mit zwei Steuereingängen und das D-Flipflop mit einem Dateneingang), hängt der Folgezustand also von drei Binärvariablen ab.

Mit drei Variablen kann man $2^{2^3} = 256$ verschiedene Schaltfunktionen bilden. Es gibt also insgesamt 256 verschiedene Flipflops (darunter sind natürlich auch Lötstellen, Unterbrechungen, Festwerte sowie alle kombinatorischen Schaltfunktionen, die von einer und von zwei Variablen abhängen, und andere nicht besonders brauchbare "Flipflops"). In der Praxis beschränkt man sich jedoch auf nur wenige Typen (Abbildung 1.12; hier lernen Sie auch die Beschreibungsmittel: Automatengleichung, Automatentabelle und Zustandsgraph am Beispiel kennen).

Abbildung 1.12 Wichtige Flipflop-Arten

Das *JK- Flipflop* ist ein modifiziertes RS-Flipflop, bei dem der "verbotene" Zustand (J, K beide Eins) erlaubt ist und die Wirkung hat, die Ausgangsbelegung zu ändern (aus Eins wird Null und umgekehrt; Trigger- oder "Toggle"-Funktion). Das T-Flipflop (Trigger- bzw. Toggle-Flipflop) wechselt mit jedem Takt seine Belegung. Das *DV-Flipflop* finden wir nicht als einzelnen Schaltkreis. Es ist ein D-Flipflop, erweitert um einen Erlaubniseingang V (nur bei aktivem V wird die Eingangsbelegung übernommen, ansonsten wird der bisherige Zustand gehalten gespeichert).

Wichtig ist: Alle rückkopplungsfähigen Flipflops lassen sich durch Zusatzbeschaltung in jede beliebige andere Art umwandeln. Abbildung 1.13 zeigt einige Beispiele.

Abbildung 1.13 Wechselseitige Umwandlungen von Flipflop-Arten (Beispiele)

Hinweis:
Die Anordnung, die die DV-Funktion mit einem D-Flipflop verwirklicht, ist die Grundschaltung für die Nutzung von D-Flipflops in Steuerschaltungen und "vollsynchronen" Registern.

In modernen Schaltungen werden flankengesteuerte Flipflops bevorzugt, insbesondere D-Flipflops. Dessen besonderer Vorteil: Die Schaltgleichungen, die das gewünschte zeitliche Verhalten beschreiben (Automatengleichungen), lassen sich direkt in Ansteuernetzwerke umsetzen.

Das JK-Master-Slave-Flipflop
Das "klassische" JK-Flipflop ist ein aus dem RS-Typ abgewandeltes Master-Slave-Flipflop, wobei die "über Kreuz" auf die Eingänge zurückgeführten Ausgänge das gewünschte Verhalten bei J = K = 1 gewährleisten (Abbildung 1.14).

Abbildung 1.14 JK-Master-Slave-Flipflop

Abbildung 1.14 weist auf eine charakteristische Störungsempfindlichkeit solcher Master-Slave-Flipflops hin: bei aktivem Takt wird jeder HI-Störimpuls im Master gefangen und so mit dem Abschalten des Taktes in den Slave übernommen. In den Datenblättern ist deshalb die Übernahme der Eingangsbelegung mit der Takt-Vorderflanke (LO-HI) spezifiziert, während die Ausgänge mit der Rückflanke (HI-LO) schalten. Während der Taktimpuls "HI" ist, darf sich an den Eingängen nichts ändern!

Data Lockout
Das Störproblem läßt sich vermeiden, wenn man den Master nur bis kurz nach der Vorderflanke des Taktes offenhält. Die Information erscheint aber trotzdem erst nach der Rückflanke an den Ausgängen. Es gibt Master-Slave-Flipflops mit und ohne Data Lockout.

Flankengesteuerte Flipflops
Moderne Flipflops, die man als einzelne Schaltkreise kaufen kann, sind praktisch ausnahmslos "richtig" flankengesteuert (Edge Triggered). Es gibt also wirklich nur eine schaltende Signalflanke; bei D-Typen LO-HI (Positive Edge), bei JK-Typen HI-LO (Negative Edge).

Flankengesteuerte Flipflops in Master-Slave-Schaltung
Wir bauen die Master-Slave-Anordnung als D-Flipflop auf (Abbildung 1.15) und realisieren ein ggf. erforderliches anderes Schaltverhalten durch entsprechende Zusatzbeschaltung (vgl. Abbildung 1.13). Der Grundgedanke: liegt der Takteingang auf High, so folgt das Master-Latch zwangsläufig dem D-Eingang; somit können Störimpulse auch nicht im Master gefangen werden (ein solcher Impuls würde das Master-Latch kurzzeitig setzten und dann wieder zurücksetzen). Anwendungsbeispiele: (1) manche Flipflopschaltkreise in moderneren Baureihen (Abbildung 1.16), (2) Realisierung von Flipflops in den Zellen mancher FPGA-Familien.

Abbildung 1.15 Flankengesteuertes D-Flipflop in Master-Slave-Schaltung

Abbildung 1.16 Ein moderner Flipflop-Schaltkreis des Grundtyps ‘74 (Texas Instruments)

Rückkopplungsfähigkeit
Im Gegensatz zu Latches sind Master-Slave- bzw. flankengesteuerte Flipflops *rückkopplungsfähig*, das heißt, die Ausgänge dürfen direkt oder über kombinatorische Schaltungen auf die der Taktsteuerung unterliegenden Eingänge zurückgeführt werden. (Eingänge, die der Taktsteuerung unterliegen, werden als *synchrone* Eingänge bezeichnet.) "Setzen" und "Rücksetzen" (Preset/Clear) sind meist *keine* synchronen Eingänge!

Achtung: Unbedenklich rückkoppeln dürfen Sie, wenn im Datenblatt eine Haltezeit (t_{HOLD}) Null spezifiziert ist. Ist eine Mindest-Haltezeit gefordert, so muß diese bei Rückführungen über die Verzögerungszeit des Rückkopplungsweges gewährleistet werden. "Billig"-Schaltungen sind hier nicht immer solide; man trifft keine besonderen Maßnahmen, sondern verläßt sich auf die übliche (typische) Durchschaltverzögerung (was gelegentlich schiefgehen kann...).

Flankensteilheiten
Manche flankengesteuerte Flipflops erfordern Mindest- Flankensteilheiten, Master-Slave-Flipflops und z. B. der flankengesteuerte Typ '74 nicht (die Aussage gilt nicht für alle Baureihen). Datenblatt genau beachten! Die Flankensteilheiten müssen aber auf jeden Fall den spezifizierten Grenzwerten der Baureihe entsprechen.

Verbesserung der Flankensteilheit:

- ein nicht-negierender Treiber oder 2 Negatoren in Reihe. Solche Reihenschaltungen dienen oft (1) der Verbesserung der Flankensteilheit und (2) der Signalverzögerung.
- Schmitt-Trigger (für beliebig "lahme" Eingangsflanken).

Lange Leitungen
An Ausgänge von Flipflops und Latches, die (1) selbst noch Rückführungen treiben oder (2) keine schaltkreisinternen Treiberstufen haben, sollten keine langen Leitungen (Praxiswert > 25...60 cm) direkt angeschlossen werden. In solchen Fällen sollten die Signale vor Verlassen der Leiterplatte entkoppelt sein (über Treiber bzw. Negatoren). Es ist nicht selten, daß man auf Leitungslängen-Beschränkungen nicht viel Rücksicht nimmt. Solche Nachlässigkeiten sind aber eine häufige Ursache von Fehlern, namentlich von sporadischen Funktionsstörungen. Bei Takt- und Rücksetzsignalen sollte darauf geachtet werden, daß sie frei von Störungen sind (Messen!).

Rücksetzen
Ein generelles Rücksetzen zumindest aller Steuerflipflops der Schaltung ist unbedingt sinnvoll. Beim Fehlersuchen hat es sich bewährt, zunächst das Rücksetzen statisch zu aktivieren und die Anfangsbelegungen zu überprüfen. Die allgemeine Rücksetzleitung sollte von Störungen freigehalten werden, was wegen der Leitungslänge gelegentlich Sondervorkehrungen erfordert. (Beispiel: Die Wirkung ist aktiv-LO definiert, und die Leitung wird über einen niederohmigen Widerstand auf den Pegel der Versorgungsspannung "hochgezogen". Somit können selbst außergewöhnliche Störungen keinen Schaden anrichten.)

Vorhalte- und Haltezeiten (Setup/Hold Timing)
Die Eingänge müssen - bezogen auf die jeweilige Taktflanke - mindestens um die Setup-Zeit voreilend stabil anliegen. Die Belegung muß mindestens für die Dauer der Hold-Zeit nach der jeweiligen Flanke stabil gehalten werden. Datenblätter genau beachten! Flipflops mit Haltezeit 0 erlauben die Rückführung von Ausgängen auf die (synchronen) Eingänge (J/K bzw. D).

1.2.5. Metastabilität (Synchronisation, Eintaktierung)

Was geschieht, wenn sich Eingangsbelegungen eines Flipflops im Setup-Hold-Intervall einer schaltenden Taktflanke ändern? - Das hängt ganz vom Zufall ab. Manchmal wird nichts Besonderes geschehen; das Flipflop wird entweder seinen Zustand ändern oder den bisherigen beibehalten. Es gibt aber innerhalb des Setup-Hold-Intervalls ein "kritisches Zeitfenster" (10...150 ps; abhängig von der jeweiligen Schaltkreis-Baureihe). Fällt die Eingangs-Änderung in dieses Zeitfenster, gelangt das Flipflop in einen instabilen (*metastabilen*) Zustand: seine Ausgänge verhalten sich undefiniert; es kommt für eine gewisse Zeit zu "wilden" Schwingungen (Abbildung 1.17).

Abbildung 1.17 Das metastabile Verhalten eines Flipflops

Diese Tatsache hat eine große praktische Bedeutung. Trotz aller Mühe läßt sich nämlich nie vollkommene Synchronität gewährleisten. Auch wenn die eigentliche Schaltung vollsynchron arbeitet, sind die Schnittstellen zur Außenwelt naturgemäß immer asynchron (Schalter, Tasten, Sensoren, Interfaces anderer Funktionseinheiten usw. "wissen" nun einmal nichts vom internen Takt unserer Hardware). An diesen Schnittstellen ist also stets das Problem der *Synchronisation* (Eintaktierung) zu lösen. Die naheliegende Lösung: das asynchrone Signal auf den Dateneingang eines D-Flipflops legen, der mit dem jeweils passenden Takt angesteuert wird. Und genau hier ist mit metastabilen Zuständen zu rechnen! Überlegen wir einmal, wie oft ein solcher Zustand eintreten wird. Das Flipflop werde mit der Taktfrequenz f_{CLK} betrieben. Das einzutaktierende Signal ändere sich mit der Frequenz f_{IN} (grundsätzlich muß gelten $f_{IN} < f_{CLK}$; sonst würde die Eintaktierung gar nicht funktionieren). Die Dauer des kritischen Zeitfensters sei t_W. Nun muß, um einen metastabilen Zustand hervorzubringen, eine Datenflanke im kritischen Zeitfenster der Taktflanke auftreten. Das mittlere Zeitintervall zwischen zwei solchen Ereignissen (die MTBF; Mean Time between Failures) ergibt sich - wir verzichten auf die Herleitung - zu:

$$\mathrm{MTBF} = \frac{1}{f_{IN} \cdot f_{CLK} \cdot t_W}$$

Im Beispiel: $f_{CLK} = 2$ MHz, $f_{IN} = 10$ kHz, $t_W = 50$ ps. Damit ergibt sich:

$$\mathrm{MTBF} = \frac{1}{2\ \mathrm{MHz} \cdot 10\ \mathrm{kHz} \cdot 50\ \mathrm{ps}}$$

Es ist also etwa alle Sekunde mit einer Fehlfunktion zu rechnen.

Wie läßt sich Abhilfe schaffen?
Grundsätzlich: Es gibt kein 100%ig wirkendes Gegenmittel. Man gibt sich vielmehr zufrieden, wenn man - rechnerisch und meßtechnisch nachgewiesen - einen so hohen MTBF-Wert erhält, daß Fehlfunktionen und Ausfälle aus anderen Ursachen viel wahrscheinlicher sind (so ist eine MTBF von 10^4 Jahren ein solcher ausreichender Wert). Dabei kann es nur darum gehen, den metastabilen Zustand selbst oder dessen Auswirkungen zu verhindern, nicht aber, ein definiertes Schaltverhalten des Flipflops zu erzwingen (wenn Takt- und Datenflanke zusammentreffen, ist es nach wie vor Glückssache, ob das Flipflop schaltet oder nicht). Die Gegenmaßnahmen im einzelnen:

1. Auswahl von Flipflops, die von Natur aus nur selten in metastabile Zustände übergehen. Wichtig hierfür ist ein möglichst kleines kritisches Zeitfenster t_W. Flankengesteuerte Flipflops schneller Baureihen haben sich hier als durchaus geeignet erwiesen (so sind D-Flipflops 74AS74, 74F74, 74AC74 gut geeignet, ebenso D-Flipflop-Register und Schieberegister).
2. Ausgänge solcher Synchronisations- bzw. Eintaktierungs-Flipflops dürfen nach der schaltenden Taktflanke erst dann ausgewertet werden, wenn mögliche metastabile Zustände abgeklungen sind. Je länger die Wartezeit, um so höher die Funktionssicherheit (bzw. MTBF).
3. durch Hintereinanderschalten von zwei Flipflops wird die Wartezeit gleichsam automatisch gewährleistet (Doppelsynchronisation; Abbildung 1.18). Das Signal wird dadurch um eine weitere Taktperiode verzögert. Solche Anordnungen sind als fertige Schaltkreise erhältlich (Metastable Resistant Flipflops; Beispiele: SN74AS3074, AS3374, AS 4374, 74AC(T)11478).

Abbildung 1.18 Doppelsynchronisation (Texas Instruments)

Abbildung 1.19 veranschaulicht das metastabile Verhalten verbreiteter Logikfamilien. In der Waagerechten ist das Zeitintervall t_x aufgetragen, um das die Auswertezeit verlängert wird (in der Schaltung nach Abbildung 1.18 entspricht t_x näherungsweise der Periodendauer des Systemtaktes). Die Senkrechte enthält die verbleibende Unsicherheit (ausgedrückt als MTBF).

Abbildung 1.19 Das metastabile Verhalten verbreiteter Logikfamilien (Texas Instruments)

Wir merken uns:

1. wenn in Datenblättern, z. B. von Mikroprozessoren, Eingänge als synchron spezifiziert sind, dann muß dafür gesorgt sein, daß die Signale die Setup- und Haltezeitforderungen erfüllen, andernfalls funktioniert es nicht (dies äußert sich zumeist als sporadische Fehlfunktion). In einer gut entworfenen und funktionsfähigen Schaltung ist kaum mit Fehlern infolge metastabiler Zustände zu rechnen (die MTBF beträgt etliche Jahre). Ohne entsprechende Maßnahmen kann die MTBF hingegen im Bereich von Minuten bis Sekunden liegen. Bei wiederholten aufeinanderfolgenden Funktionsstörungen ist also die Synchronisations-Hardware entsprechend mit zu verdächtigen.

2. sind Eingänge als asynchron spezifiziert, so wird bei Nichteinhaltung der Anforderungen die Funktion an sich nicht beeinträchtigt, das jeweilige Signal kann aber möglicherweise erst in späteren Taktperioden wirksam werden (Beispiel: Annahme von Interrupts). Erklärung: Einem "asynchronen" Eingang sind im Schaltkreis entsprechende Synchronisierschaltungen nachgeordnet.
3. ein Signal, das synchronisiert werden soll, muß wenigstens so lange aktiv sein, daß es von einem Taktimpuls garantiert wenigstens einmal erfaßt wird. Faustregel: Aktivierung > 1 Taktperiode; Taktfrequenz wenigstens das Doppelte der Impulsfolgefrequenz des Eingangssignals ($f_{CLK} \leq 2\ f_{IN}$). Kurze Eingangsimpulse müssen passend verlängert werden (einige Schaltungen dazu zeigen wir im nächsten Abschnitt).
4. Latches, gleich welcher Art, eignen sich überhaupt nicht zu Synchronisations--zwecken.
5. auch die üblicherweise als "asynchron" bezeichneten Setz- und Rücksetzein-gänge von Flipflops (Preset/Clear) haben kritische Zeitfenster in Bezug auf den Takt. Trickschaltungen (wir werden im Folgenden einige kennenlernen), die diese Eingänge für funktionelle Zwecke ausnutzen und mit asynchronen Signalen belegen, sind deshalb nicht völlig unbedenklich (sie funktionieren aber üblicherweise).

1.3. Elementarschaltungen mit Latches und Flipflops

Natürlich kann man Latches und Flipflops sozusagen "wild" verschalten, um bestimmte Funktionen zu verwirklichen. Es gibt aber - für immer wieder benötigte elementare Funktionen - bewährte Schaltungsprinzipien, die immer wieder eingesetzt werden.

1.3.1. Fangschaltungen

Die Aufgabe einer Fangschaltung: einen zu beliebiger Zeit auftretenden (kurzen) Impuls zu registrieren und solange aktiv zu bleiben, bis die Erregung weiterverarbeitet worden ist (dies wird über ein Rückstellsignal angezeigt).

Solche Schaltungen sind beispielsweise wichtig, um ankommende Impulse unabhängig von einem Takt bis zur Verarbeitung (auch: bis zur programmseitigen Abfrage) "aufzufangen" und zwischenzuspeichern.

Zum "Fangen" von Impulsen eignen sich sowohl RS-Latches als auch flankengesteuerte Flipflops (Abbildung 1.20).

Abbildung 1.20 Fangschaltungen

Der Vorteil des RS-Latches (Abbildung 1.20a): Das ankommende Signal erscheint mit nur einer Gatterlaufzeit verzögert am Ausgang.

Beim flankengesteuerten Flipflop kann man den zu fangenden Impuls auf den Takteingang führen (Abbildung 1.20b). Über den Löscheingang läßt sich das Flipflop zurücksetzen. Das Rücksetzen dominiert hier über einen währenddessen ankommenden neuen Impuls (ein solcher Impuls würde verlorengehen).

Wird der zu fangende Impuls auf einen Setzeingang geführt (Abbildung 1.20c), so dominiert dieser über ein Rückstellsignal, dessen Flanke am Takteingang wirksam wird.

Die Impulsfalle

Eine Fangschaltung ist ein nützliches Hilfsmittel zum Nachweisen von Impulsen, das auch bei nichtperiodischer Erregung wirkt und oft teure Meßmittel (Logikanalyzer) ersetzen kann (Impulsfalle, Pulse Memory). Abbildung 1.21 veranschaulicht das Prinzip, Abbildung 1.22 zeigt eine bewährte Ausführung*). Eine solche Anordnung kann man in ein Kunststoff-Kästchen einbauen, das man in die zu prüfende Hardware einfach hineinfallen läßt Das Prinzip der Nutzung: wir wollen nachweisen, ob bestimmte Signale (auch in Kombination) auftreten oder nicht. Die Signale schließen wir an die Eingänge der Impulsfalle an. Dann betätigen wir die Rücksetztaste und lösen erforderlichenfalls den jeweiligen Ablauf aus. Schaltet nur das erste Flipflop ein, sind die betreffenden Signale einmal aufgetreten (1 Impuls). Schaltet auch das zweite Flipflop ein, sind die Signale zwei- oder mehrmals aufgetreten. Nun betätigen wir die Rücksetztaste. Schalten die Flipflops nach dem Loslassen wieder ein, so handelt es sich um eine zyklische (immer wiederkehrende) Erregung.

*): die hier in einer bestimmten Hinsicht als schlechtes Beispiel dienen soll... (Zur Übung: (1) was ist hier unschön?, (2) schlagen Sie eine bessere Lösung vor.)

Abbildung 1.21 Impulsfalle (Prinzip)

Abbildung 1.22 Impulsfalle (Ausführungsbeispiel)

Praxisfragen:

1. wieviele Impulse registrieren? - Es gibt Zusätze zu Logikprüfstiften und auch Prüfstifte mit eingebauter Registrierfunktion (Pulse Memory). Oft begnügt man sich mit dem Registrieren nur eines Impulses. Das ist aber zuwenig. Andererseits kann man natürlich mehrere Flipflops hintereinanderschalten oder einen Zähler vorsehen. Dadurch wird aber das einfache Hilfsgerät wieder recht unhandlich. Die Praxis zeigt, daß die Unterscheidung "ein Impuls/mehr als ein Impuls/fortlaufende Folge" zum Fehlersuchen meistens vollkommen ausreicht. (Kommt man damit nicht aus, ist das Problem vermutlich so komplex, daß das Weiterarbeiten mit einfachen Prüfmitteln ohnehin nur Zeitverschwendung wäre.)
2. wieviele Signale, welche logischen Verknüpfungen? - Erfahrungsgemäß brauchen wir meistens die UND-Verknüpfung weniger Signale, die aber wahlweise als HI oder LO nachzuweisen sind. (Beispiel: Tritt an einem SRAM ein Schreibimpuls auf, wenn gleichzeitig der Datenausgang aktiv geschaltet ist? - Das wäre ein schwerwiegender Fehler. Um zu prüfen, ob diese Vermutung berechtigt ist oder nicht, müssen wir das Schreibsignal /WE und das Aufschalterlaubnissignal /OE konjunktiv verknüpft an die Impulsfalle anschließen, also /WE & /OE bilden.) Bis zu vier Signale, wahlweise invertierbar, sind vollkommen ausreichend. Wenn wir nicht damit auskommen: siehe Punkt 1.

1.3.2. Eintaktierungsschaltungen

Eintaktierung kurzer Impulse
Ein Eingangsimpuls, der zu kurz ist, kann von einem üblichen Synchronisations-Flipflop nicht immer erfaßt werden. Abbildung 1.23 zeigt eine Kombination aus Fang- und Synchronisationsflipflop, die Abhilfe schafft.

Abbildung 1.23 Eintaktierung kurzer Impulse (1)

Der Nachteil: liegt der Eingangsimpuls so, daß er die Taktflanke überlappt, so ist der Ausgangsimpuls 2 Takte lang.

Um diesen Nachteil zu vermeiden, kann man das Eingangssignal auf den Takteingang des Fang-Flipflops geben (vgl. Abbildung 1.20c und Abbildung 1.21). Dann muß man aber das Fang-Flipflop "nach getaner Arbeit" zurücksetzen, beispielsweise indem man den Ausgang des Eintaktierungs-Flipflops auf den Rücksetzeingang des Fang-Flipflops zurückführt. Das hat aber einen anderen Nachteil: ein während des Rücksetzens eintreffender Impuls wird nicht gefangen.

Die Schaltung gemäß Abbildung 1.24 beseitigt beide Effekte. Hier wird während des Löschens des ersten ein zweites Fang-Flipflop wirksam.

Abbildung 1.24 Eintaktierung kurzer Impulse (2)

Solche Schaltungen lassen sich auch dazu modifizieren, um Impulse zwecks programmseitiger Abfrage zu fangen.

Ausblenden kurzer Störimpulse
Mit der Schaltung nach Abbildung 1.25 wird ein Impuls nur dann als "gültig" erkannt, wenn er in mindestens einer Taktperiode stabil anliegt. Alle kürzeren Impulse führen nicht zu einem aktiven Ausgangssignal.

Abbildung 1.25 Ausblenden kurzer Störimpulse

1.3.3. Flankenerkennung

Wenn man ein Signal verzögert und das verzögerte mit dem ursprünglichen Signal kombinatorisch verknüpft, lassen sich die Signalflanken erkennen (die jeweilige Flanke führt zu einem Impuls von der Dauer der Verzögerungszeit). Man kann wahlweise mit Flipflops oder mit "analogen" Baustufen verzögern (Abbildung 1.26).

Abbildung 1.26 Flankenerkennung

1.3.4. Einzelimpulserzeugung (Single Shot)

Die Aufgabe: Auf Grund eines auslösenden Signals wird ein einzelner Impuls (Single Shot) aus dem durchlaufenden Takt ausgeblendet. Das auslösende Signal muß erst wieder inaktiv werden, ehe durch erneute Aktivierung wieder ein Ausgangsimpuls abgegeben werden kann. Abbildung 1.27 zeigt eine einfache Schaltung.

Abbildung 1.27 Einzelimpulserzeugung (Single Shot)

1.3.5. Synchrones Durchsteuern von Taktimpulsen

Die Schaltung gemäß Abbildung 1.27 liefert zwar einen taktsynchronen Ausgangsimpuls, jedoch ist dieser eine ganze Taktperiode lang (und zudem gegenüber dem eigentlichen Takt um eine Flipflop- und eine Gatter-Schaltverzögerungszeit verschoben). Hingegen ist es manchmal erforderlich, einzelne Taktimpulse mit möglichst geringer Verzögerung "auszublenden", also - abhängig von der Stellung eines Steuer-Flipflops - entweder durchzusteuern oder zu sperren. Abbildung 1.28 zeigt eine entsprechende Schaltung.

Abbildung 1.28 Synchrones Durchsteuern von Taktimpulsen

Hinweise:

1. Das Steuern von Takten über ebenfalls taktierte Flipflops ist eine wichtige Funktion in vielen Schaltungen. Das Prinzip: um den Takt sauber (ohne störende "Nadeln") zu steuern, muß das Flipflop, wie in Abbildung 1.28 gezeigt, mit der jeweils invertierten Flanke geschaltet werden.
2. Achten Sie bei gesteuerten Taktimpulsen grundsätzlich auf ein sauberes Umschalten. Takte dürfen nicht beliebig "abgeschnitten" werden (wie dies geschehen kann, wenn man den Takt einfach über ein UND-Gatter führt, dessen andere Eingänge asynchron umschalten). In gut entworfener Hardware sollte so etwas selbstverständlich sein. Allerdings kann eine defekte Synchronisation zu solchen Effekten führen (die sich dann zumeist als zeitweilige Fehler bemerkbar machen).

1.3.6. Sequenzerkennung

Die Aufgabe: Wir haben impulsförmige Signale A, B, C usw., und wir wollen eine bestimmte Reihenfolge ihres Auftretens erkennen. Die betreffende Schaltung soll aktiv werden, wenn beispielsweise eine Folge A - C - D... eintritt, nicht aber bei beliebigen anderen Folgen. Dies erreicht man durch eine Kette von D-Flipflops, deren Takteingänge in der jeweils geforderten Weise mit den einzelnen Signalen beschaltet sind (Abbildung 1.29). Das letzte Flipflop der Kette wird nur dann aktiv werden, wenn alle Signale in der geforderten Reihenfolge geschaltet haben.

Abbildung 1.29 Sequenzdetektor

Auf Grundlage der gezeigten Schaltung kann man ein Codeschloß aufbauen. Wir müssen nur gemäß dem jeweils gewünschten Code die Tasten mit den Takteingängen der Flipflops

verbinden. (Denksportaufgaben: (1) wie kann man ein solches Schloß knacken, (2) wie kann man - mit Überwachungsschaltungen - das Knacken erkennen und verhindern?)

1.3.7. Arbitrierung

Mit Arbitrierung bezeichnet man das Vermitteln konkurrierender Anforderungen. Beispiele: Zwei Prozessoren wollen gleichzeitig auf einen gemeinsamen Speicher zugreifen, mehrere Einrichtungen wollen gleichzeitig Daten über ein Bussystem übertragen usw. In solchen Fällen können (1) mehrere Anforderungen gleichzeitig entstehen, es kann aber (2) nur jeweils eine Anforderung bedient werden. Um die zu bedienende Anforderung auszuwählen, braucht man eine Vermittlungseinrichtung. Dual-Port-RAMs und Bussysteme haben eine "eingebaute" Vermittlungslogik. Hier betrachten wir elementare Vermittlungsschaltungen, wie sie gelegentlich in Speichersubsystemen, in Bussteuerungen, an Interfaces usw. verwendet werden.

Asynchrone Vermittlung
Wenn es nur um zwei anfordernde Einrichtungen geht, kann man auch ohne Takt vermitteln. Zwei hintereinandergeschaltete RS-Latches gewährleisten, daß auch bei zwei gleichzeitig anliegenden Anforderungssignalen (/REQUEST1, /REQUEST2) beide Ausgänge der Schaltung unterschiedlich belegt sind (Exclusiv-Latch; Abbildung 1.30). Die disjunktive Verknüpfung der Anforderungssignale liefert zusätzlich die Aussage, daß überhaupt eine Anforderung anhängig ist (REQUEST PENDING). Die auswertende Schaltung kann daraufhin gemäß der Ausgangsbelegung des Exclusiv-Latch die jeweilige Anforderung bedienen (GRANT RQ1 bzw. RQ2).

Abbildung 1.30 Asynchrone Vermittlung (Exclusiv-Latch)

Erklärung: Die auswertende Schaltung darf die GRANT-Signale nur bei aktivem REQUEST PENDING auswerten. Wird die aktuell vermittelte Anforderung inaktiv und ist keine weitere Anforderung anhängig, verbleibt das Latch in der jeweiligen Lage.

Synchrone Vermittlung
Die Taktsteuerung ermöglicht es, eine Vielzahl von Anforderungen zu berücksichtigen und auch vielfältige "Vermittlungsstrategien" zu verwirklichen (Abbildung 1.31). Die anhängigen Anforderungen werden in ein Anforderungsregister übernommen. Aus dessen Belegung wird mit kombinatorischen Verknüpfungen (Prioritätsnetzwerk, Priority Encoder) die jeweils nächste zu bedienende Anforderung ermittelt. Demgemäß wird ein Steuerregister (für den nachfolgenden Ablauf, z. B. für den Speicherzugriff) geladen. Wichtig ist: Für das Eintaktieren der Anforderungen (in das Anforderungsregister) und für das Schalten der "vermittelten" Steuersignale ist jeweils ein gemeinsamer Takt vorzusehen. Der Abstand zwischen den beiden steuernden Taktflanken ist das Vermittlungsintervall, das für das Durchlaufen des Prioritätsnetzwerkes zur Verfügung steht.

Abbildung 1.31 Synchrone Vermittlungsschaltung

Beachten Sie, daß das Prioritätsnetzwerk an sich beliebig komplex ausgelegt sein kann. So können Speicher vorgesehen sein, die die zuletzt vermittelten Anforderungen registrieren, man kann beliebige Prioritätszuordnungen verwirklichen, wie etwa "rotierende" Prioritäten usw.

(Denksportaufgabe: Wie weisen Sie eine "Mehrfachvermittlung" nach? Erklärung: Dies ist eine Fehlersituation, in der mehr als eine Anforderung vermittelt wird, also mehr als ein GRANT-Signal gleichzeitig aktiv ist.)

2. Register

2.1. Daten- bzw. Halteregister

Register sind Aneinanderreihungen von Flipflops bzw. Latches mit gemeinsamen Takt. Abbildung 2.1 gibt eine Übersicht über die verschiedenen Ausführungsformen, Abbildung 2.2 veranschaulicht einige Einzelheiten.

Abbildung 2.1 Daten- bzw. Halteregister: eine Übersicht

Abbildung 2.2 Einzelheiten: Register mit durchlaufendem und mit gesteuertem Takt

Hinweis:
Übliche Registerschaltkreise enthalten lediglich Flipflops oder Latches, deren Takt herausgeführt ist und somit von außen gesteuert werden muß. Ein durchlaufender Takt ist bei einigen Schieberegister- und Zähler- Schaltkreisen vorgesehen. In "vollsynchronen" Schaltungen werden deshalb solche Schaltkreise gelegentlich als einfache Register genutzt.

Takttoleranzprobleme
Oft ist es nicht möglich, ein einziges Taktsignal an allen Registern absolut gleichzeitig wirken zu lassen. Vielmehr sind Treiberstufen, steuernde Gatter usw. zwischengeschaltet. Das hat zur Folge, daß die Taktimpulse an den Eingängen der einzelnen Register zeitlich gegeneinander verschoben sind (Takttoleranzen; Clock Skew). Was bedeutet dies praktisch?

- die Takttoleranzen müssen von vornherein bei der Festlegung der Zeitabläufe eingerechnet werden.
- Takttoleranzen müssen in Problemfällen überprüft werden.
- bei Rückführungen: wenn Register, die einen "zeitigen" Takt erhalten (und gegebenenfalls nachgeordnete Kombinatorik,) eher umgeschaltet haben als der "späteste" Takt an einem in die Rückführung einbezogenen Register ankommt, gibt es Fehlfunktionen.

Rücklesbare Register (Read-Back Latches)
Diese sind zum Anschluß an bidirektionale Datenbusleitungen vorgesehen. Ein Taktimpuls bewirkt die Übernahme der Datenbusbelegung, die daraufhin an den Ausgängen des Registers erscheint. Der Registerinhalt kann über zusätzliche Treiberstufen auf den Datenbus gelegt und so (vom ansteuernden Prozessor) zurückgelesen werden (Abbildung 2.3).

Abbildung 2.3 Register mit Rücklesemöglichkeit (Read-Back Latches; Texas Instruments)

2.2. Einzelbitauswahl (Addressable Latches)

Es gibt Registerschaltkreise, bei denen man jedes Bit einzeln auswählen und gemäß einer Eingangsbelegung stellen kann. Abbildung 2.3 zeigt ein Beispiel.

Abbildung 2.4 Register mit Einzelbitauswahl (Addressable Latch '259)

2.3. Schieberegister

Im Schieberegister sind die Flipflops hintereinandergeschaltet. Vom Dateneingang wird ein Bit in das erste Flipflop geschrieben, mit dem nächsten Takt in das nächste Flipflop übernommen usw. Bei n Flipflops erscheint dieses Bit nach n Taktimpulsen am Ausgang des Schieberegisters. Ein solches Schieberegister wird als Serial In-Serial Out bezeichnet. Abbildung 2.5 gibt einen Überblick über die verschiedenen Arten von Schieberegistern.

Abbildung 2.5 Schieberegister

Takttoleranzproblem
Der Takt in nachfolgenden Flipflops darf nicht später wirksam werden als in vorgeordneten, da sonst die gelieferten Bits nicht übernommen werden können, sondern verlorengehen. Abbildung 2.6 veranschaulicht Problem und Lösungsmöglichkeiten.

Abbildung 2.6 Takttoleranzproblem bei Schieberegistern

Hinweis:
Takttreiber können bei "langen" Schieberegistern erforderlich sein, die aus mehreren Schaltkreisen aufgebaut sind.

Vorzeichenerweiterung
Es gibt Schieberegister-Schaltkreise, bei denen ein Verschieben mit Vorzeichenerweiterung vorgesehen ist: bei Rechtsverschiebung bleibt das höchstwertige Bit erhalten und wird mit jedem Taktimpuls in eine weitere niederwertige Bitstelle übernommen.

Parallelausgabe
Wenn die einzelnen Flipflops des Schieberegisters ausgangsseitig zugänglich sind, kann man den Registerinhalt parallel abnehmen (Abbildung 2.7). Das Problem: beim Schieben ändern sich ständig die Belegungen. Abhilfe schafft ein Zwischenregister (Pufferregister), in das die jeweils auszugebende Belegung zum passenden Zeitpunkt übernommen wird (Abbildung 2.8).

Abbildung 2.7 Schieberegister mit Parallelausgabe ('164; Texas Instruments)

Abbildung 2.8 Schieberegister mit Parallelausgabe über Pufferregister ('594; Texas Instruments)

Paralleleingabe (Laden)
Zum parallelen Laden müssen die Schieberegister-Flipflops eingangsseitig zugänglich sein (Abbildung 2.9). Wenn man laden will, ohne auf das Schieben Rücksicht nehmen zu müssen, braucht man ein vorgeschaltetes Pufferregister (Abbildung2.10).

Abbildung 2.9 Schieberegister mit Paralleleingabe ('165; Texas Instruments)

Abbildung 2.10 Schieberegister mit Paralleleingabe über Pufferregister ('597; Texas Instruments)

Synchrones und asynchrones Laden
Ein Schieberegister (aber auch z. B. ein Zähler) hat flankengesteuerte Flipflops, sonst würde das Schieben (bzw. Zählen) gar nicht funktionieren. Bei der Paralleleingabe gibt es aber Unterschiede:

- das *synchrone* Laden wird durch denselben Takt gesteuert wie das Schieben bzw. Zählen. Die Flipflops verhalten sich auch beim Laden flankengesteuert und können somit in Rückführungen einbezogen werden.
- das *asynchrone* Laden ist hingegen ein durch ein Übernahmesignal gesteuertes Setzen (beim Laden einer 1) oder Rücksetzen (beim Laden einer Null). Beim Laden verhält sich also das Flipflop wie ein Latch. Das in Abbildung 2.9 gezeigte Schieberegister ist so ausgeführt.

Pufferregister
Sowohl vor- als auch nachgeschaltete Pufferregister können entweder mit transparenten Latches oder mit flankengesteuerten Flipflops aufgebaut sein.

Serializer/Deserializer (SERDES)
Die Kopplung aus einem eingangsseitigen Pufferregister, einem Schieberegister und einem ausgangsseitigen Pufferregister ist eine wichtige Grundschaltung zur Parallel- Serien- und Serien-Parallel-Wandlung (man spricht kurz von Serialisierung und Deserialisierung). Derartige Baugruppen findet man in Floppy-Disk- und Festplatten-Controllern, in Netzwerk- (LAN-) Adaptern usw.

Die universelle Bedeutung des Schieberegisterprinzips
Das Schieberegister hat einen wichtigen Vorteil: man kann es so lang ausführen wie man will und braucht, um Bits zu transportieren, nur einen Dateneingang, einen Datenausgang und einen Takt. Da man mit den übertragenen Bits natürlich noch etwas anfangen will, braucht man zusätzlich irgendwelche Steuersignale. Dafür reicht sogar eine einzige Leitung: wird sie aktiviert, so wird die eingeschobene Information übernommen und durch auszugebende Information ersetzt, die dann nur noch herausgeschoben werden muß.

Bedenken Sie: grundsätzlich muß man bei der Auslegung eines Interfaces Steuerungsaufwand und Leitungsaufwand gegeneinander abwägen. Eine Vielzahl von Einzelleitungen hat keinen Steuerungsaufwand, aber einen hohen Leitungsaufwand. Eine rein bitserielle Übertragung hat geringsten Leitungsaufwand, aber hohen Steuerungsaufwand (Serialisierung → Modulation → Demodulation → Deserialisierung; Ablaufsteuerung). Das Schieberegister hat hier eine bemerkenswerte Zwischenstellung: nur wenige Leitungen, aber sehr geringen Steuerungsaufwand.

Wichtige Anwendungen des Schieberegisterprinzips sind:

- elementare Ablaufsteuerschaltungen, insbesondere dann, wenn vorwiegend zeitstarre Impulsfolgen benötigt werden (Beispiel: DRAM-Ansteuerung),
- das Prüfen von Digitalschaltungen (Stichworte: Scan Design und Boundary Scan),
- das Programmieren programmierbarer Schaltkreise (GALs, EPLDs usw.); diese haben serielle Schiebewege für Programmierdaten,
- Elementar-Interfaces zwischen hochintegrierten Schaltkreisen.

Das Schieberegister als Elementar-Interface
Das Schieberegisterprinzip wird mehr und mehr als kostengünstiges, universelles und funktionssicheres Interface zur Verbindung von Schaltkreisen und Funktionseinheiten innerhalb von Geräten (ja sogar auf Leiterplatten) verwendet (eine typische Anwendung ist z. B. der Anschluß eines Bedienfeldes an den zentralen Mikrocontroller).

Microwire
Microwire ist ein von National Semiconductor entwickeltes Interface zum Verbinden peripherer Einrichtungen mit einer zentralen Steuerung (üblicherweise einem Mikrocontroller). Das Interface umfaßt - an der angeschlossenen Einrichtung - 4 Leitungen: Dateneingang DI, Datenausgang DO, Schiebetakt SK und Schaltkreisauswahl CS (Abbildung 2.11). Es ist auch möglich, Microwire-Verbindungen mit 3 Leitungen aufzubauen; DI und DO sind dann zu einer bidirektionalen Datenleitung zusammengefaßt.

Abbildung 2.11 Der Microwire-Bus

Die zentrale Steuerung wählt jeweils eine angeschlossene Einrichtung aus, indem sie die betreffende CS-Leitung aktiviert. Über die DI-Leitung kann dann bereits der Verfügbarkeits-Zustand der Einrichtung abgefragt werden. Auszugebende Information wird über DO geliefert und in der ausgewählten Einrichtung mit der Vorderflanke der SK-Impulse übernommen; einzugebende Information wird über DI mit SK-Taktimpulsen Bit für Bit abgeholt.

Serien-Parallelwandler-Schaltkreise
Die Schaltkeise CXP200... der Fa. Sony enthalten ein Schieberegister und ein paralleles Ausgaberegister (Abbildung 2.12). Es gibt Ausführungen für 8, 12, 16 und 24 Bits. Mit Impulsen am Eingang SCK werden die Datenbits in das Schieberegister übernommen. Ein Impuls über STB lädt das Ausgaberegister (dieses ist - z. B. beim Einschalten - über CLR löschbar).

Abbildung 2.12 Serien-Parallelwandler-Schaltkreis CXP2003 (Sony)

Leistungsschalter mit seriellem Eingang
Die Fa. Texas Instruments hat einige originelle Schaltkreise entwickelt, die Logik und Leistungsstufen vereinigen (TPIC-Serie; Power Logic). Der Typ TPIC6595 ist ein Serien-Parallelwandler ähnlich Abbildung 2.12, aber mit MOS-Transistoren an den Ausgängen, die bis zu 250 mA schalten können (impulsweise sogar bis 1,5 A; als Ausgangsspannung sind bis zu 45 V zulässig). Abbildung 2.13 zeigt Schaltsymbol und Innenschaltung.

Abbildung 2.13 8-Bit-Schieberegister TPIC6595 der Power-Logic-Reihe (Texas Instruments)

3. Zähler und Teiler

Zähler und Teiler liefern ihre Ausgangssignale in Abhängigkeit von der *Anzahl* der einlaufenden Taktimpulse.

Bezeichnungsweise
Bei einem Zähler (Counter) kann man die "irgendwie" codierte aktuelle Anzahl der gezählten Taktimpulse parallel abnehmen, beim Teiler (Divider) hingegen nicht (anders ausgedrückt: Teiler haben nur einen Ausgang, Zähler soviele, wie Flipflops vorgesehen sind). Wir werden deshalb Zähler und Teiler weitgehend gemeinsam behandeln und dabei "Zähler" als Allgemeinbegriff verwenden.

Codierung
Hat man n Flipflops, so lassen sich damit maximal 2^n verschiedene Zustände codieren und also auch abzählen. Die Codierung des Zählers bestimmt, wie die einzelnen Flip-flop-Belegungen bei aufeinanderfolgenden Taktimpulsen gebildet werden bzw. auseinander hervorgehen. Wir betrachten im folgenden nur jene Codierungen, die (1) in der Praxis häufig vorkommen, für die es (2) fertige Zählerschaltkreise gibt bzw. die (3) vergleichsweise einfach aufgebaut werden können. Abbildung 3.1 gibt einen Überblick über die verschiedenen Arten von Zählern und Teilern.

Abbildung 3.1 Zähler und Teiler: eine Übersicht

Zählen "modulo n"
Das ist die Bezeichnung dafür, daß sich nach jeweils n Taktimpulsen der Zählablauf wiederholt. Gleichbedeutender Begriff: *Zählweite*.

Der Zähler als State Machine
Herkömmlicherweise wird oft nur der Binär- oder Dezimalzähler als Zähler im eigentlichen Sinne angesprochen. Diese Auffassung aber ist fachlich nicht exakt und erschwert das Verständnis moderner, auf programmierbaren Schaltkreisen beruhender Entwürfe. Wichtig sind zunächst allein die Anzahl der Maschinenzustände und vorgesehenen Zustandsübergänge, unabhängig von der Zustands*codierung* (Näheres in Kapitel 4). Wir merken uns deshalb: Ein modulo-n-Zähler ist eine State Machine mit n Zuständen, wobei jeder Takt einen Zustandswechsel zum nächsten Zustand veranlaßt und nach n Takten wieder der erste Zustand eingenommen wird (Abbildung 3.2). Dieser einfache Ablauf (ständiges Zählen) wird oft durch Zustände des Ladens und durch Ruhezustände modifiziert.

Abbildung 3.2 Der modulo-n-Zähler als State Machine

3.1. Ringzähler

Ringzähler zählen mit n Flipflops im 1-aus-n-Code modulo n. In jedem Zustand ist ein Flipflop aktiv, während die anderen inaktiv sind. Ein solcher Zähler ist praktisch ein

Schieberegister, in dem eine Eins zyklisch umläuft. Man muß dazu den Ausgang auf den Eingang zurückführen und dafür sorgen, daß anfänglich ein Flipflop gesetzt wird, während alle anderen gelöscht werden (Abbildung 3.3).

Abbildung 3.3 Ringzähler

Wird die Rückführung über eine NAND- oder NOR- Verknüpfung der ersten n-1 Flipflops geführt (Abbildung 3.4), so schwingt sich die Schaltung nach spätestens n Taktimpulsen selbst ein. Bei NAND-Rückführung liefern die Flipflop-Ausgänge eine invertierte 1-aus-n-Codierung, bei NOR-Rückführung eine "wahre". Solche Schaltungen sind beispielsweise zur Bildung von Mehrphasen-Takten üblich.

Abbildung 3.4 Selbst einschwingende Ringzähler

3.2. Johnson-Zähler

Ein solcher Zähler zählt mit n/2 Flipflops modulo n. Dazu wird das Ringzählerprinzip abgewandelt: (1) wird die invertierte Ausgangsbelegung zurückgeführt, (2) werden anfänglich alle Flipflops gelöscht. Abbildung 3.5 veranschaulicht Schaltung und Zählweise.

Abbildung 3.5 Johnson-Zähler

Hinweis:
Die Schaltkreise CD4017, 4018 und 4022 sind Johnson-Zähler. Sie sind selbst-einschwingend ausgelegt, brauchen also nicht unbedingt ein Rücksetzen, um korrekt "anzulaufen". Der 4018 ist ein modulo-10-Zähler mit 5 Ausgängen, an denen das Verhalten eines Johnson-Zählers beobachtet werden kann. Die Typen 4017 und 4022 haben hingegen im 1-aus-n-Code erregte Ausgänge (10 bzw. 8).

3.3. Binärzähler

Beim Binärzähler entspricht jedes Flipflop einer Binärstelle. Mit n Flipflops beträgt die maximale Zählweite 2^n. Die einzelnen Werte liegen zwischen 0 und 2^n - 1. Binärzähler können synchron oder asynchron aufgebaut sein. Sie können für eine beliebige (kleinere) Zählweite und für beide Zählrichtungen (vorwärts bzw. rückwärts) eingerichtet werden.

Asynchronzähler (Ripple Counter)
Diese werden mit T-Flipflops aufgebaut (Abbildung 3.6).

Abbildung 3.6 Asynchronzähler

Das ist die einfachste Zählerschaltung. Sie hat aber einen wesentlichen Nachteil, den wir sofort erkennen: jedes Flipflop schaltet erst, nachdem das vorhergehende geschaltet hat. Bei n Flipflops dauert es somit n Flipflop-Schaltzeiten, bis der neue Zählwert gebildet ist.

Synchronzähler
Alle Flipflops sind gemeinsam mit dem Zähl-Takt beschaltet. Für jedes Flipflop werden besondere Zählbedingungen mit kombinatorischen Verknüpfungen gebildet. Immer dann, wenn eine solche Bedingung aktiv ist, schaltet das betreffende Flipflop entsprechend um. Die Bedingungen lassen sich für Vorwärts- und Rückwärtszählen einrichten (Abbildung 3.7).

Abbildung 3.7 Synchrone Binärzähler

Beliebige Zählweite
Auf dem Prinzip der binären Zählung beruhend kann man eine beliebige Zählweite verwirklichen. In fertigen Schaltkreisen ist allerdings neben der rein binären praktisch nur noch die dezimale Zählung (mit den BCD-Werten von 0 bis 9) vorgesehen. Um eine beliebige Zählweite einzustellen, muß man entweder die Schaltbedingungen für die Flipflops entsprechend bestimmen, man muß die letzte gewünschte Stellung decodieren und den Zähler damit zurücksetzen, oder man muß ihn entsprechend voreinstellen, so daß er nicht von Null an, sondern von einem Anfangswert $a = 2^n - 1 - z$ zählt, wobei z die gewünschte Zählweite angibt (Abbildung 3.8).

Abbildung 3.8 Zählen mit beliebiger Zählweite

Zählen durch Addieren bzw. Subtrahieren
Man kann einen Zähler auch aufbauen, indem man ein Flipflop-Register mit einem Addiernetzwerk zusammenschaltet.

Damit ist grundsätzlich jede Schrittweite und Zählrichtung realisierbar. Für die Schrittweite Eins kann man das Addiernetzwerk deutlich vereinfachen: Man muß nur in der niedrigstwertigen Stelle addieren; in den verbleibenden Stellen ist dann nur noch der Übertrag zu verrechnen, so daß man dort mit Halbaddierern auskommt. Auch für die Subtraktion (zum Rückwärtszählen um Eins) läßt sich das Netzwerk vereinfachen. Solche Schaltungsanordnungen werden auch als Incrementier- und Decrementiernetzwerke bezeichnet (Increment: Erhöhen; Decrement: Vermindern).

Anmerkung: Dieses Schaltungsprinzip ist allgemein üblich, um in Prozessoren Zählfunktionen (z. B. die Befehlszählung) zu verwirklichen.

Decodieren von Zählerständen
Oft braucht man Signale, die bestimmten Zählerstellungen entsprechen. Nur beim Ringzähler sind diese durch die Flipflop-Ausgänge direkt gegeben. Ansonsten müssen den Flipflops Decodierschaltungen nachgesetzt werden. Das Problem: Wir erhalten nicht immer "saubere" Signale. Vielmehr ist, wenn mehrere Flipflops gleichzeitig schalten, durch die unvermeidlichen Unterschiede in den Schaltzeiten mit Störungen (Glitches, Spikes) zu rechnen. Wenn man die Ausgänge der Decodierschaltung zum Umschaltzeitpunkt nicht auswertet, schaden die Störungen nicht.

Hinweis:
Beim Johnson-Zähler ändert sich mit jedem Takt nur eine Flipflop-Belegung, er kann somit *glitchfrei* decodiert werden.

Teiler
Ein üblicher Teiler gibt nach n Eingangsimpulsen einen Ausgangsimpuls ab; er untersetzt die Impulsfolgefrequenz im Verhältnis n:1. Teiler werden vorzugsweise wie Asynchronzähler aufgebaut. Die Vorteile: (1) schnellstes Schalten, da keine kombinatorischen Netzwerke durchlaufen werden müssen, (2) nur an den ersten Flipflop werden die höchsten Geschwindigkeits-Anforderungen gestellt, der zweite wird bereits mit der halben Frequenz betrieben usw.

Vorteiler (Prescaler)
Wenn man Impulse sehr hoher Frequenz zu zählen hat, ist es gelegentlich möglich, durch einen vorgeschalteten Frequenzteiler die Impulsfolgefrequenz in eine beherrschbare Größenordnung zu bringen. So muß ein "digitaler" UKW-Tuner mit Impulsfolgefrequenzen von über 100 MHz zurechtkommen; Tuner für Fernsehen und Mobiltelefon mit einigen hundert MHz bis hinein in den GHz-Bereich (Erklärung: um die Zwischenfrequenz - ZF - hochgenau über eine PLL-Schaltung zu erzeugen, muß die ankommende Hochfrequenz in ein Impulssignal gewandelt und "heruntergeteilt" werden). Dafür werden einzelne Teiler-Flipflops und Vorteiler-Schaltkreise (Prescaler) in besonders "schnellen" Technologien (wie ECL und GaAs) gefertigt (in ECL kommt man bis auf 650...750 MHz, in GaAs bis auf 5 GHz).

Genaues Zählen trotz schneller Vorteilung: Pulse Swallowing
In den Typenlisten der Hersteller finden wir Vorteiler, die - über einen Eingang steuerbar- wahlweise durch 5 oder 6, durch 10 oder 11, durch 128 oder 129 usw. teilen können. Die Abbildungen 3.9, 3.10 zeigen die Innenschaltungen zweier solcher Schaltkreise.

Abbildung 3.9 650-MHz-Vorteiler in ECL-Technologie (1): der 11C90 (National Semiconductor)

Abbildung 3.10 650-MHz-Vorteiler in ECL-Technologie (2): der 11C91 (National Semiconductor)

Erklärung:
Der 11C90 teilt durch 10 oder 11, der 11C91 durch 5 oder 6. Die Schaltkreise sind unmittelbar in einer TTL-Umgebung einsetzbar. Beachten Sie, daß die Zählfunktion durch "trickreiche" Zustandsübergänge verwirklicht ist (es geht darum, einerseits mit wenig Flipflops und andererseits mit geringer Schaltungstiefe der Rückführungen auszukommen).

Wozu ist die Umschaltbarkeit gut? - Man kann damit - trotz Vorteilung - auch bei höchsten Impulsfolgefrequenezen bis auf den einzelnen Impuls genau zählen!

Abbildung 3.11 zeigt den Aufbau eines einstellbaren Teilers, der aus Dezimalzählern aufgebaut ist, die mit dem Komplement des Zählwertes (bzw. Teiler- Faktors) voreingestellt werden.

Abbildung 3.11 Einstellbarer schneller Frequenzteiler (National Semiconductor)

Der erste 4-Bit-Dezimalzähler ist der Steuerzähler. Die zu zählenden Impulse gelangen auf den Vorteiler, der zunächst durch 11 teilt. Jeder 11. Impuls schaltet also den Steuerzähler weiter. Hat er seinen Endwert erreicht, setzt er sich über die Rückführung selbst in Ruhestellung und schaltet den Vorteiler auf 1:10-Teilung um. Dann wird von der zweiten Dezimalstelle "ganz normal" jeder 10. Impuls gezählt. Hat der gesamte Zähler den Endwert erreicht, wird der Steuerzähler wieder aktiviert und der Vorteiler auf 1:11 umgestellt. Dann beginnt das Spiel von neuem.

Funktioniert das auch richtig? - Sei der Steuerzähler zum Zählen von a Impulsen voreingestellt, der "restliche" Dezimalzähler zum Zählen von b Impulsen. Insgesamt müssen also, gemäß der Einstellung, a + 10b Takte abgezählt werden (Stellenwertsystem).

Zunächst werden 11 · a Impulse gezählt. Dann schaltet der Steuerzähler auf 1:10- Teilung um. Der Inhalt des "restlichen" Dezimalzählers ist aber - da dieser ebenfalls vom Vorteiler angesteuert wird - "heruntergezählt" worden, und zwar mit a ("heruntergeteilten") Zählimpulsen. Bis zum Ende sind also noch b-a Zählimpulse erforderlich. Der verbleibende Zählerinhalt wird alle 10 Eingangsimpulse verändert.

Insgesamt werden also gezählt: 11a Impulse

+

10 (b-a) Impulse.

Das ergibt 11a + 10b - 10a = a + 10b Impulse, also genau so viel, wie es die exakte Zählung erfordert. Durch die Schaltung wird also die maximale Zählfrequenz des einstellbaren Zählers praktisch verzehnfacht.

3.4. Typische Zählerschaltkreise

Die meisten Zählfunktionen werden heutzutage in höher integrierten Schaltkreisen verwirklicht. Trotzdem haben wir es gelegentlich noch mit "diskret" aufgebauten Zählern zu tun. Zudem arbeiten heutige Hardware-Entwickler nach wie vor oft mit den bewährten Schaltkreistypen, nur werden die logischen Funktionen automatisch (vom Entwurfssystem) in die jeweilige Technologie umgesetzt (Stichwort: Soft Macros). Wir skizzieren deshalb im folgenden einige typische Schaltkreise (Abbildung 3.12).

Hinweise:

1. Nicht alle Anbieter verwenden die gleichen Signalbezeichnungen. (Wir haben uns hier an Texas Instruments orientiert.)
2. Viele andere Zählerschaltkreise sind ähnlich ausgelegt wie die hier beschriebenen Typen. Das betrifft die Steuerung, die Kaskadierung, das Laden usw.

Abbildung 3.12 Typische Zählerschaltkreise

Dezimal- und Binärzähler (vorwärts/rückwärts) '192, '193
Die Schaltkreistypen '192, '193 sind Vorwärts-Rückwärts-Zähler mit unabhängigen Takteingängen für jede Zählrichtung. Sie haben einige Besonderheiten:

- der einzelne Zählerschaltkreis zählt synchron, Laden und Rücksetzen wirken jedoch asynchron.
- Kaskadierung: "Längere" Zähler kann man durch Hintereinanderschalten mehrerer Schaltkreise aufbauen. Dabei ist der Vorwärts-Übertrag (Carry Out CO) des vorgeordneten Schaltkreises mit dem Vorwärts-Takteingang (UP) des nachgeordneten zu verbinden. Sinngemäß ist der Rückwärts-Übertrag (Borrow Out BO) an den Rückwärts-Takteingang (DWN) anzuschließen. Dies führt also zu einem teilweise asynchronen Zähler (die höchstwertigen Stellen schalten später als die niedrigstwertigen).
- gezählt wird jeweils mit der LO-HI-Taktflanke. Dabei muß der andere Takteingang auf *HI* gehalten werden. (Im Verdachtsfall überprüfen!)
- bei gleichzeitiger Erregung: Laden hat Vorrang vor Zählen, Rücksetzen hat Vorrang vor Laden.

Dezimal- und Binärzähler (vollsynchron) '160...'163
Diese Zählerschaltkreise (sie können nur vorwärts zählen) arbeiten vollsynchron. Es können ständig Taktimpulse anliegen; die jeweilige Funktion kommt dann durch Erregen der entsprechenden Steuereingänge zustande. Besonderheiten:

- bei gleichzeitiger Erregung: Laden hat Vorrang vor Zählen, Rücksetzen hat Vorrang vor Laden.
- Zählerlaubnis: Damit der Schaltkreis zählt, müssen beide Erlaubniseingänge ENT, ENP aktiv sein.
- Kaskadierung: Der Schaltkreis liefert einen Ausgangsübertrag RCO (Ripple Carry Out). Dies ist eine rein kombinatorische Decodierung des Zähler-Endstandes. Störimpulse (Glitches, Spikes) auf RCO sind also nicht notwendigerweise Fehleranzeichen! "Längere" Zähler baut man auf einfachste Weise, indem man RCO an ENP und ENT des Nachfolge-Schaltkreises anschließt.
- Rücksetzen: Bei den Typen mit synchronem Rücksetzeingang ('161, '163) ist zum Rücksetzen ein Taktimpuls erforderlich. Prüfen Sie, ob diese Bedingung erfüllt ist, wenn der Zähler nicht ordnungsgemäß zurückgesetzt wird.
- wegen des vollsynchronen Ladens werden diese Schaltkreise gelegentlich auch als Register ohne Zählfunktion eingesetzt.

Im folgenden zeigen wir einige typische Sonderschaltungen mit diesen Schaltkreisen.

Schnelle Kaskadierung
Die Schaltung (Abbildung 3.13) zeigt, weshalb zwei Zählerlaubniseingänge vorgesehen sind. Das Problem: Der Ausgangsübertrag RCO muß, damit alles funktioniert, kombinatorisch aus Zähler-Endstellung und einem einlaufenden Übertrag gebildet werden (ein Zählerschaltkreis an n-ter Position darf erst dann weiterzählen, wenn alle vorgeordneten Schaltkreise in die Endstellung gelangt sind). Dieser einlaufende Übertrag

wird über ENP zugeführt. Damit hat man - bei mehreren Schaltkreisen "in Reihe" eine beachtliche Verzögerungszeit des Übertragssignals der "letzten" Schaltkreise (alle vorgeordneten Schaltkreise müssen nacheinander durchlaufen werden).

Der Ausweg: An der zweiten Stufe wird ENT fest aktiviert, und die 2., 3. usw. Stufen erhalten über ENP den Ausgangsübertrag der ersten zugeführt. Die Weiterschaltung von RCO erfolgt dann über die Erlaubniseingänge ENT von der 3. Stufe an. Der Erfolg: Wenn die erste Stufe einen Ausgangsübertrag abgibt, wird sofort die Zählerlaubnis aller anderen Stufen parallel aktiviert; es gibt keine Zeitverluste. Da die folgenden Stufen nur bei jedem "Umlauf" der ersten Stufe weiterzählen müssen, steht für das Durchschalten des "seriellen" Ausgangsübertrags genügend Zeit zur Verfügung.

Abbildung 3.13 Schnelle Kaskadierung mit '160...'163

Weitere Beschleunigung

Die Schaltung von Abbildung 3.14 zeigt ein weiteres Prinzip, das in schnellen Zählschaltungen oft verwendet wird. Das Problem: es bereitet keine Schwierigkeiten, fortlaufend zu zählen. Nur beim Erreichen der Endstellung und beim Auslösen der dann notwendigen Maßnahmen (üblicherweise ein Laden und Zählen von Anfang an) kann die Zeit knapp werden. Der Ausweg: Man decodiert die letzten Zählerstellungen und steuert den Umlauf (das parallele Laden) über gesonderte Flipflops (diese sind schneller, da sie direkt taktiert werden und nicht von Übertragssignalen abhängen). Im Beispiel ist ein Flipflop gezeigt; man kann aber auch (zeitig genug) ein Schieberegister ansteuern, das dann die beim Zähler-Umlauf auszuführenden Wirkungen erbringt.

Abbildung 3.14 Weitere Beschleunigung ('160...'163)

Zählen mit einstellbarer Zählweite

Abbildung 3.15 zeigt ein entsprechendes Beispiel.

Abbildung 3.15 Einstellbarer Zähler (modulo-n-Zähler) mit '160...'163

Die Schaltung ist einfach, erfordert aber das Einstellen des Zweier- oder des Zehnerkomplements. Kein Problem, wenn es sich um einen Festwert handelt oder um einen Wert, der softwareseitig eingestellt wird. Ansonsten ist die hardwaremäßige Komplementbildung nicht ganz einfach. Abbildung 3.16 zeigt einen Ausweg: der Umlauf wird nicht über RCO bei "Endanschlag" ausgelöst, sondern - über zusätzliche Decodierung - eine Stellung vorher (dezimal: Stellung 8, binär: Stellung 14). Somit können die Zähler mit dem leichter zu bildenden Neuner- oder Einerkomplement beschaltet werden.

Abbildung 3.16 Zählen im Neuner- oder Einerkomplement ('160...'163)

Teiler-Flipflops als Prüf- und Testhilfe

Ein 2:1-Teiler macht aus jeder Folge schmaler Impulse einen schönen symmetrischen Mäander (Abbildung 3.17). Die Nutzanwendung: Signale, die ansonsten schwer zu beobachten sind, an eine Teilerstufe anschließen - und schon kann man sie mit einem

üblichen Oszilloskop wenigstens überschlägig kontrollieren - ja man kann sie gelegentlich sogar mit Software abfragen. (Manche Steuerschaltkreise haben entsprechende “Toggle”-Funktionen eingebaut. Es handelt sich um Testbetriebsarten, in denen Signale, die an sich schwer zu erfassen sind, an programmseitig abfragbare 2:1-Teiler geführt werden.)

1. Beispiel: Wie messen Sie die Refresh-Intervalle im (herkömmlichen) PC? Es handelt sich um Impulse von wenigen hundert ns, die etwa alle 15 µs auftreten. - Bei Anschluß an einen 2:1-Teiler können Sie die Refresh-Periode unmittelbar auf dem Schirm des Oszilloskops (anhand der Skalenteilung) auszählen.

2. Beispiel: Wie überprüfen Sie einen 100-MHz-Takt? - 2:1-Teilung ergibt 50 MHz, 4:1-Teilung ergibt 25 MHz usw.

3. Beispiel: Mit einem sehr schnellen Teiler kann man extrem schmale Störimpulse nachweisen.

Abbildung 3.17 Teiler-Flipflop als Prüfhilfe

4. Reguläre sequentielle Steuerschaltungen (State Machines)

Grundlagen

Selbstverständlich kann man Steuerschaltungen sozusagen intuitiv aufbauen, also Gatter, Flipflops usw. so verschalten, wie es gerade zweckmäßig erscheint. Solche Schaltungen sind nicht sehr übersichtlich, und sie bereiten nicht nur dem Servicetechniker (der sich darin zurechtfinden soll) Kummer, sondern sie erfordern schon in der Entwicklung oft einiges an Änderungen, bis sie zufriedenstellend funktionieren.

Reguläre, übersichtliche Schaltungsprinzipien sind deshalb sehr wünschenswert. Die direkte Anwendung automatentheoretischer Überlegungen bietet hier eine Lösung. Man geht von den funktionell notwendigen *Maschinenzuständen* aus, ordnet jedem Zustand eine bestimmte Belegung der Speichermittel (Flipflops) zu (Zustandscodierung) und gestaltet die kombinatorischen Verknüpfungen so, daß sie zu jedem Taktzeitpunkt aus aktuellem Zustand und aktueller Eingangsbelegung den Folgezustand und die geforderte Ausgangsbelegung bestimmen (Funktionszuordner). Abbildung 4.1 zeigt das grundsätzliche Schema einer solchen State Machine (sprich: Steht Mäschiehn - leider gibt es keinen allgemein akzeptierten deutschsprachigen Fachbegriff).

Abbildung 4.1 State Machine

Eine Tendenz im modernen Hardware-Entwurf besteht darin, Steuerschaltungen in überschaubare State Machines zu zerlegen und diese über Zustandsgraphen und die Schaltgleichungen der Funktionszuordner zu dokumentieren.

Die Funktionszuordner selbst werden oft mit programmierbaren Schaltkreisen realisiert. Wenn es nicht auf die letzte ns ankommt, kann man sogar das Schema von Abbildung 4.1

"wörtlich nehmen" und die Funktionszuordner mit (PROM-) Speicherschaltkreisen aufbauen (Abbildung 4.2). Die Vorteile: unbeschränkte Komplexität (auch komplizierteste Zustandsübergänge kosten keinen Mehraufwand), größte Einfachheit, leichte Änderbarkeit. Passende PROM-Register-Anordnungen gibt es auch als fertige Schaltkreise (State Machine PROMs).

Abbildung 4.2 State Machine mit PROM-Zuordnern

Zustandscodierung
Wenn man n Maschinenzustände (States) vorsehen muß - wieviele Flipflops werden dazu benötigt und wie ordnet man diese den einzelnen Zuständen zu? - Dies ist das Problem der Zustandscodierung. Grundsätzlich gibt es eine unabsehbare Vielfalt von Codierungsmöglichkeiten (auch sehr trickreiche), aber nur wenige haben in der Praxis weite Verbreitung gefunden:

Binärcodierung
n Zustände werden mit ceil (ld n) Flipflops binär codiert. Dies ergibt die geringste Flipflopanzahl, erfordert aber oft einen komplexen Funktionszuordner (viele Gatter, hohe Schaltungstiefe).

1-aus-n-Codierung.
Ganz einfach: jeder Zustand wird durch einen Flipflop repräsentiert. Man braucht n Flipflops für n Zustände, wobei jeweils nur ein Flipflop aktiviert ist (im Fachenglisch spricht man auch vom One-Hot Encoding, OHE). Das erfordert vergleichsweise viele Flipflops. Dafür wird die Zuordner- Kombinatorik einfach und schnell.

Gemischte Codierung
Manchmal sind bestimmte Codierungen in der Schaltungspraxis einfach nicht anwendbar. Einen 20-Bit-Zähler (über eine Million Zustände!) wird man kaum mit 1-aus-n-Codierung verwirklichen. So liegt es nahe, Zustände, die durch Zählprozesse aufeinanderfolgen, binär zu codieren und solche, zwischen denen kompliziertere Übergänge vorgesehen sind, im 1-aus-n-Code. Ein weiteres Beispiel ist die Abwandlung des 1-aus-n-Codes dahin, daß der "Ruhezustand" (Idle State) nicht durch ein gesondertes Flipflop, sondern durch die Nullbelegung aller verbleibenden Flipflops codiert wird.

Auswahl
Ein guter Schaltungsentwickler strebt minimale Kosten im Rahmen der gegebenen Technologie an. Akademische Minimalkriterien (etwa aus der Schaltalgebra und Automatentheorie) sind dabei nur bedingt hilfreich. So ist es ein wichtiger Gesichtspunkt, ob man die State Machine noch in einem bestimmten GAL oder Gate Array unterbringt. Moderne Technologien legen es nahe, die 1-aus-n-Codierung zu bevorzugen: an Flipflops herrscht kaum Mangel, aber komplizierte Verknüpfungsnetzwerke fressen Siliziumfläche bzw. sind (im einzelnen programmierbaren Schaltkreis) gar nicht realisierbar. Hingegen ist bei PROM-Realisierung oft die binäre (minimale) Zustandscodierung besser, um die Anforderungen an die Speicherkapazität zu reduzieren (ein Bit weniger, und Sie kommen mit einem halb so großen PROM aus!). Die Kompliziertheit der kombinatorischen Verknüpfungen spielt hingegen beim PROM keine Rolle. Das gilt sinngemäß auch für CPLDs (nur wenige Flipflops, aber “dicke” UND-ODER-Knoten).

Schaltungsbeispiele

Start-Stop-Generator
Ein Start-Stop-Generator wird einem "durchlaufenden" Taktgenerator nachgeschaltet. Er soll folgende Betriebsarten haben:

1. durchlaufender Takt (Laufzustand; Running): die Taktimpulse des Taktgenerators erscheinen fortlaufend am Ausgang,
2. kein Takt (Stopzustand, Stopped): infolge einer *Stopbedingung* wird der Takt "angehalten"; es gelangen keine Taktimpulse mehr zum Ausgang,
3. Schrittbetrieb (Single Step): Wird eine Taste betätigt, so gelangt ein einziger Taktimpuls zum Ausgang. Diese Betriebsart ist nur im Stopzustand auslösbar.

Stopbedingung: Diese wird wirksam (1) durch ein von außen zugeführtes Stopsignal oder (2) durch Umlegen eines Betriebsarten-Wahlschalters.

Übergang aus dem Stopzustand in den Laufzustand: durch Tastenbetätigung.

Abbildung 4.3 zeigt den Zustandsgraphen, Abbildung 4.4 eine Schaltungslösung.

Hinweise:

1. Wir haben hier eine Praxisschaltung mit allen Einzelheiten gezeigt. Dies hat zusätzlich noch eine Betriebsart "Takt von Hand" (MAN), in der das Ausgangssignal direkt von der Starttaste abgeleitet wird.
2. Wenn aufschmale, asynchrone Impulse zu stoppen ist, muß dem STOP- Eingang eine Fangschaltung vorgeschaltet werden (Impulsfalle).
3. Die eigentliche Start-Stop-Schaltung hat nur zwei Zustände, die mit einem Flipflop codiert sind. Das zweite Flipflop dient zum Fangen des Tastendrucks; es wird im Laufzustand zurückgesetzt und ist somit (im Schrittbetrieb) nach der Rückkehr in den Stopzustand wieder in der Lage, auf eine Flanke vom Tasten- Entprell-Latch (vgl. Lehreinheit 5) zu reagieren.

Abbildung 4.3 Zustandsgraph eines Start-Stop-Generators

Abbildung 4.4 Start-Stop-Generator

Eine solche Schaltung ist eine nützliche Hilfe beim Erproben von und Experimentieren mit digitalen Schaltungen. Anwendung: der Takt der Schaltung, die Sie "in Arbeit" haben, wird über den Start-Stop-Generator geliefert. Im Schrittbetrieb können Sie Takt für Takt "von Hand" durchgehen und Signale in aller Ruhe statisch überprüfen. Auch kann man eine Bedingung erfassen und den zunächst laufenden Takt anhalten, um dann Schritt für Schritt weiterzugehen.

Digitale Zeitstufen

Eine digitale Zeitstufe gibt, wenn von außen durch einen Impuls angeregt, einen Ausgangsimpuls definierter Länge ab (diese Länge wird durch einen Zählvorgang bestimmt).

Wir unterscheiden zwischen nicht retriggerbaren und retriggerbaren Zeitstufen. Bei einer nicht retriggerbaren Zeitstufe haben weitere, nach der Aktivierung ankommende Impulse keinen Einfluß auf den Zeitablauf. Hingegen bewirkt bei einer retriggerbaren Zeitstufe jeder neue Eingangsimpuls, daß die Zeitzählung von neuem beginnt; der Ausgangsimpuls wird also entsprechend verlängert. Abbildung 4.5 zeigt die Zustandsgraphen beider Ausführungen.

Abbildung 4.5 Digitale Zeitstufen

Hinweise:

1. Wir haben hier die vielen aufeinanderfolgenden Zähler-Zustände zu einem einzigen Knoten "while counting" zusammengefaßt (ein übliches Verfahren, Zustandsgraphen verständlich zu gestalten).
2. Zähler können (durch Kaskadierung) beliebig "lang" ausgeführt werden.

Abbildung 4.6 ist das Schaltbild einer nicht retriggerbaren Zeitstufe. Wir haben hier Zähler der Typen '192, '193 eingesetzt, um deren Ansteuerung zu veranschaulichen.

Abbildung 4.6 Nicht retriggerbare Zeitstufe

Abbildung 4.7 zeigt eine retriggerbare Zeitstufe; diesmal in vollsynchroner Auslegung mit Zählern '160...'163.

Abbildung 4.7 Retriggerbare Zeitstufe

DRAM-Ablaufsteuerung

Die in elementaren DRAM-Subsystemen erforderlichen zeitversetzten Impulse (RAS, CAS, WE, OE) kann man noch mit "analogen" Verzögerungsschaltungen erzeugen. Besser sind taktgesteuerte, synchron arbeitende Steuerschaltungen.

Es liegt nahe, die zeitversetzten Impulse mit einem Schieberegister zu erzeugen. Die Abbildungen 4.7 bis 4.9 zeigen ein Beispiel. Dabei wird ein DRAM auf dreifache Weise angesteuert: (1) durch interne Zugriffs-Anforderungen (LOCAL REQUESTs), (2) durch Zugriffs-Anforderungen über den Systembus (SLAVE REQUESTs) und (3) durch Refresh-Anforderungen (REFRESH REQUESTs).

Hinweis:
Es handelt sich um eine Praxisschaltung. Lassen Sie sich von - hier unwesentlichen - Einzelheiten nicht ablenken. Das Wesentliche: Einer Vermittlungsschaltung ist ein Schieberegister nachgeordnet, das im Ruhezustand "leer" ist. Mit Zyklusbeginn (CYCLE RUNNING) wird eine Eins durchgeschoben. Aus den zeitversetzten Ausgangssignalen werden die benötigten Steuerimpulse gebildet. Am Ende (END oder RELEASE) wird das Schieberegister parallel mit Null geladen.

Abbildung 4.8 Vermittlungs- und Ablaufsteuerschaltung für DRAM-Zugriffe

Abbildung 4.9 Einzelheiten der Ablaufsteuerschaltung

Abbildung 4.10 Ablaufschema der DRAM-Zugriffe

DRAM-Steuerschaltungen für moderne Hochleistungssysteme fallen noch um einiges komplizierter aus. Die Darstellung als detaillierter Schaltplan wäre viel zu unübersichtlich. Man entwirft deshalb die Schaltung unmittelbar als State Machine und dokumentiert sie über Zustandsgraphen und Schaltgleichungen. Diese Beschreibung wird dann in CPLDs verwirklicht (einen Schaltplan braucht man auch gar nicht, weil die Entwurfssoftware aus den Gleichungen heraus unmittelbar die Programmierdaten für die CPLD-Schaltkreise ableitet). In den Abbildungen 4.11 und 4.12 wollen wir dies anhand eines ausschnittsweise wiedergegebenen Beispiels veranschaulichen.

Hinweise:

1. Die Tabellen neben den Zustandsgraphen enthalten die Zuordnung der Zustände zu Flipflops (Zustandscodierung).
2. Es geht hier lediglich darum, die Darstellungsweise zu zeigen. Es ist nicht notwendig, daß Sie versuchen, die Funktionsweise zu enträtseln.

Abbildung 4.11 DRAM-Ansteuerung: State Machine zur Bildung der RAS-Impulse (Intel)

Abbildung 4.12 DRAM-Ansteuerung: State Machine zur Bildung der CAS-Impulse im Page Mode (Intel)