ASYNCHRONE STATE-MACHINES

P. Balog

HTL und FhE am Technologischen Gewerbemuseum, Wien

ZUSAMMENFASSUNG:
In der asynchronen Schaltungstechnik ist der automatisierte, korrekte Entwurf von Ablaufsteuerungen von besonderer Bedeutung. Nachdem die Ziele definiert sind und die üblichen Entwurfsstile kurz aufgezählt wurden, liegt der Schwerpunkt dieses Beitrags auf dem *(Extended-)-Burst-Mode* Automaten und dessen Implementierung mit der 3D-Maschine.

Die Hauptanwendung von asynchronen Entwurfsmethoden sind im Bereich der Steuerungen (*Controller*, Automaten, etc.) zu finden. Kenneth Yi Yun befaßt sich in seiner Dissertation /1/ über asynchrone Steuerungen (*Controller*) in heterogenen Systemen allgemein mit diesem Thema und definiert als Ziel den Entwurf von korrekten und effizienten Steuereinheiten. Der Entwurfstil besteht aus 3 Methoden (Spezifikation, Impelementierung und Synthese), welcher sowohl die Qualität des fertigen Entwurfs als auch das Anwendungsgebiet selbst beeinflußt. Bei einem gegebenen Entwurfsstil für Steuerungen stellt Yun die beiden Fragen:

- *Can the signaling and timing constaints of most controllers be described in this design style ?*
- *Can every legal specification in this design style be implemented efficiently and correctly by automatic synthesis tools ?*

Für den Konstrukteur ist vor allem die zweite Frage von entscheidender Bedeutung hinsichtlich der Wahl einer Modellierungssprache. Viele dieser Modellierungssprachen

dienen primär der Beschreibung,- d.h. nicht jeder spezifizierbare Automat ist dann auch wirklich implementierbar.

Die Entwurfsstile für asynchrone Steuerungen bzw. Automaten der letzten Jahrzehnte lassen sich grob in drei Klassen einteilen:

1. Transformation von Spezifikationen basierend auf parallelen Hochsprachen (z.B. OCCAM) bzw. Formalismen (z.B. CSP) zumeist in *Delay-Insensitive*- oder *Speed-Independent*-Systeme /2/.
2. Synthese von graphentheoretischen Spezifikationen /2/ /3/ (z.B. Petri-Netze, *Signal Transition Graphs, State Graphs*, etc.).
3. *Multiple-Input-Change* AFSM (*Asynchronous Finite State Machine*) Synthese /1/ /2/ .

Während die Klassen (1) und (2) erfolgreich für kleine, extrem parallele Systeme angewandt werden können, ermöglichen die AFSMs einen Entwurf von komplexeren Ablaufsteuerungen sehr analog zu den synchronen FSMs. Der Großteil der asynchronen und synchronen FSMs basieren auf der Struktur der *Huffman-Mode*-AFSM (Abb.-1a), die jedoch heute wegen ihrer Einschränkungen (*single-input-change, fundamental mode*) keinerlei Bedeutung mehr hat.

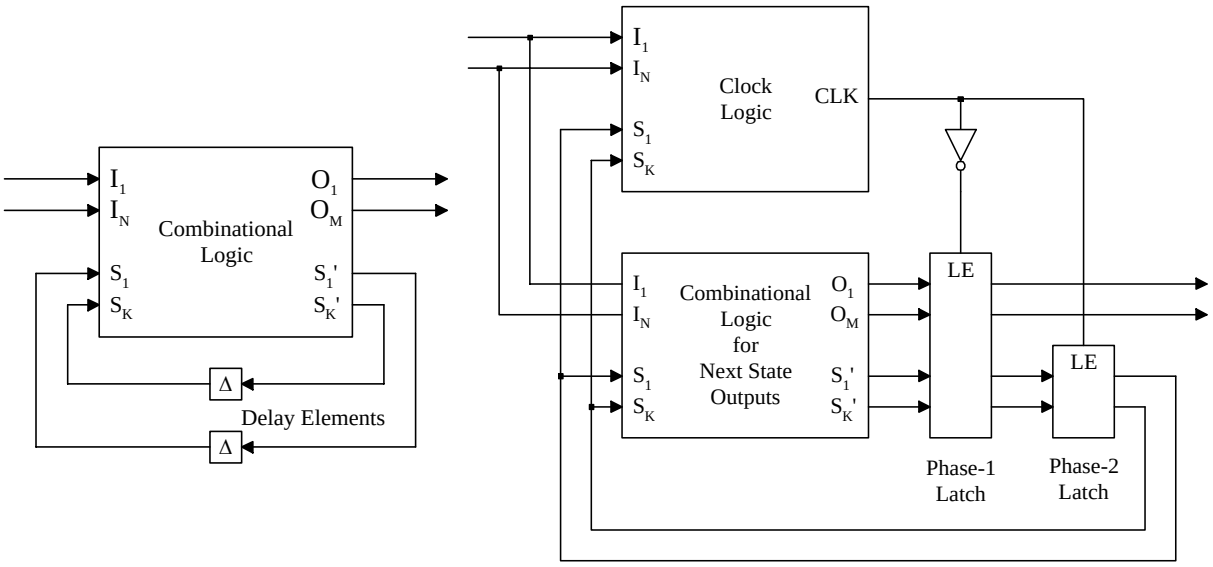

Abb. 1 a) Huffman Mode AFSM (links) b) Locally Clocked AFSM (rechts)

Eine Struktur von AFSMs, die von synchronen Automaten abgeleitet wurde, ist die *Locally-Clocked-Machine* (Abb.-1b), die erfolgreich als Implementierung für *Burst-Mode*- Spezifikationen verwendet werden kann /4/.

<u>*Burst-Mode* Spezifikation</u>

Der *Burst-Mode* Automat /1/ ist ein modifizierter *Mealy*-Automat, der *Multiple-Input-Change* in Form eines Eingangs-*Bursts* erlaubt. Für einen gegebenen Zustand wartet der Automat bis alle Flanken des Eingangs-*Bursts* angekommen sind, um dann die Ausgänge (*Output-Burst*) zu ändern und in einen neuen Zustand (*State-Burst*) zu wechseln. Die Flanken des Eingangs-*Bursts* dürfen in beliebiger Reihenfolge auftreten, was eine Parallelität auf der Eingangsseite garantiert.

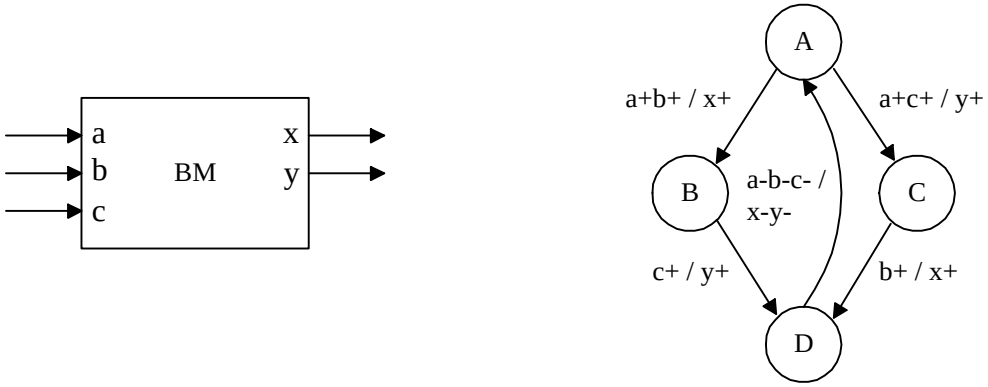

Abb. 2 Beispiel einer Burst-Mode Spezifikation (aus /3/ p75)

Ein *Burst-Mode* Automat wird mit einem Zustandsdiagramm (Abb.-2) beschrieben, welches aus Kreisen für Zustände und gerichteten Kanten für Zustandübergänge besteht. Die Zustandsübergänge werden mit *„Input-Burst / Output-Burst“* benannt. Für die Signalflanken gilt, daß s+ eine steigende und s- eine fallende Flanke beschreibt. Es gelten die folgenden Einschränkungen:

1. *Fundamental Mode*
2. *Subset Restriction*,- d.h. kein Übergang aus einem Zustand darf eine Untergruppe eines anderen Übergangs aus diesem Zustand sein. Z.B. der direkte Übergang von A nach D mit a+b+c+ / x+y+ ist unzulässig, da die anderen Übergänge a+b+ bzw. a+c+ eine Untermenge von a+b+c+ darstellen.
3. *Unique State Entry Restriction*,- d.h. jeder Übergang in einen bestimmten Zustand muß die selben *„Input-Burst / Output-Burst“* Definition haben.
4. Eingangssignale sind immer flankenrelevant (*transition signaling*),- Pegel sind nicht zulässig
5. Eingangs- und Ausgangsänderungen können nicht parallel zu einander erfolgen.

Extended Burst Mode

Die XBM-Spezifikation /1/ erweitert die BM-Spezifikation um die folgenden Eigenschaften, wobei die Restriktionen (4) und (5) eliminiert werden:

- *Directed-Don't-Cares* erlauben, daß sich Eingangssignale parallel zu Ausgangssignalen ändern können (*IO-concurrency*).
- *Conditionals* ermöglichen den Ablauf von Signalpegeln abhängig zu machen,- d.h. *Conditionals* sind pegelrelevante Qualifikationssignale für die flankenrelevanten Eingangs-*Burst*-Signale.

Das Beispiel in Abb. 3 zeigt einen Automaten mit drei flankenrelevanten Eingängen (ok, frin, dackn), einem *Conditional* Eingang (cntg1) und zwei Ausgängen (dreq, faout). Signale die nicht in spitzen Klammern <> stehen und mit + oder - enden sind sogenannte *Terminating Signals*, deren Flanken relevant sind. Signale in spitzen Klammern sind *Conditionals*, deren Pegel relevant sind. Ein pegelrelevanter Eingang (*Conditional*) darf nie als flankenrelevanter Eingang verwendet werden und umgekehrt. Ein Zustandsübergang erfolgt nur dann, wenn alle *Conditionals* erfüllt und alle *Terminating Signals* eingetroffen sind. Z.B. bedeutet die Übergangsdefinition <c+><d->x+ / y+z- : „ wenn c=1 und d=0 zu dem Zeitpunkt x eine steigende Flanke hat, dann schaltet y auf 1 und z auf 0 “.

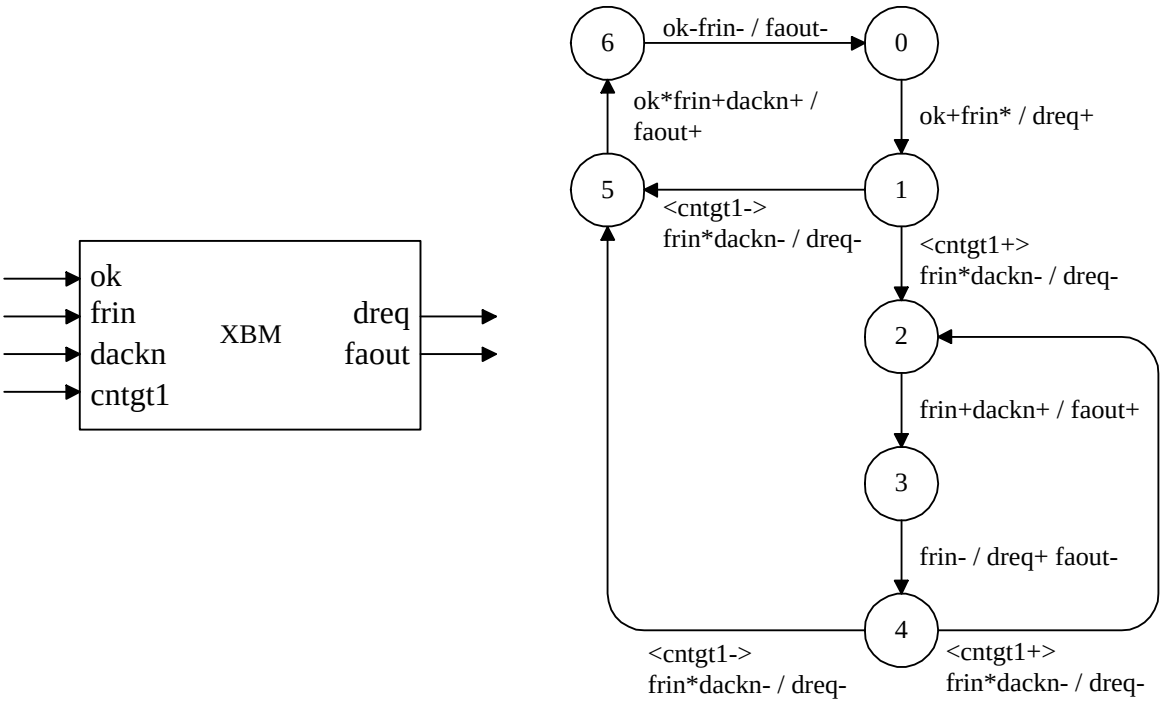

Abb. 3 Beispiel einer XBM-Spezifikation (aus /1/ p19)

Analog zu synchronen FSMs müssen die *Conditionals* gegenüber den *Terminating Signals* eine *Setup*- und *Hold*-Zeitrestriktion erfüllen,- d.h. alle *Conditionals* müssen gültig und stabil

sein vor der ersten Flanke eines *Terminating Signals* des Eingangs-*Bursts* (*setup*) und müssen gültig und stabil bleiben bis nach der Flanke des letzten *Terminating-Signals* des Eingangs-*Bursts* (*hold*). Außerhalb des *Setup-/Hold*-Zeitfensters, welches auch als *Sampling-Periode* bezeichnet wird, sind die *Conditionals* ohne Belang (*don't care*).

Ein Signal gefolgt von einem * ist ein *Directed-Don't-Care*, was bedeutet, daß sich das Signal ändern kann aber nicht ändern muß. Bei einem *Directed-Don't-Care* handelt es sich jedoch um ein monotones Signal im Unterschied zu einem „echten" *Don't Care*. Das folgende Beispiel verdeutlicht den Sachverhalt; die Sequenz a- a* a* a* a+ bedeutet, daß sich das Signal a nach a- auf 0 befindet. Im nächsten Zustand kann es auf 1 schalten oder aber 0 bleiben. Dasselbe gilt für die nächsten beiden Zustände. Spätestens im letzten Zustand muß das Signal a wegen a+ auf 1 schalten (*Terminating Signal*). Ist das Signal a jedoch während einem a* auf 1 gegangen, dann darf es nicht mehr auf 0 zurückschalten, d.h. das Signal darf nicht oszillieren.

Der synchrone *Moore*-Automat fällt auch in die Klasse von XBM-Automaten,- alle Eingänge des synchronen *Moore*-Automaten sind *Conditonals*, d.h. pegelrelevant, und der Takt ist das einzige flankenrelvante Signal welches permanent alterniert (*compulsory signal*).

3D-Maschine

Ein XBM-Automat (natürlich auch ein BM-Automat) kann mit einer 3D-Maschine /1/ implementiert werden. Die automatisierte Synthese für rückgekoppelte 2-stufige UND-ODER Strukturen (PLDs, CPLDs) benötigt keinerlei Speicherbauelemente, wie Register, Latches oder C-Elemente. Die Speicherung des momentanen Zustands erfolgt rein durch eine statische kombinatorische Rückkopplung.

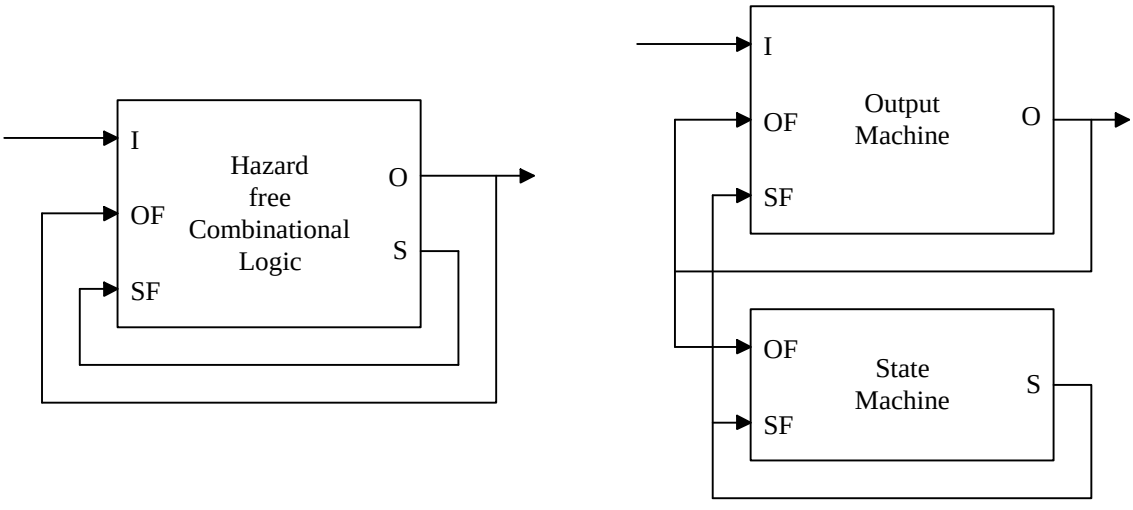

Abb. 4 a) Struktur der 3D-Maschine (links) b) Type II (rechts)

Eine neuere automatische Synthesetechnik für höhrere Verarbeitungsgeschwindigkeiten verwendet pseudostatische, asymmetrische CMOS-Gatter (gC - *Generalized C-Elements*) /5/ als explizite Speicherelemente.

Die prinzipielle Struktur entspricht der *Huffman-Mode*-AFSM, jedoch werden bei der 3D-Maschine primär die Ausgänge verwendet um den aktuellen Zustand zu speichern. Nur dann, wenn die Ausgänge nicht ausreichen eine eindeutige und hazardfreie Zustandscodierung zu erzielen, werden zusätzliche Zustandsbits eingeführt.

Die 3D-Maschine unterstützt 3 Arten von Maschinenzyklen:

- Typ I: Auf einen Eingangs-*Burst* folgt ein Ausgangs- / Zustands-*Burst* (parallel)
- Typ II: Auf einen Eingangs-*Burst* folgt ein Ausgangs-*Burst* auf den ein Zustands-*Burst* folgt (Abb.-4b).
- Typ III: Auf einen Eingangs-*Burst* folgt ein Zustands-*Burst* auf den ein Ausgangs-*Burst* folgt.

Die 3D-Implementierung aus der XBM-Beschreibung erfolgt über eine dreidimensionale Wahrheitstabelle der *Next-State-Function*, welche die {Eingänge, Ausgänge, Zustandsbits} in die {Ausgänge, Zustandsbits} abbildet. Das 3D-Synthesewerkzeug generiert die „funktional hazardfreie“ Wahrheitstabelle, aus der eine „logisch hazardfreie“ zweistufige UND-ODER Schaltung erzeugt wird. Dieses Schaltung kann dann entweder auf eine PLD- oder auf eine gCs basierende CMOS-Struktur abgebildet werden.

Literatur:

/1/ K. Y. Yun, „*Synthesis of Asynchronous Controllers for Heterogeneous Systems*“, PhD thesis, Stanford, 1994

/2/ S. Hauck, „*Asynchronous Design Methodologies: An Overview*“, Proceedings of the IEEE, Vol. 83, No. 1, Jannuary 1995, p69-p93

/3/ L. Lavagno, „*Synthesis and Testing of Bounded Wire Delay Asynchronous Circuits from Signal Transition Graphs*“, PhD thesis, UCB, 1992, p25-p106

/4/ S. M. Nowick, „*Automatic Synthesis of Burst-Mode Asynchronous Controllers*“, PhD thesis, Stanford University, 1993

/5/ K. Y. Yun, „*Automatic Synthesis of Extended-Burst-Mode Circuits Using Generalized C-Elements*“, Proceedings of the 1996 European Design Automation Conference, 1996, p290-p295