

Diplomarbeit

Abstrakte Datenflussmodelle für
GALS-Schaltungen zum Nachweis
nicht-funktionaler Eigenschaften

Alexandra Julius

14. Mai 2007



Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

Gutachter:
Prof. Dr. Wolfgang Reisig
Prof. Dr. Karsten Wolf

Zusammenfassung

In dieser Arbeit stellen wir Möglichkeiten vor, nicht-funktionale Eigenschaften einer GALS-Schaltung mittels Modellierung, Abstraktion und Verifikation zu analysieren. Im Mittelpunkt der Untersuchungen steht vor allem das Zeitverhalten und die Funktionssicherheit der Schaltung. Mit geeigneten Datenflussmodellen, in denen Zeit explizit modelliert ist, werden diese Eigenschaften untersucht. Da die Modellierung von Zeit die Komplexität des Modells erhöht und damit auch die Verifikation erschwert, modellieren wir den Datenfluss möglichst abstrakt. Wir schlagen in dieser Arbeit mehrere Abstraktionstechniken vor, die den Zustandsraum des Datenmodells reduzieren.

Inhaltsverzeichnis

1. Einleitung	6
2. Hintergrund	8
2.1. Global-asynchrone lokal-synchrone Schaltungen	8
2.2. WLAN-Basisbandprozessor	11
2.3. Überblick des Receivers	13
2.4. Verifikationsaufgaben	17
3. Formales Modell	21
3.1. Petrinetze und Zeitpetrinetze	21
3.2. Modell	30
3.3. Verifikationsaufgabe am Modell	43
4. Abstraktion des Modells	48
4.1. Abschätzung des Zustandsraums	48
4.2. Strukturelle Reduktion	50
4.3. Reduktion durch Verhaltensbeobachtung	59
4.4. Abstraktion der Anfangsmarkierung	69
4.5. Vergleich der Modelle	78
5. Zusammenfassung und Ausblick	81
5.1. Zusammenfassung	81
5.2. Ausblick	82
A. Glossar und Abkürzungen	84
Abbildungsverzeichnis	85
Tabellenverzeichnis	87
Literaturverzeichnis	88
Danksagung	90
Erklärung	91

1. Einleitung

Global-asynchronen lokal-synchrone (GALS-) Schaltkreise sind komplexe Systeme, die aus mehreren nebenläufigen Komponenten bestehen und als reaktive Systeme in hohem Maße mit ihrer Umgebung interagieren. Durch die hohe Komplexität dieser Schaltungen wirken sich Fehler meist auf große Teile des Systems aus, sind gleichzeitig jedoch schwer lokalisierbar. Insbesondere sind Analysen relevant, die nicht-funktionale Eigenschaften, wie Skalierbarkeit, Kosten, Betriebssicherheit bzw. -verlässlichkeit einer Schaltung untersuchen. Für den in dieser Arbeit behandelten Schaltkreis sind diese Eigenschaften bisher nur durch Tests untersucht worden. Allerdings können Testläufe nur die An-, nicht aber die Abwesenheit von Fehlern aufzeigen. Fehlerfreiheit kann letztlich nur durch systematisches Durchmustern *aller möglichen* Systemzustände sichergestellt werden [CES86].

Um eine derartige Analyse zu realisieren, bilden wir den GALS-Schaltkreis auf ein mathematisches Modell ab. Die zu überprüfenden Eigenschaften werden als gewünschte Eigenschaften des Modells formuliert. Ein Verifikationswerkzeug generiert und analysiert für dieses Modell den Zustandsraum. Die Größe des Zustandsraums ist abhängig von der Struktur des Modells, den Systemgrößen und der Ausgangssituation im Modell. Häufig ist es nicht möglich, den Zustandsraum des Systems vollständig zu berechnen. Um die erschöpfende Analyse aller möglichen Systemzustände zu gewährleisten, stellen wir in dieser Arbeit mehrere Abstraktionsmethoden für das Modell der Schaltung vor. In diesem Zusammenhang weisen wir nach, dass die zu untersuchenden Eigenschaften im abstrakten Modell bewahrt sind und beschreiben so einen Ansatz, die Analyse von komplexen Systemen zu realisieren.

Wir betrachten in dieser Arbeit den in [KG04] und [KGSP06] vorgestellten GALS-Schaltkreis. Dieser Schaltkreis wurde in [KGS05] bereits implementiert und getestet und ist in [SRK05] auf funktionale Eigenschaften überprüft worden. Da im Mittelpunkt unserer Untersuchungen die nicht-funktionalen Eigenschaften der GALS-Schaltung stehen, modellieren wir das System nicht wie in [SRK05] mit Stellen-Transitions-Netzen: Um Aussagen über das Zeitverhalten des Schaltkreises treffen zu können, wählen wir einen Formalismus, der die Größe Zeit berücksichtigt. Auf Grund des hohen Grads an Nebenläufigkeit in der GALS-Schaltung, wählen wir Zeit-Petrinetze als Formalismus des zu untersuchenden Modells. Anschließend diskutieren wir mehrere Reduktions- und Ab-

straktionsmöglichkeiten, die die Analyse des Modells mit Verifikationswerkzeugen erst ermöglichen.

Dazu stellen wir in Kapitel 2 zunächst grundlegenden Aufbau und Funktionsweise von GALS-Schaltungen vor und spezifizieren den zu untersuchenden Schaltkreis: den Receiver des WLAN-Basisbandprozessors. Weiterhin beschreiben wir die nicht-funktionalen Eigenschaften, die für den Receiver zu untersuchen sind. Wir zählen vier relevante Fragen auf, die für den korrekten Betrieb der Schaltung von entscheidender Bedeutung sind. Im Mittelpunkt der Untersuchungen stehen hier insbesondere das Zeitverhalten der Schaltung, mögliche Latenzen im Datendurchsatz, Speichertiefen der Pipelines und Toleranzen gegenüber der Änderung einzelner Taktraten.

In Kapitel 3 geben wir zunächst eine Einführung in die Petrinetztheorie und definieren den von uns gewählten Formalismus. Anschließend stellen wir das von uns erstellte Modell des Receivers vor und erläutern die Angemessenheit der Modellierung. Des Weiteren überführen wir die technischen Fragestellungen aus Kapitel 2 in zu überprüfende Modelleigenschaften und zeigen Analysemöglichkeiten auf. In diesem Zusammenhang führen wir die ursprünglichen Fragen auf Erreichbarkeitseigenschaften eines Petrinetzes zurück.

Anschließend diskutieren wir in Kapitel 4 mehrere Möglichkeiten, das vorliegende Modell zu reduzieren. Dazu spezifizieren wir zunächst strukturelle Reduktionregeln für Stellen-Transitions-Netze und passen diese an die Semantik von Intervall-Petrinetzen an. Des Weiteren untersuchen wir das regelmäßige Verhalten im Modell und fassen Teile des Modells so zusammen, dass für die Analyse weniger Zustände zu betrachten sind. Weiterhin stellen wir eine Abstraktionsfunktion vor, mit deren Hilfe wir die Anfangsmarkierung des Modells verkleinern. Wir weisen für diese Abstraktionsfunktion nach, dass durch die Abstraktion des Netzes die zu untersuchende Erreichbarkeitseigenschaft im abstrakten Modell bewahrt bleibt.

Zuletzt fassen wir in Kapitel 5 die Resultate der Arbeit zusammen, beschreiben die offen gebliebenen Probleme der Verifikation des Zustandsraums und diskutieren mögliche Lösungswege.

2. Hintergrund

Wir erklären in Kapitel 2.1 zunächst die grundlegenden Prinzipien und Eigenschaften einer global-asynchronen lokal-synchronen (GALS-) Schaltung und gehen auf deren Funktionsweise ein. Anschließend stellen wir in Kapitel 2.2 eine Anwendung des beschriebenen GALS-Schaltkreises vor, einen WLAN-Basisbandprozessor. Der Daten- und Kontrollfluss dieser GALS-Schaltung sind Gegenstand der Untersuchungen in dieser Arbeit. In Kapitel 2.3 gehen wir auf die eigentliche Aufgabenstellung ein und zählen Fragen an den Schaltkreis auf, die durch Testen bisher nicht beantwortet werden konnten.

2.1. Global-asynchrone lokal-synchrone Schaltungen

2.1.1. GALS-Prinzip

Global-asynchrone lokal-synchrone Schaltungen (GALS-Schaltungen¹) sind ein neuer Ansatz im Entwurf von VLSI-Systemen (Very Large Scale Integrated Systems) und versprechen, die Vorteile des synchronen und asynchronen Schaltkreisentwurfs zu kombinieren und dabei deren Nachteile zu umgehen. Erstmals vorgestellt wurde dieser Ansatz von Chapiro [Cha84]. Eine GALS-Schaltung besteht aus mehreren lokal-synchronen Modulen (LS-Module). Jedes LS-Modul arbeitet in sich synchron und ist zugleich in einen Wrapper eingebettet, der eine asynchrone Schnittstelle zu anderen Modulen anbietet. Ein Wrapper und sein eingebettetes LS-Modul bilden einen GALS-Block. Die Kommunikation zwischen GALS-Blöcken ist asynchron und geschieht mit bekannten Handshake-Protokollen. Die asynchronen Wrapper sind spezielle Schaltungen, die getaktete und asynchrone Berechnungen miteinander koordinieren. Die Wrapper einer GALS-Schaltung sind durch Pipelines miteinander verbunden. Jeder Wrapper stellt dem jeweiligen LS-Modul unter anderem einen lokalen pausierbaren Taktgeber zur Verfügung, der beliebig kalibriert werden kann. Dies ermöglicht eine an die jeweilige Aufgabe des LS-Moduls angepasste, individuelle Frequenz, mit der ein Block arbeitet. Die Vorteile dieses heterochronen GALS-Schaltkreises sind die Reduktion der elektromagnetischen Interferenzen gegenüber

¹Die in dieser Arbeit aufgeführten Abkürzungen sind im Index zu finden.

einer äquivalenten synchronen Schaltung und die Umgehung von Taktverteilungsproblemen des synchronen Schaltungsentwurfs.

2.1.2. GALS-Block

Wir betrachten in dieser Arbeit die in [KG04] vorgeschlagene GALS-Technologie, die auf einem Request-gesteuerten Betrieb der LS-Module basiert. Die Idee dieses Konzepts ist, dass ein LS-Modul ein eingehendes Handshake-Signal während des Datenempfangs als Taktsignal benutzt. Eine spezielle Time-out Schaltung ermittelt den Moment, in dem das Handshake-Signal nicht mehr aktiv ist. Wenn über einen bestimmten Zeitraum keine weiteren Handshakes auftreten, wird die Erzeugung des Taktsignals für das LS-Modul vom Taktgeber des Wrappers übernommen. Dies dient dem Leeren interner Pipeline-Stufen des LS-Moduls. Die lokale Uhr eines GALS-Blocks bleibt aktiv, solange Daten aufgenommen, verarbeitet und ausgegeben werden. Ansonsten befindet sich der Block im Ruhezustand. Daraus ergibt sich ein deutlich effizienterer Energieverbrauch für die gesamte Schaltung aus [KG04] und [KGSP06].

Im Folgenden erklären wir kurz den Aufbau und die Struktur des GALS-Blocks aus Abb. 2.1. Um das Request-gesteuerte Prinzip zu realisieren, ist es nötig, spezielle Kontrollstrukturen in den Wrapper einzubinden. Dieser besteht aus den fünf Komponenten Input Port, Output Port, Pausable Clock, Time-out Generator, Clock Control und Pausable Clock. Die Komponenten Input Port und Output Port regeln die Handshakes zur Umgebung. Der lokale Taktgeber ist in die Pausable Clock eingebettet. Die Komponenten Time-out Generator, Clock Control und Pausable Clock unterstützen die Aktivierung eines GALS-Blocks bei der Datenaufnahme und seine Deaktivierung, nachdem keine Daten mehr zu bearbeiten sind.

Wir unterscheiden zwischen vier verschiedene Phasen, in denen sich der GALS-Block aus Abb. 2.1 befinden kann: der *Ruhezustand*, die *Request-gesteuerte Phase*, die *Time-Out Phase* und die *Local-Clock Phase*. Die Übergänge der einzelnen Phasen sind in Abb. 2.2 dargestellt. Im Folgenden gehen wir auf diese Phasen kurz ein.

Befindet sich ein GALS-Block im Ruhezustand, so ist weder das lokale Taktsignal (*LCLK* in Abb. 2.1) aktiviert noch findet Datentransfer statt. Erreicht den Block eine Handshake-Anfrage aus der Umgebung (REQ_A+ ² in Abb. 2.1) geht der Schaltkreis in die Request-gesteuerte Phase über, in der das LS-Modul Daten aus der Umgebung aufnimmt und dabei das Taktsignal von den eingehenden Request-Signalen bezieht. Jedes Datum wird im

² REQ_A+ steht hier für einen hohen Signalpegel des Request-Signals von Block A, also $REQ_A = 1$. Analog bildet REQ_A- einen niedrigen Signalpegel ab, also $REQ_A = 0$.

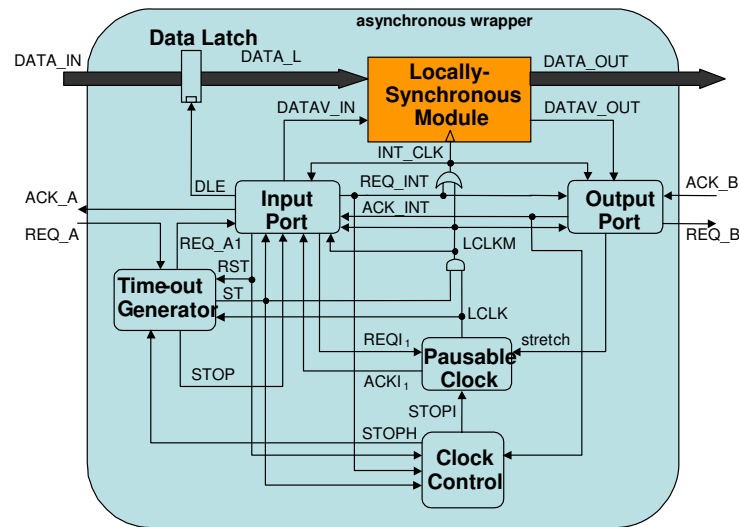


Abbildung 2.1.: Aufbau und Struktur eines GALS-Blocks.

LS-Modul mittels des korrespondierenden Request-Signals, welches dem Modul verzögert zugeführt wird, verarbeitet.

Der Taktgeber des Wrappers wird aktiviert, sobald das erste Request-Signal am Input Port anliegt. Das resultierende lokale Taktsignal wird dem Time-out Generator zugeführt. Diese Komponente ermittelt den Zeitpunkt, an dem die Datenaufnahme im GALS-Block beendet ist und die Time-out Phase beginnt. Vergehen vier Taktzyklen ohne eingehendes Request, so sind alle Daten aus der Umgebung aufgenommen und bearbeitet worden. Diese Zeit wurde von den Entwicklern auf Grund von Test- und Simulationserfahrungen als Dauer der Time-out Phase bestimmt. Während dieser Phase gibt der Time-out Generator $ST+$ aus, ein Kontrollsignal, das das lokale Taktsignal für das LS-Modul freigibt und die Local-Clock Phase einleitet. Die Time-out Phase dauert für jeden Block vier Taktzyklen des korrespondierenden Taktgebers an. Da in jedem Wrapper das lokale Taktsignal jedoch diverse Gatter und Flip Flops auf dem Weg ins LS-Modul passiert, nehmen wir vier bis sechs Taktzyklen für die Dauer der Time-out Phase an.

Während der anschließenden Local-Clock Phase werden die verarbeiteten Daten durch die Pipeline zum nachfolgenden GALS-Block transportiert. Das LS-Modul wird währenddessen ausschließlich von der Pausable Clock mit einem Taktsignal versorgt. Die Komponente Clock Control beendet diesen Vorgang, sobald die Pipeline geleert ist, indem sie

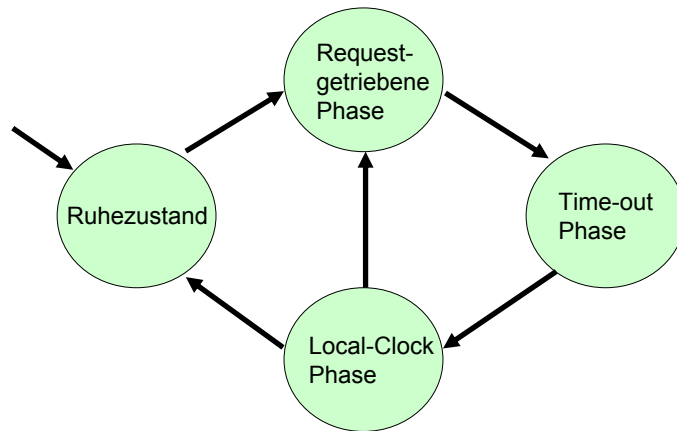


Abbildung 2.2.: Übergänge der Phasen eines Wrappers.

den Taktgeber stoppt und alle Wrapper-Komponenten zurücksetzt. Der Wrapper kehrt in den Ruhezustand zurück und ist bereit, neue Request-Signale zu empfangen.

Sendet die Umgebung während der Local-Clock Phase erneut eine Handshake-Anfrage, unterbricht der GALS-Block die Datenausgabe nach Abschluss des aktuellen Handshakes und geht wieder in die Request-gesteuerte Phase über, um die Daten aus der Umgebung aufzunehmen. Danach wird wieder - wie oben beschrieben - die Time-out und Local-Clock Phase aufgenommen.

Die GALS-Blöcke, die den zu untersuchenden WLAN-Basisbandprozessor implementieren, sind teilweise abgeänderte Varianten des hier beschriebenen Wrappers. Unter anderem arbeiten einige der Wrapper die Datenaufnahme und -abgabe und die Berechnungen der Daten während der Local-Clock Phase nebenläufig ab. Im Folgenden werden wir genauer auf die Besonderheiten, der einzelnen Wrapper eingehen und diese bezüglich ihres Verhaltens während der Request-gesteuerten, Time-out und Local-Clock Phase charakterisieren.

2.2. WLAN-Basisbandprozessor

Der in dieser Arbeit zu untersuchende GALS-Schaltkreis ist ein 802.11.a WLAN-Basisbandprozessor, der in [KG04] und [KGSP06] vorgestellt ist. Der Basisbandprozessor besteht aus den zwei Komponenten Receiver und Transmitter. Beide Komponenten implementieren mehrere verschiedene GALS-Blöcke. Der Receiver führt die für den WLAN-

Verkehr notwendige Synchronisation, Demodulation und Entschlüsselung der Daten durch, der Transmitter deren Verschlüsselung, Kodierung und Modulation. Beide Komponenten sind eingebettet in eine Signalverarbeitungskette die in Abb. 2.3 veranschaulicht ist.

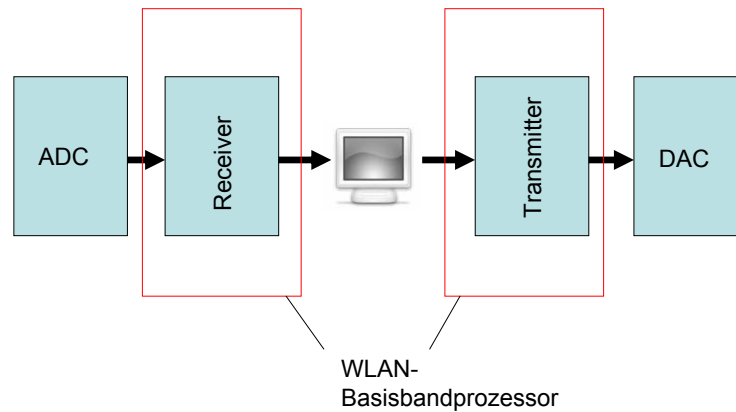


Abbildung 2.3.: Der Basisbandprozessor empfängt digitalisierte Daten aus dem Analog Digital Converter (ADC). Daten, die der Schaltkreis sendet, werden vom Digital Analog Converter analogisiert (DAC).

Um WLAN-Signale zu senden und zu empfangen, sind komplexe Berechnungen notwendig, die auf die einzelnen GALS-Wrapper des Transmitters und Receivers verteilt sind. Zum drahtlosen Senden von Informationen werden im Transmitter die zu übertragenden Daten auf ein Trägersignal moduliert. Das Ergebnis dieser Berechnung ist ein Datenpaket, das aus einer *Präambel* und *Nutzdaten* besteht. Dieses Datenpaket nennen wir *Symbol*. Die Präambel enthält die Übertragungsparameter der Nutzdaten (z.B. Frequenz, Phase, Amplitude, Abtastrate). Ein Symbol besteht aus 48 *Token*. Token sind die kleinste Dateneinheit im Schaltkreis. Ein Token entspricht einem Bitvektor.

Ein Datenpaket, das der Basisbandprozessor empfängt, durchläuft im Receiver zunächst mehrere Synchronisationsstufen. Das synchronisierte Datenpaket wird anschließend demoduliert: Mit Hilfe der Übertragungsparameter aus der Präambel werden die zu übertragenden Daten aus den Nutzdaten separiert. Aufgrund der mobilen Anwendung des Basisbandprozessors, ist es möglich, dass die Präambel nicht korrekt übertragen wird, da sich die Frequenz des Trägersignals während des Betriebes ändern kann. Um jedes eingehende Datenpaket trotzdem demodulieren zu können, wird im Receiver eine Kanalschätzung durchgeführt. Dies bedeutet, dass zur Demodulation des Symbols $i + D$ die Demodulation des Symbols i notwendig ist, wobei D die Verzögerung darstellt, die entsteht, während Daten zurückgeführt werden. Zur Realisierung dieser Berechnungen

implementiert der Receiver eine Rückführung, die es ermöglicht, neu empfangene Symbole mit vorherigen zu vergleichen.

In dieser Arbeit beschränken wir uns auf die Analyse des Zeitverhaltens des Receivers. Daher abstrahieren wir von den einzelnen Signalverarbeitungsprozessen und modellieren in Kapitel 3 lediglich Daten- und Kontrollfluss der GALS-Blöcke, die den Receiver implementieren. Der Transmitter ist nicht Gegenstand der Untersuchungen in dieser Arbeit.

2.3. Überblick des Receivers

Der Receiver realisiert im WLAN-Basisbandprozessor die Synchronisation und Demodulation der empfangenen Daten. Zu diesem Zweck führt der Schaltkreis Berechnungen zur Datensynchronisation, Kanalschätzung und Angleichung der Daten aus. Der Receiver ist unterteilt in fünf wesentliche Komponenten, von denen vier als GALS-Blöcke implementiert sind. Wir werden sie im Folgenden mit B1, B2, B3 und B4 bezeichnen. Mit M1 bis M4 bezeichnen wir die jeweiligen LS-Module, mit W1 bis W4 deren jeweilige Wrapper. Die fünfte Komponente ist die *Asynchrone Queue*, durch die die Daten von B4 zu B2 zurückgeführt werden. In Abb. 2.4 ist eine stark vereinfachte Darstellung des Receivers zu sehen. Mit *Env* bezeichnen wir die Umgebung, die Daten an den Receiver abgibt. Die Pfeile bilden hier den Datenfluss ab, die Inschriften die Datendurchsatzrate (MSPs, siehe Index). Die Eingangsdaten des Receivers sind digitalisierte Werte des Eingangssignals. Der generelle Ablauf gestaltet sich derart, dass B1 und B2 zunächst die Gültigkeit der Daten prüfen und Berechnungen zur Datensynchronisation ausführen. Nachdem B1 den Beginn der Nutzdaten ermittelt hat, synchronisiert B2 die entsprechenden Nutzdaten und führt die Kanalschätzung durch. Anschließend werden die Symbole in B3 von 20 auf 80 MSPs beschleunigt. In B4 werden die Nutzdaten, mittels der Ergebnisse der Kanalschätzung endgültig entschlüsselt. An dieser Stelle wird außerdem der Datenstrom zweigeteilt, der Receiver gibt die dekodierten Daten aus und führt sie zum Zwecke der Kanalschätzung in die *Asynchrone Queue* zurück.

Der Schaltkreis empfängt und sendet Daten in Form von *Frames*. Die Länge eines Frames ist durch die Anzahl seiner Symbole definiert. Diese ist variabel und kann bis zu 1300 Symbole betragen. Der Receiver ist aktiv, solange ein Frame aufgenommen, verarbeitet und ausgegeben wird. Sind alle empfangenen Symbole eines Frames entschlüsselt, wird der gesamte Schaltkreis zurückgesetzt. Es ist darum im Folgenden lediglich nötig, das Verhalten des Receivers während der Verarbeitung eines Frames zu untersuchen.

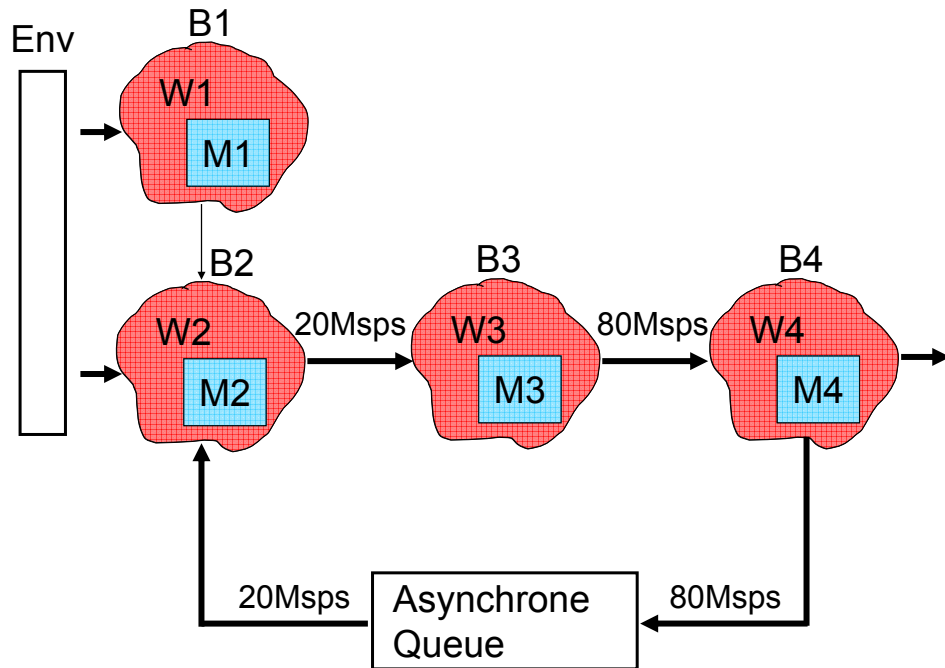


Abbildung 2.4.: Datenfluss im Receiver in Mega Samples per Second (Msp/s).

Umgebung

Die Umgebung kann jederzeit Daten an den Receiver abgeben. Bevor die Daten den Receiver erreichen, werden sie gepuffert. Dadurch werden Unregelmäßigkeiten im Datenstrom kompensiert, so dass der Schaltkreis toleranter gegenüber Verzögerungen am Eingang ist. Selbst wenn kurzfristig keine Daten anliegen, erreichen Token den Receiver gleichmäßig im 20 MHz-Takt. Die Umgebung gibt kontinuierlich Symbole an die Blöcke B1 und B2 ab, bis ein Frame vollständig abgearbeitet ist.

Block B1

B1 ist ein GALS-Block, der für die Synchronisation der eingehenden Daten verantwortlich ist und mit einer Frequenz von 20 MHz arbeitet. Mittels Autokorrelation berechnet M1 den Anfang der Nutzdaten im Bitstrom und signalisiert B2, ab welchem Zeitpunkt Nutzdaten empfangen werden. Anschließend geht B1 in den Ruhezustand über. Nachdem

B2 ein Datenpaket aufgenommen hat, liegt das nächste Symbol des Frames am Eingang des Receivers an und B1 wird wieder in Betrieb gesetzt.

Block B2

B2 ist ein GALS-Block, der die Aufnahme, Berechnung und Abgabe von Daten nebenläufig und mit einer Taktrate von 20 MHz vollzieht. Dieser Wrapper unterbricht seine Berechnungen nur, wenn von der Umgebung oder der *Asynchronen Queue* keine Daten mehr bereitgestellt werden. Während der Verarbeitung eines Frames ist B2 ständig aktiv und vereinigt den kontinuierlichen Datenstrom aus der Umgebung und aus der Rückführung. B2 bestimmt für jedes Symbol, mit welcher Frequenz das empfangene Signal abgetastet werden muss, um es korrekt rekonstruieren zu können. Unter anderem analysiert M2 das Synchronisationsmuster der Nutzdaten eines jeden Symbols.

Auf Grund der notwendigen Kanalschätzung muss jedes Symbol im Receiver zurückgeführt werden. Initial nimmt B2 lediglich Symbole aus der Umgebung auf und gibt sie an B3 weiter. Auf diesem Weg gelangt jedes Symbol über B4 und die *Asynchrone Queue* nach einiger Zeit wieder bei B2 an. Sobald das erste Symbol bis in die *Asynchrone Queue* propagiert wurde, berechnet B2 Daten aus der Umgebung nur, wenn gleichzeitig ein Symbol in der *Asynchronen Queue* bereitliegt. Ist diese Bedingung erfüllt, generiert B2 aus einem Symbol aus der Umgebung *und* einem Symbol aus der Rückführung ein neues Symbol und gibt es an B3 weiter. Der Block verfügt zusätzlich über interne Pipelines, die es ermöglichen, Symbole aus der Umgebung und der *Asynchronen Queue* zu empfangen, während bereits berechnete Symbole ausgegeben werden.

Block B3

B3 ist ein GALS-Block, der die in Kapitel 2.1.2 beschriebenen Phasen befolgt und dabei von B2 unterbrochen werden kann. Zudem schließen sich Datenaufnahme und -abgabe in B3 gegenseitig aus, während in den anderen Blöcken des Receivers Daten nebenläufig aufgenommen und ausgegeben werden. Das von B3 eingebettete LS-Modul M3 implementiert eine synchrone Queue mit 48 Stufen. Pro Stufe speichert die Struktur ein Token, insgesamt also ein vollständiges Symbol. Während der Request-getriebenen Phase nimmt B3 ein Symbol mit 20 MHz auf, da B2 die Daten mit dieser Rate an B3 übermittelt. Der in B3 eingebettete lokale Taktgeber ist auf eine Frequenz von 80 MHz eingestellt. Demzufolge übermittelt B3 ein Symbol während der Local-Clock Phase mit 80 MHz an B4.

Block B4

B4 vollzieht komplexe Berechnungen, die die Ergebnisse der Kanalschätzung umsetzen und die Nutzdaten endgültig entschlüsseln. Dabei finden in B4 Aufnahme, Berechnung und Abgabe von Symbolen nebenläufig und mit einer hohen Taktrate von 80 MHz statt. Der Betrieb von B4 wird unterbrochen, wenn es für den Vorgänger B3 nicht mehr möglich ist, Daten an B4 weiterzugeben bzw. wenn die *Asynchrone Queue* keine Daten von B4 entgegennimmt. Ansonsten ist B4 während der Verarbeitung eines Frames stets aktiv. Zusätzlich implementiert dieser Schaltkreis tiefe Pipelines, die mehrere Symbole abspeichern. Sobald B4 Daten empfängt, muss zunächst ein Symbol vollständig aufgenommen und in der Pipeline abgelegt werden. Symbol i wird von B4 erst bearbeitet, wenn das nachfolgende Symbol $i+1$ vollständig aufgenommen wurde. Nach Aufnahme des $i+1$ -ten Symbols beginnt B4 mit der Berechnung von Symbol i .

Zusätzlich wird der Ausgabestrom in B4 zweigeteilt. Ein berechnetes Symbol wird zum Ausgang des Receivers transportiert und gleichzeitig in die *Asynchrone Queue* zurückgeführt. Zur selben Zeit finden in B4 Aufnahme und Berechnung der nächsten Symbole des Frames statt. Der Block B4 gibt das Symbol i erst aus, nachdem er das Symbol $i+2$ aufgenommen hat. Die einzige Ausnahme bildet das letzte Symbol eines Frames, das direkt verarbeitet wird.

Asynchrone Queue

Die *Asynchrone Queue* basiert auf dem FIFO-Konzept und besteht aus 48 Stufen. Ihre Architektur ist an [FD96] angelehnt. Pro Stufe speichert die Struktur ein Token, insgesamt also ein vollständiges Symbol. Diese Struktur ist kein GALS-Block, sie enthält weder einen Wrapper noch ein LS-Modul. B4 übermittelt die Daten an die *Asynchrone Queue* mit einer Taktrate von 80 MHz. Während des Datentransfers innerhalb der *Asynchronen Queue* werden Handshakes zwischen benachbarten Pipelineinstufen vollzogen. Dabei ergibt sich eine ungefähre Verzögerung von 3 bis 5 ns pro Stufe. Die genaue Latenz des Rückwärtsflusses ist jedoch unbekannt, da nicht vorhersagbar ist, welche Stufen frei sind, während ein Token in die *Asynchronen Queue* aufgenommen wird. Der Rückwärtsfluss der Daten im Receiver erfolgt asynchron, das heißt, Daten werden unabhängig zueinander ein- und ausgegeben. Die Verzögerung, die die Queue im Datenfluss darstellt, kann je nach Füllstand entweder größer oder kleiner sein. Wenn zwei benachbarte Stufen der *Asynchronen Queue* Daten speichern, muss die zweite Stufe zunächst ihre Daten in Richtung Ausgang der *Asynchronen Queue* weitergeben, bevor die erste Stufe Daten zum Ausgang propagiert. Die Ausgabe der Daten erfolgt über Handshakes mit B2. Da dieser

GALS-Block mit einer Rate von 20 MHz arbeitet, verlangsamt sich der Datenfluss an dieser Stelle von 80 auf 20 MHz.

2.4. Verifikationsaufgaben

Das Zusammenspiel der einzelnen GALS-Blöcke im Receiver obliegt bestimmten zeitlichen Einschränkungen, die eingehalten werden müssen, um die korrekte Funktion der Schaltung zu gewährleisten. Um diese zeitlichen Einschränkungen abzuschätzen und die Schaltung zu testen, wurde ein VHDL-Modell (VLSI Circuit Hardware Description Language)[VHD87] des Basisbandprozessors erstellt. Bei diesem Prozess wird die Schaltung in Software überführt. Das VHDL-Modell kann durch Simulation automatisch getestet werden ([KGS05], [Jul05]). Dies ermöglicht den Entwicklern, mögliche Frequenzen und Verzögerungen für die Schaltung abzuschätzen. Es kann allerdings für diese Angaben nicht garantiert werden, dass sie adäquat gewählt sind. Generell können die Entwickler wichtige nicht-funktionale Eigenschaften der Schaltung, wie Latenz, Datendurchsatz, Toleranzgrenzen bezüglich entstehender Verzögerungen und Flächenverbrauch, durch reines Testen nicht zusichern. Wegen der hohen Komplexität des VHDL-Modells, ist es außerdem nicht möglich, dieses computergestützt zu verifizieren. Darum erstellen wir in Kapitel 3 ein vereinfachtes Modell des Receivers, das auf die nachfolgenden Fragestellungen zugeschnitten ist. Im Folgenden formulieren wir vier Fragen, die für den korrekten Betrieb der Schaltung von entscheidender Bedeutung sind.

1. Ist es möglich, dass der Receiver nach einer gewissen Initialisierungsphase Daten konstant mit 80 MHz zur Verfügung stellt, das heißt B4 Daten mit 80 MHz ausgibt?

Im Allgemeinen ist unbekannt, ob der Receiver unter den gegebenen Bedingungen jederzeit korrekt arbeitet. Insbesondere stellt sich die Frage, ob es möglich ist, dass alle GALS-Blöcke stets rechtzeitig die benötigten Daten zu Verfügung stellen. Für jeden GALS-Block und die Asynchrone Queue ist eine, für den Datenfluss mögliche, minimale und maximale Verzögerung bekannt. Jedoch ist nicht klar, ob der Schaltkreis für alle möglichen Reaktionszeiten einer jeden Komponente korrekt arbeitet. Reagiert ein Block sehr schnell und ein anderer sehr langsam und summieren sich während des Betriebs Verzögerungen derart auf, dass Unterbrechung im Datenfluss entstehen, resultiert daraus der kurzzeitige Ausfall der Datenübertragung. Grund für Zweifel an der Korrektheit gibt insbesondere die Vereinigung des Datenflusses am Eingang von B2. Dieser Block kann neue Daten aus der Umgebung nur aufnehmen, wenn die Vorherigen in der Rückführung be-

reit liegen. Ansonsten unterbricht die Datenübertragung bzw. die Datenübermittlung am Ausgang von **B4**. Da jedoch der Daten- und Kontrollfluss des Blocks **B3** unterbrochen werden kann und zudem die Latenz einer jeden Komponente des Receivers wegen minimaler und maximaler Reaktionszeiten variiert, sind Vorhersagen über die Korrektheit der Datenübertragung durch reines Testen sehr schwer zu beantworten. Um zu untersuchen, ob der Schaltkreis Verzögerungen gegenüber tolerant ist, betrachten wir im Modell den Ausgang des Blocks **B4**.

Block **B4** transportiert ab einem bestimmten Zeitpunkt (also einige Zeit nachdem dieser das dritte Symbol eines Frames aufgenommen hat) Token mit einer Frequenz von 80 MHz an die *Asynchrone Queue*. Ist die Übertragungsrate des Schaltkreises korrekt, nimmt die *Asynchrone Queue* dann alle 12,5 ns ein Token auf. Sobald das erste Symbol gespeichert ist, muss es für **B2** möglich sein, alle 50 ns ein Token am Ausgang der *Asynchronen Queue* entgegenzunehmen, dies entspricht einer Frequenz von 20 MHz. Ist es ab diesem Zeitpunkt für **B2** nicht möglich, während der Aufnahme eines externen Symbols (innerhalb von 4000 Nanosekunden), 2400 Nanosekunden lang ein internes Symbol von der *Asynchronen Queue* entgegenzunehmen, so kann das externe Symbol in **B2** nicht berechnet werden und die Datenübertragung des Receivers ist unterbrochen.

Konkret stellen wir uns die Frage, ob es für **B4** immer möglich ist, Daten mit einer Rate von 80 MHz abzugeben, nachdem eine gewisse Initialisierungsphase verstrichen ist, in der die ersten Symbole eines Frames aufgenommen werden.

2. Ist es möglich, den Receiver mit einer niedrigeren Taktrate zu betreiben?

Der gegebene WLAN-Basisbandprozessor ist vor allem für mobile Anwendungen konzipiert. Wichtig ist dabei, dass der Energiebedarf möglichst gering gehalten wird. Die meiste Energie verbrauchen die lokalen Taktgeber in den *Pausable-Clocks* der Blöcke **B3** und **B4**, da diese mit 80 MHz betrieben werden. Mittels Simulation der Schaltung stellten die Entwickler mögliche Toleranzen für die Taktrate fest. Ziel der Untersuchungen in dieser Arbeit ist es zu zeigen, ob der Schaltkreis mit niedrigerer Taktrate noch immer korrekt funktioniert. Konkret stellt sich die Frage, wie stark die Frequenz der lokalen Uhren in **B3** und **B4** verringert werden darf, so dass gleichzeitig das Datenübertragungsverhalten des Receivers bewahrt ist.

3. Ist der Receiver tolerant gegenüber Verzögerungen des Datenflusses in der Asynchronen Queue?

Gegenstand weiterer Untersuchungen ist die Asynchrone Queue, die die Rückführung der Daten von B4 nach B2 realisiert und ein Symbol speichert. Der Datenfluss findet in dieser Struktur unabhängig und asynchron statt. Zwischen den Stufen der Queue, sowie am Eingang und Ausgang, findet für jedes Token ein Handshake statt. Ist die Asynchrone Queue initial leer, werden die Token schnell zum Ausgang der Struktur transportiert, so dass die korrekte Datenübertragung gesichert ist. Es gibt jedoch Szenarien, in denen ein Token eine größere Verzögerung hat. Liegen in der Asynchronen Queue mehrere Token willkürlich verteilt, kann die Latenz insgesamt so groß sein, dass die Struktur letztendlich Token am Eingang nicht mehr entgegennimmt. Innerhalb des Schaltkreises gehen an dieser Stelle keine Informationen verloren, da die Token mittels Handshakes übertragen werden und B4 kein Symbol weitergibt, bevor das vorherige nicht entgegengenommen wurden. Am Ausgang der Queue können dann allerdings die Daten nicht mehr rechtzeitig an B2 übermittelt werden. Diese Verzögerung hätte den Ausfall der Datenübertragung zur Folge, da B2 Daten aus der Umgebung nur entschlüsseln kann, wenn die vorherigen Daten in der Rückführung bereitliegen. Es stellt sich die Frage, ob die Latenz der Asynchronen Queue jederzeit niedrig genug ist, um das korrekte Verhalten der gesamten Schaltung zu gewährleisten. Wir untersuchen diese Frage am Ein- und Ausgang der Asynchronen Queue und ermitteln ob es möglich ist, dass die Asynchrone Queue alle 4000 ns ein Symbol aufnehmen bzw. ausgeben kann.

4. Ist es möglich, den Receiver zu betreiben, wenn die Asynchrone Queue weniger Stufen enthält?

Beim Schaltungsentwurf ist immer von Interesse, die Siliziumfläche des Schaltkreises möglichst gering zu halten, um höhere Materialkosten, sowie den erhöhten Energiebedarf, der für die Taktverteilung auf großen Schaltungen notwendig ist, zu vermeiden. Ein Ansatz zur Reduktion der Fläche des Receivers ist die Verkleinerung der Asynchronen Queue. Diese Struktur besteht aus 48 Stufen, da die Daten symbolweise zurückgeführt werden und ein Symbol genau 48 Token enthält. Es ist jedoch nicht zwingend notwendig, Daten symbolweise zu transportieren. Demzufolge stellt sich die Frage, ob 48 Stufen tatsächlich notwendig sind, um die einzelnen Token rechtzeitig nach B2 zu transportieren. Möglicherweise sind weniger Stufen ausreichend. Dies hätte zum Vorteil, den Basisbandprozessor mit weniger Chipfläche und damit einhergehenden niedrigeren Kosten herzustellen. Die Asynchrone Queue soll dahingehend untersucht werden, ob die Anzahl ihrer Stufen reduziert werden kann, jedoch gleichzeitig das Zeitverhalten des Gesamtsystems bewahrt

wird. Konkret betrachten wir dazu wieder den Ein- und Ausgang der Asynchronen Queue und untersuchen die 3. Frage für weniger Stufen: Ist es für die Asynchrone Queue möglich, mit weniger Stufen alle 4000 ns ein Symbol aufnehmen bzw. ausgeben?

Das von den Entwicklern abgeschätzte Verhalten der Schaltung wurde durch Testläufe ermittelt. Die Eigenschaften der Schaltung können jedoch durch reines Testen nicht garantiert werden. Um den Basisbandprozessor zu verbessern, benötigen die Entwickler bestimmte Aussagen über die Schaltung. Diese Aussagen können mittels computergestützter Verifikation getroffen werden. In Kapitel 3 bilden wir den Receiver auf ein Petrinetz-Modell ab, anhand dessen wir die hier erläuterten Fragen mittels Verifikation (bzw. Erreichbarkeitsanalyse) beantworten können. Zur Zusicherung bestimmter funktionaler Eigenschaften eines GALS-Blocks wurden bereits klassische Petrinetze verwendet und in [SRK05] erfolgreich zur Hazardanalyse der Schaltung eingesetzt. Wir verwenden im Folgenden Zeit-Petrinetze, da Zeit die relevante Größe unserer Untersuchungen ist. In Kapitel 3.3 zeigen wir, wie die hier besprochenen Fragestellungen im Modell beantwortet werden können. Dazu bilden wir die vier Fragen auf Eigenschaften von Petrinetzen ab und fragen, ob diese Eigenschaften im Modell erfüllt sind.

3. Formales Modell

Nachdem wir in Kapitel 2 den zu untersuchenden Schaltkreis beschrieben und die zu analysierenden Fragen erläutert haben, beschäftigen wir uns in diesem Kapitel mit der Modellierung der Schaltung. Um die vorgestellte Problematik zu untersuchen, benutzen wir ein mathematisches Modell, dessen formale Verifikation zeigen soll, ob die Implementation des Receivers seiner Spezifikation entspricht bzw. die Datenübertragung in dieser Komponente korrekt vollzogen wird. Durch reines Testen kann die Korrektheit des Systems nicht nachgewiesen werden, da bei diesem Verfahren lediglich einige Abläufe, jedoch nicht alle möglichen Abläufe untersucht werden. Die formale Verifikation dagegen ermöglicht es uns, den gesamten Zustandsraum der Schaltung zu untersuchen und somit genauere Aussagen über das System zu treffen.

Auf Grund der in Kapitel 2 beschriebenen nebenläufigen Abläufe in einigen der GALS-Blöcke und der Relevanz des zeitlichen Verhaltens für die Untersuchung des Schaltkreises, verwenden wir im Folgenden Zeit-Petrinetze. Dazu führen wir in Kapitel 3.1 zunächst den angewendeten Formalismus ein. Weiterhin beschreiben wir in Kapitel 3.2 eingehend die Modellierung des Receivers und validieren das Modell. In Kapitel 3.3 zeigen wir, dass sich die Fragen aus Kapitel 2.3 bzw. die zu untersuchenden Eigenschaften der Schaltung auf Eigenschaften des Petrinetz-Modells abbilden lassen. Des Weiteren erklären wir, wie die Fragestellungen mit Hilfe des erstellten Modells untersucht werden können.

3.1. Petrinetze und Zeitpetrinetze

Der zu untersuchende Schaltkreis besteht aus mehreren einzelnen GALS-Blöcken, die asynchron kommunizieren. Alle Blöcke der Schaltung führen ihre Funktionen auf verschiedene Art aus: B2, B4 und die Asynchrone Queue berechnen Daten nebenläufig, B1 und B3 führen sequentielle Berechnungen durch. Um die Vorgänge im Receiver adäquat darzustellen ist ein Formalismus nötig, der nebenläufige und sequentielle Abläufe modellieren kann. In [SRK05] wurden Petrinetze zur Modellierung der gegebenen Schaltung bei der Untersuchung funktionaler Eigenschaften bereits erfolgreich eingesetzt. Da das Zeitverhalten der Schaltung im Mittelpunkt unserer Untersuchungen steht, liegt es nahe, diesen Formalismus mit der Modellgröße Zeit zu erweitern.

3.1.1. Definitionen

Im Folgenden definieren wir Petrinetze bzw. Stellen-Transitions-Netze (siehe [Rei85] für eine Einführung) und einige relevante Eigenschaften. Petrinetze stellen Nebenläufigkeit im Sinne einer kausalen Unabhängigkeit dar und bieten eine Vielzahl von Analysemöglichkeiten. Die Zeit als relevante Größe im Modell kann mit diesem Formalismus jedoch nicht untersucht werden.

Wir führen zunächst den Begriff der Multimenge ein. Für eine Menge M ist die Abbildung $m : M \rightarrow \mathbb{N}$ eine Multimenge. Mit $[\]$ bezeichnen wir die leere Multimenge ($[\](x) = 0$ für alle x) und mit $[x]$ die einelementige Multimenge ($x = 1$, $[x](y) = 0$ mit $y \neq x$). Für $k \in \mathbb{N}$ bezeichnen wir die Multimenge mit k Vorkommen von x mit $[k \cdot x]$ ($[k \cdot x](x) = k$, $[k \cdot x](y) = 0$ für $y \neq x$).

Definition 1 (Petrinetz)

Ein 5-Tupel $N = (P, T, F, V, m_0)$ heißt genau dann ein markiertes Petrinetz N (kurz: Petrinetz), wenn gilt:

- P und T sind zwei endliche Mengen mit $P \cap T = \emptyset$. Die Elemente von P heißen Plätze und die Elemente von T heißen Transitionen.
- F ist eine zweistellige Relation mit $F \subseteq (P \times T) \cup (T \times P)$. Die Elemente von F heißen Kanten. F heißt die Flussrelation von N .
- $V : F \rightarrow \mathbb{N}$ ist die Vielfachheit (das Gewicht) der Kanten.
- Die Anfangsmarkierung m_0 ist eine Markierung, das heißt eine Multimenge.

┘

Wir benutzen die übliche graphische Darstellung von Petrinetzen und stellen Plätze durch Kreise und Transitionen durch Rechtecke dar. Aus Platzgründen schreiben wir die Transitionennamen in die Rechtecke. Die Flussrelation stellen wir durch gerichtete Kanten dar, die mit Vielfachheiten beschriftet sind (siehe das Netz N_1 in Abb. 3.1). Wenn für ein $f \in F$ das Gewicht $V(f) = 1$ ist, verzichten wir auf eine Beschriftung (in Abb. 3.1 haben alle unbeschrifteten Kanten das Gewicht 1). Jedem Petrinetz wird eine Anfangsmarkierung m_0 zugewiesen, die jedem Platz eine natürliche Zahl zuordnet. Solange lediglich eine Marke auf einem Platz liegt, stellen wir diese als schwarzen Punkt dar. Auf dem Platz p_1 in Abb. 3.1 liegt eine Marke ($m_0(p_1) = 1^1$). Ansonsten beschreiben wir die Marken-

¹Wir benutzen serifenlose Schrift, wenn wir uns auf Netze oder Knoten in Abbildungen beziehen.

anzahl eines Platzes mit einer natürlichen Zahl. So beschreibt die Multimenge $[1'p_1]$ die Anfangsmarkierung von N_1 .

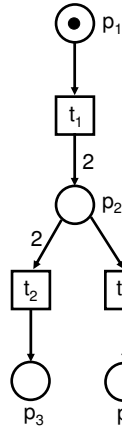


Abbildung 3.1.: N_1 ist ein Petrinetz.

Bevor wir die Schaltregel für Petrinetze definieren, erläutern wir einige Grundbegriffe.

Definition 2 (Vorbereich und Nachbereich eines Knotens)

Sei $x \in P \cup T$ ein Knoten eines Petrinetzes N . Wir definieren $\bullet x = \{y \mid [y, x] \in F\}$ als Vorbereich von x und $x \bullet = \{y \mid [x, y] \in F\}$ als Nachbereich von x . \lrcorner

Die Markierung eines Petrinetzes gibt Auskunft über Anzahl und Verteilung der Marken auf dessen Plätzen. Ist der Vorbereich einer Transition im Petrinetz den Kantengewichten entsprechend markiert, so ist diese Transition aktiviert (konzessioniert). Konzessionierte Transitionen, die keinen gemeinsamen Vorbereich haben, sind unabhängig und schalten nebenläufig, also zu unvorhersehbaren Zeitpunkten und ohne eine bestimmte Reihenfolge.

Definition 3 (Konzession, Schalten)

Sei $N = (P, T, F, V, m_0)$ ein Petrinetz. Eine Transition $t \in T$ hat genau dann Konzession bei Markierung m (ist aktiviert in Markierung m), wenn für alle Plätze $p \in \bullet t$ gilt: $m(p) \geq V([p, t])$. Eine aktivierte Transition t kann schalten und führt dann von Markierung m zu Markierung m' ($m \xrightarrow{t} m'$) mit: $m'(p) = m(p) - V([p, t]) + V([t, p])$ für alle Plätze $p \in P$. \lrcorner

Definition 4 (Schaltsequenz)

Sei $N = (P, T, F, V, m_0)$ ein Petrinetz und m eine Markierung von N . Sei w eine Schaltsequenz mit:

- $m \xrightarrow{\varepsilon} m$ (ε ist die leere Schaltsequenz),
- Wenn $m \xrightarrow{w} m_1$ und $m_1 \xrightarrow{t} m'$, so gilt $m \xrightarrow{wt} m'$.

┘

Zusammen mit Definition 3 folgt:

Korollar 3.1 (Monotonie des Schaltens) Sei $N = (P, T, F, V, m_0)$ ein Petrinetz, w eine Schaltsequenz und m, m', m_1 und m'_1 Markierungen in N . Sei $m \xrightarrow{w} m'$ und $m_1(p) \geq m(p)$. Dann gibt es eine Markierung m'_1 mit $m_1 \xrightarrow{w} m'_1$.

Definition 5 (Erreichbare Markierung)

Sei $N = (P, T, F, V, m_0)$ ein Petrinetz. Eine Markierung m' von N heißt erreichbar in N von m aus, wenn eine Schaltsequenz w von m nach m' in N existiert. Wenn $m = m_0$ ist, nennt man m' in N erreichbar.

┘

Die Menge $RG_N(m_0) = \{m_0 \mid m_0 \xrightarrow{*} m\}$ beschreibt alle von der Anfangsmarkierung m_0 erreichbaren Markierungen in einem Petrinetz N . Dabei bezeichnet $*$ eine beliebige Schaltsequenz. Wir bezeichnen die Menge $RG_N(m_0)$ als den *Erreichbarkeitsgraph* von N .

Definition 6 (Erreichbarkeitsgraph)

Sei $N = (P, T, F, V, m_0)$ ein Petrinetz und m eine Markierung. Der Graph $RG_N(m)$ ist wie folgt definiert:

- m ist ein Knoten von $RG_N(m)$.
- Wenn m' ein Knoten von $RG_N(m)$ ist und es eine Transition t mit $m' \xrightarrow{t} m''$ gibt, so ist m'' ein Knoten und (m', m'') eine mit t beschriftete Kante von $RG_N(m)$.

Wir nennen $RG_N(m_0)$ den Erreichbarkeitsgraph von N .

┘

Beim Schalten einer Transition wechselt das Petrinetz von einer Markierung in eine andere Markierung. Im Erreichbarkeitsgraph entspricht dieser Vorgang einem Zustandswechsel. Im Folgenden bezeichnen wir den Erreichbarkeitsgraph eines Petrinetzes auch als Zustandsraum. Der Zustandsraum eines Petrinetzes N enthält alle erreichbaren Zustände und Zustandsübergänge von N und beschreibt so vollständig dessen Verhalten.

Die Ein- und Ausgabeplätze eines Petrinetzes nennen wir die *Schnittstellenplätze* P_I mit $P_I = P_{in} \cup P_{out}$, $P_I \subseteq P$, $P_{in} \cap P_{out} = \emptyset$ und für alle Transitionen $t \in T$ gilt: wenn $p \in P_{in}$ ($p \in P_{out}$), so $[t, p] \notin F$ ($[t, p] \in F$). Wir komponieren Petrinetze, indem wir ihre „passenden“ Schnittstellenplätze verschmelzen.

Definition 7 (Komposition von Petrinetzen)

Seien N und N' zwei Petrinetze. Die Komposition $m \oplus m' : P \cup P' \rightarrow \mathbb{N}$ zweier Markierungen m von N und m' von N' ist definiert als $(m \oplus m')(p) = m(p)$ mit $p \in P$ und $(m \oplus m')(p) = m'(p)$ mit $p \in P'$. Entsprechend definieren wir die Komposition der Gewichtsfunktion als $(W \oplus W')(f) = W(f)$ mit $f \in F$ und $(W \oplus W')(f) = W'(f)$ mit $f \in F'$.

Die Komposition von N und N' ist das Netz $N \oplus N'$, dessen Komponenten wie folgt definiert sind:

- $P_{N \oplus N'} = P \cup P'$,
- $T_{N \oplus N'} = T \cup T'$,
- $F_{N \oplus N'} = F \cup F'$,
- $W_{N \oplus N'} = W \oplus W'$,
- $m_{0_{N \oplus N'}} = m_0 \oplus m'_0$.
- $P_{in_{N \oplus N'}} = (P_{in_N} \setminus P_{out_{N'}}) \cup (P_{in_{N'}} \setminus P_{out_N})$.
- $P_{out_{N \oplus N'}} = (P_{out_N} \setminus P_{in_{N'}}) \cup (P_{out_{N'}} \setminus P_{in_N})$.

┘

Die Definition eines Petrinetzes mit Schnittstellenplätzen und ihre Komposition ist an die Definition offener Workflow-Netze (siehe beispielsweise [LMW07]) angelehnt.

3.1.2. Eigenschaften von Petrinetzen

Im Folgenden definieren wir einige Eigenschaften von Petrinetzen, die im Rahmen der Untersuchungen des Receivers relevant sind.

Definition 8 (Beschränktheit)

Sei $N = (P, T, F, V, m_0)$ ein Petrinetz. N heißt genau dann beschränkt, wenn die Menge der erreichbaren Markierungen von N , $RG_N(m_0)$, endlich ist.

┘

Das Netz N ist k -beschränkt, wenn für alle Markierungen $m \in RG_N(m_0)$ gilt $m(p) \leq k$. Wir bezeichnen einen Platz p als *sicher*, wenn für alle Markierungen $m \in RG_N(m_0)$ gilt $m(p) \leq 1$. Für beschränkte Petrinetze können eher Verifikationsresultate ermittelt werden, als für unbeschränkte.

Da sich diese Arbeit unter anderem mit dem Auffinden toter und lebendiger Transitionen in Petrinetzen beschäftigt, definieren wir auch diese.

Definition 9 (Tote Transition)

Sei $N = (P, T, F, V, m_0)$ ein Petrinetz und m eine Markierung von N . Eine Transition $t \in T$ heißt tot bei m , falls t bei keinem $m' \in RG_N(m)$ Konzession hat. \lrcorner

Definition 10 (Lebendige Transition)

Sei $N = (P, T, F, V, m_0)$ ein Petrinetz und m eine Markierung in N . Eine Transition $t \in T$ heißt lebendig bei m , falls von jeder Markierung $m' \in RG_N(m)$ eine Markierung m'' erreichbar ist, in der t aktiviert ist. \lrcorner

Zuletzt definieren wir Verklemmungen:

Definition 11 (Verklemmung)

Sei $N = (P, T, F, V, m_0)$ ein Petrinetz. Eine Verklemmung (ein Deadlock) ist eine Markierung von N , in der keine Transition aktiviert ist. \lrcorner

3.1.3. Intervall-Petrinetze

Die klassischen Petrinetze beinhalten die Zeit nur als kausalen Zusammenhang. Für die Untersuchung der stark zeitabhängigen Berechnungen des Receivers ist jedoch die explizite Angabe der Zeit notwendig. Darum beschreiben und analysieren wir den gegebenen Schaltkreis mit einer Klasse von Zeit-Petrinetzen. Wir wählen Intervall-Petrinetze als Formalismus des zu untersuchenden Modells. Intervall-Petrinetze eignen sich besonders für die Modellierung des Receivers, da diese Klasse von Petrinetzen minimale und maximale Reaktionszeiten modelliert. Die folgenden Definitionen entstammen [PZ06] und wurden für den Rahmen dieser Arbeit angepasst.

Intervall-Petrinetze entstehen aus Stellen-Transitions-Netzen, indem jeder Transition t ein Intervall $[eft(t); lft(t)]$ zugeordnet wird. Ist eine Transition t konzessioniert, ist sie frühestens nach $eft(t)$ Zeiteinheiten aktiviert. Die Transition t schaltet spätestens nach $lft(t)$ Zeiteinheiten, es sei denn, sie hat ihre Konzession verloren. Dies geschieht, wenn

eine andere Transition schaltet, die mit t einen gemeinsamen Vorplatz hat. Die Zeit $eft(t)$ ist also die frühest mögliche Schaltzeit für t (earliest firing time), die Zeit $lft(t)$ die spätest mögliche Schaltzeit (latest firing time). Das Schalten einer Transition findet dabei zeitlos statt. Die Zeit wird bei Intervall-Petrinetzen mit reellen Zahlen modelliert, wobei die Intervallgrenzen nichtnegative rationale Zahlen sind.

Wir gewinnen ein Stellen-Transitions-Netz aus einem Intervall-Petrinetz, indem wir an allen Transitionen die existierenden Intervalle durch $[0; \infty]$ ersetzen.

Definition 12 (Intervall-Petrinetz, Skelett)

Ein 6-Tupel $Z = (P, T, F, V, m_0, I)$ heißt genau dann Intervall-Petrinetz, wenn:

- das 5-Tupel $N(Z) = (P, T, F, V, m_0)$ ein Petrinetz ist (das Stellen-Transitions-Netz $N(Z)$ nennen wir das Skelett von Z) und
- $I : T \rightarrow \mathbb{Q}_0^+ \times (\mathbb{Q}_0^+ \cup \{\infty\})$ die Intervallfunktion ist mit $I_1(t) \leq I_2(t)$ für jedes $t \in T$, wobei $I(t) = [I_1(t); I_2(t)]$.

┘

Intuitiv können wir uns Intervall-Petrinetze als Stellen-Transitions-Netze vorstellen, deren Transitionen je eine Uhr zugeordnet ist. Initial sind in einem Intervall-Petrinetz alle Uhren auf Null gesetzt. Die Schaltregel besagt, dass die Uhr einer Transition t zu laufen beginnt, sobald t konzessioniert ist. Sobald die untere Intervallgrenze $eft(t)$ erreicht ist, ist t aktiviert. Bei Stellen-Transitions-Netzen fallen die Begriffe Konzession und Aktivierung zusammen.

Definition 13 (Aktivierung, Transitionsmarkierung)

Sei $Z = (P, T, F, V, m_0, I)$ ein Intervall-Petrinetz. Eine Transition t heißt aktiviert, falls

- t konzessioniert ist und
- $h(t) \geq eft(t)$, wobei $h : T \rightarrow \mathbb{R}_0^+ \cup \{\#\}$ eine Transitionsmarkierung ist.

┘

Die Transitionsmarkierung gibt Auskunft über die Dauer einer Konzessionierung und ordnet jeder aktivierten Transition die Zeit zu, die seit ihrer letzten Konzessionierung vergangen ist. Transitionen, die nicht aktiviert sind, ordnet die Transitionsmarkierung den Wert $\#$ zu.

Definition 14 (Zustand, Anfangszustand)

Sei $Z = (P, T, F, V, m_0, I)$ ein Intervall-Petrinetz, m eine Markierung und h eine Transitionsmarkierung in Z . Dann ist das Paar $z = (m, h)$ ein Zustand in Z genau dann, wenn:

- $\forall t \in T : \begin{cases} h(t) = \#, & \text{falls } t \text{ nicht konzessioniert ist,} \\ h(t) \in \mathbb{R}_0^+ \wedge h(t) \leq lft(t), & \text{sonst.} \end{cases}$

Den Zustand $z_0 = (m_0, h_0)$ mit:

- $h_0(t) = \begin{cases} 0, & \text{falls } t \text{ in } m_0 \text{ konzessioniert ist,} \\ \#, & \text{sonst.} \end{cases}$

nennen wir Anfangszustand von Z . ┘

Schaltet die Transition t innerhalb der vorgegebenen Intervallgrenze, wird deren Uhr anschließend auf Null gesetzt. Falls zwei Transitionen mit mindestens einem gemeinsamen Vorplatz konzessioniert sind und eine dieser Transitionen schaltet, wird auch in dem Fall, dass die andere Transition immer noch Konzession hat, ihre Zeit neu gezählt, das heißt ihre Uhr auf Null gesetzt.

Definition 15 (Schaltregel)

Sei t eine Transition und $z = (m, h)$ ein Zustand des Intervall-Petrinetzes Z . Dann kann t im Zustand z schalten, falls t aktiviert in z ist mit $(z \xrightarrow{t} z')$. Nach dem Schalten befindet sich das Intervall-Petrinetz im Nachfolgezustand $z' = (m', h')$ mit

- $\forall p \in P : m'(p) = m(p) - V([p, t]) + V([t, p]),$
- $\forall t' \in T : h'(t') = \begin{cases} \#, & \text{falls } t' \text{ in } m' \text{ nicht konzessioniert ist,} \\ h(t'), & \text{falls } t' \text{ in } m \text{ und } m' \text{ konzessioniert ist und } \bullet t' \cap \bullet t = \emptyset, \\ 0, & \text{sonst.} \end{cases}$

┘

Während in Stellen-Transitions-Netzen die Transitionen unabhängig schalten können sobald sie konzessioniert sind, ergibt sich für Intervall-Petrinetze ein Schaltzwang für alle Transitionen t , sofern $lft(t) \neq \infty$ ist. Insbesondere schaltet eine Transition t mit $I(t) = [0; 0]$ *instantan* sobald sie konzessioniert ist. Daraus ergibt sich für die Intervall-Petrinetze eine gewisse Schaltreihenfolge bei aktivierten Transitionen. Abbildung 3.2 verdeutlicht, dass ein Intervall-Netz gegenüber seinem Skelett im Verhalten eingeschränkt sein kann. Das Einführen von Zeitintervallen bewirkt hier, dass t_2 tot ist, da t_3 stets frü-

her schaltet und t_2 die Konzession entzieht. Im Skelett von Z_1 , dem Netz N_1 aus Abb. 3.1, ist dies nicht der Fall.

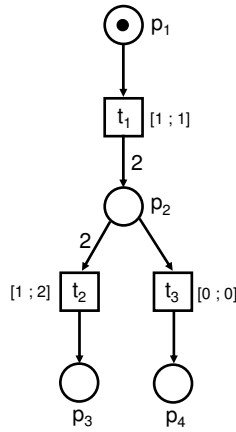


Abbildung 3.2.: Das Intervall-Petrinetz Z_1 .

Auch im Modell des Receivers, legen wir für bestimmte Transitionen einen Schaltzwang fest, indem wir $eft(t) = lft(t) = 0$ setzen. Für andere Transitionen des Modells bestimmen wir, dass sie erst nach einer gewissen Zeit schalten. Dadurch ist es möglich, das reale Verhalten des GALS-Schaltkreises adäquat abzubilden.

Zur besseren Übersichtlichkeit verwenden wir im Folgenden für das Intervall einer Transition t die Notation $[x]$, wenn $eft(t) = lft(t) = x$ ist. So sind in großen Netzen Transition wie t_1 in Abb. 3.2 mit dem Intervall $[1]$ beschriftet. Jede Transition, die wir in Abbildungen nicht mit einem Intervall versehen, schaltet instantan.

Jeder Zustand eines Intervall-Petrinetzes ist ein geordnetes Paar, das aus einer Markierung² (Tupel aus natürlichen Zahlen) und einer Transitionsmarkierung (Tupel aus reellen Zahlen) besteht. Bevor t schaltet, muss die Transition mindestens $eft(t)$ Zeiteinheiten konzessioniert gewesen sein. Das bedeutet, dass zwischen dem Schalten zweier Transitionen stets Zeit vergeht. Sei $\tau = \tau_1 \dots \tau_n$ mit $\tau_i \in \mathbb{R}_0^+$ (mit $i \in \{1, \dots, n\}$) eine Sequenz von Zeiten. Die Sequenz $w = \tau_1 t_1 \tau_2 \dots \tau_n t_n$ nennen wir eine Schaltsequenz in Z . Mit $z \xrightarrow{\tau} z'$ notieren wir einen Zustandswechsel bei dem die Markierungen gleich bleiben, die Transitionsmarkierungen sich jedoch ändern.

²Popova-Zeugmann führt in [PZ06] den Begriff Platzmarkierung ein, der mit unserer Definition der Markierung zusammenfällt.

Definition 16 (Schaltsequenz)

Sei Z ein Petrinetz und $z = (m, h)$ ein Zustand von Z . Sei $w = \tau_1 t_1 \tau_2 t_2 \dots \tau_n t_n$ eine Schaltsequenz mit:

- $z \xrightarrow{\varepsilon} z$ (ε ist die leere Schaltsequenz),
- Wenn $z \xrightarrow{\tau_1 t_1 \dots \tau_n t_n} z_1$ und $z_1 \xrightarrow{\tau_{n+1}} z_2$ und $z_2 \xrightarrow{t_{n+1}} z'$, so gilt $z \xrightarrow{wt} z'$.

┘

Im Folgenden definieren wir die Zustände als Knoten des Erreichbarkeitsgraphen eines Intervall-Petrinetzes. Zwei Zustände z und z' werden mit einer gerichteten Kante verbunden, wenn es möglich ist aus dem Zustand z durch das Schalten einer Transition oder durch das Fortlaufen der Zeit in den Zustand z' zu gelangen.

Die Begriffe *erreichbarer Zustand* und *Erreichbarkeitsgraph* werden kanonisch auf Intervall-Petrinetze erweitert. Der Erreichbarkeitsgraph eines Intervall-Petrinetzes wird von einem Verifikationswerkzeug berechnet. Intervall-Petrinetze sind äquivalent zu Turingmaschinen, das heißt, dass wir Erreichbarkeit für diesen Formalismus auf das Halteproblem zurückführen können: Erreichbarkeit ist für Intervall-Petrinetze unentscheidbar. Ist jedoch für alle Transitionen eines Intervall-Petrinetzes eine obere Zeitschranke ($lft(t) \neq \infty$) vermerkt, handelt es sich um ein *finites* Netz, dessen Erreichbarkeitsgraph auf einen endlichen Graphen abgebildet werden kann.

Satz 3.2 (aus [PZ06]) *Sei Z ein finites Intervall-Petrinetz. Dann gilt: Die Menge aller in Z erreichbaren ganzzahligen Zustände ist genau dann endlich, wenn die Menge aller in Z erreichbaren Markierungen endlich ist.*

Die Definitionen 8, 9, 10 und 11 der in 3.1.2 aufgeführten Petrinetz-Eigenschaften werden ebenfalls kanonisch auf Intervall-Petrinetze erweitern.

3.2. Modell

In diesem Abschnitt modellieren wir die Komponenten des Receivers mit Intervall-Petrinetzen. Für jede Komponente präsentieren wir ein Netz (Abb. 3.5 bis 3.10). Diese Netze wurden in Zusammenarbeit mit den Entwicklern des Schaltkreises validiert. Ihr Einverständnis mit der Korrektheit der Modellierung des Schaltkreises ist die Basis für alle weiteren Untersuchungen. Im Folgenden motivieren wir unsere Designentscheidungen und erläutern die Überführung der gegebenen Schaltung in ein formales Modell.

3.2.1. Grundlegende Designentscheidungen

Die kleinste Dateneinheit, also das Token im Schaltkreis entspricht im Petrinetzmodell einer Marke. Da wir die nicht-funktionalen Eigenschaften der Schaltung analysieren, abstrahieren wir von den Werten der Token der Schaltung und stellen diese als einfache, ununterscheidbare Marken im Intervall-Petrinetz dar. Liegt in der Schaltung ein bestimmtes Signal an einem Gatter an, entspricht dies im Petrinetzmodell einer Marke auf einem Platz. Das Auswerten des Signalpegels entspricht im Petrinetz dem Schalten einer Transition, also dem Konsum der Marke. Ein Signalpegelwechsel in der Schaltung wird durch das Produzieren einer weiteren Marke auf dem Platz widerspiegelt.

Ein Symbol, das aus der Umgebung in den Schaltkreis eintritt, besteht aus 80 Token, wir nennen es auch *externes Symbol*. Block B2 nimmt ein externes Symbol auf und generiert dafür ein *internes Symbol*, das aus 48 Token besteht. Dieser Vorgang erfolgt für jeden Frame R -mal, wobei R die Anzahl der Symbole pro Frame ist. Nach Abarbeitung eines Frames wird der Receiver zurückgesetzt, weswegen wir lediglich das Verhalten des Receivers während der Verarbeitung eines Frames untersuchen. Frames sind von variabler Länge und enthalten maximal 1300 Symbole. Unter Absprache mit den Entwicklern der Schaltung gehen wir im Folgenden von einigen hundert Symbolen pro Frame aus.

Beim Modellieren der gegebenen Schaltung bilden wir die einzelnen Funktionen, die der Receiver implementiert, nicht explizit im Petrinetz ab. Natürlich fließen Informationen über die Berechnungen und Funktionen mit ins Modell ein. Relevant für die Beantwortungen der Fragen 1 bis 4 in Kapitel 2 ist jedoch nicht das funktionale Verhalten des Schaltkreises, sondern lediglich der Kontrollfluss und das zeitliche Verhalten des Datenflusses. Genau diese Informationen halten wir im Modell fest. Wir abstrahieren von den internen Details, die auf den Daten- und Kontrollfluss keinen Einfluss haben und halten lediglich Verzweigungen und Zusammenführungen der Daten im Modell fest. Vor allem die minimale und maximale Reaktionszeit der Summe der Gatter in den GALS-Blöcken ist zur Untersuchung der Fragen im Modell notwendig. Jedes im Folgenden beschriebene Petrinetz bildet die in Kapitel 2 beschriebenen Phasen (Request-gesteuerte, Time-out und Local-Clock Phase) auf wenige Transitionen ab. Die jeweiligen Intervallgrenzen sind durch die minimale und maximale Dauer der einzelnen Phasen, die ein GALS-Block einnehmen kann bestimmt, bzw. durch die Dauer der Funktionen, die ein Block ausführt.

Abbildung 3.3 zeigt, wie die einzelnen Petrinetze zusammenhängen. Aus der Komposition der Petrinetze N_{Env} , N_{B2} , N_{B3} , N_{B4} und N_{aQu} entsteht das Petrinetz, das den gesamten Schaltkreis modelliert (der Receiver und die Umgebung).

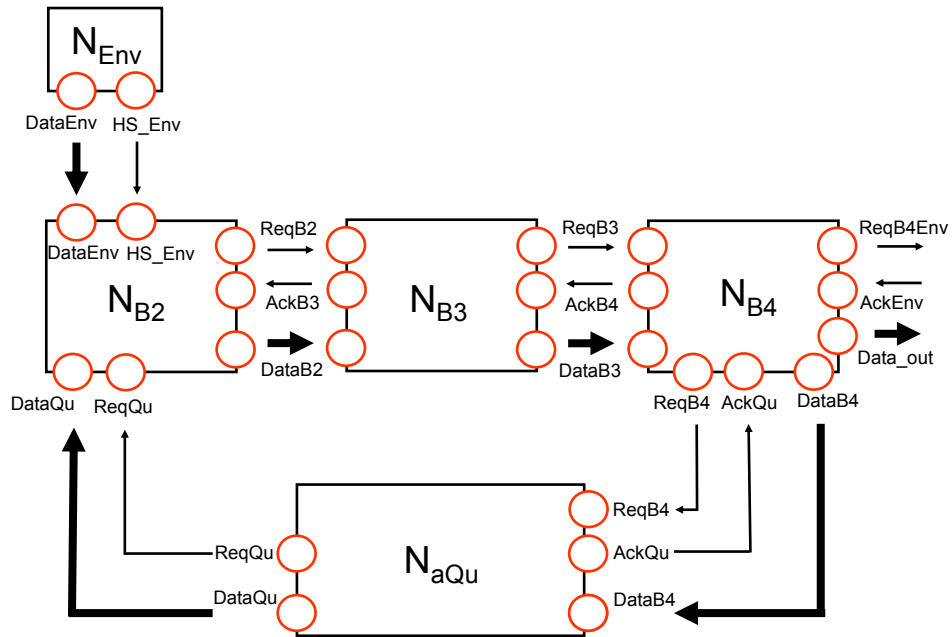


Abbildung 3.3.: Schnittstellenmodell der GALS-Blöcke des Receivers.

3.2.2. Schnittstellen

Die Schnittstellenplätze der einzelnen Petrinetze modellieren die Schnittstellen der einzelnen GALS-Blöcke im Receiver. In Abb. 3.4 ist die Struktur der Schnittstellen zweier Blöcke A und B als Petrinetze N_A und N_B veranschaulicht. Die Schnittstellenplätze in Abb. 3.4 sind rot dargestellt. Wir verwenden diese Struktur im Weiteren für die Netze, die die Blöcke B2 und B4 modellieren. Die Plätze $ReqA$ und $AckB$ stellen den Kontrollfluss dar und modellieren so eingehende und ausgehende Handshakes. Marken, die sich auf dem Platz $DataA$ ansammeln, stellen den Datenfluss dar. Sobald eine Marke auf $DataA$ produziert wird, entspricht dies in der Schaltung einem Zustand, in dem ein Pegelwechsel des Nutzsignals stattfindet, also gültige Daten am Eingang des Blocks B anliegen.

Handshakes an den Schnittstellen der Wrapper folgen einem 4-Phasen-Protokoll [Pee96]. Das heißt, dass das Kontrollsignal zwei Pegelwechsel pro Handshake vollzieht. Ein steigender Pegel am Request-Eingang des Wrappers ($ReqA+$) wird mit einem steigenden Pegel am Acknowledge-Ausgang bestätigt ($AckA+$). Diese Signalfolge signalisiert einem Block, dass anliegende Daten gültig sind und verarbeitet werden müssen. Eine neue

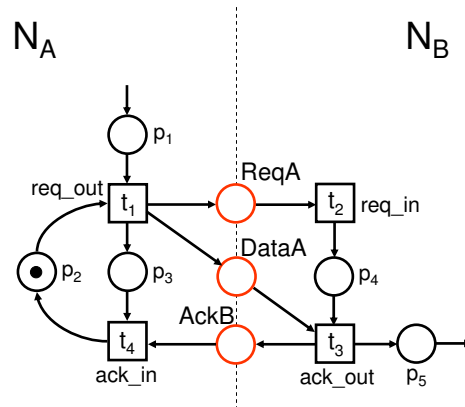


Abbildung 3.4.: Schnittstellenmodell zweier GALs-Blöcke A und B.

Anfrage am Eingang des Wrappers wird erst wieder bearbeitet, sobald die Signalfolge $ReqA-$, $AckA-$ stattgefunden hat.

Die Einhaltung des 4-Phasen-Protokolls stellt sicher, dass während der Datenaufnahme eines Wrappers keine Informationen im System verloren gehen. Sollte Block B kurzzeitig nicht in der Lage sein, Daten von Block A entgegenzunehmen, pausiert A so lange in seiner Ausgabebetätigkeit, bis B den Handshake erwidert. Am Eingang eines GALs-Blocks werden also keine gültigen Daten überschrieben. Das Petrinetz in Abb. 3.4 spiegelt dieses Verhalten wider: Ist der Platz $ReqA$ markiert (t_1 hat bereits geschaltet und t_2 ist aktiviert) entspricht dies im Schaltkreis dem Zustand $ReqA+$ am Eingang des Wrappers B. Schaltet t_2 , entspricht dies im Schaltkreis dem Ereignis $AckB+$. Die Transition t_3 ist erst dann aktiviert, wenn der Platz $DataA$ markiert ist. Im Schaltkreis kann Block B Daten erst verarbeiten, wenn sie gültig sind. Das Auswerten des anliegenden Nutzsignalpegels, wird durch den Konsum der Marke auf dem Platz $DataA$ modelliert. Die Aufnahme gültiger Daten (Schalten von Transition t_3) erfolgt nur nachdem t_2 geschaltet hat. Transition t_3 produziert die Marken, die die gültigen Daten modellieren, in das Netz N_B . Ist daraufhin $AckB$ markiert, entspricht das dem Auftreten des Signals $ReqB-$ am Eingang von Block B; das Datum wurde aufgenommen. Schaltet t_4 ist der Handshake abgeschlossen ($AckB-$). Erst dann kann ein weiterer Handshake stattfinden, sofern auch der Platz p_1 markiert ist.

3.2.3. Die Netze N_{Env} und N_{B1}

Wie bereits in Kapitel 2 beschrieben, spielt das funktionale Verhalten von Block B1 für die zu untersuchenden Probleme keine Rolle. Um den Zustandsraum von Beginn an möglichst klein zu halten, schließen wir das Verhalten von B1 in dem Petrinetz, das die Umgebung des Receivers modelliert ein (siehe N_{Env} in Abb. 3.5). Wir bilden Block B1 im Modell nicht explizit ab und halten lediglich dessen zeitliches Verhalten fest.

Im Petrinetz modelliert eine Zeiteinheit eine Nanosekunde (ns). Die Umgebung gibt alle 4000 ns ein externes Symbol bestehend aus 80 Token an den Receiver ab. B1 synchronisiert die Daten am Eingang des Receivers und beeinflusst den restlichen Schaltkreis lediglich dahingehend, dass es die zeitliche Abfolge der Nutzdaten am Eingang von B2 bestimmt: 1600 ns lang ermittelt B1 den Beginn der Nutzdaten im Datenstrom und gibt anschließend die Daten für B2 frei. Dann verweilt B1 2400 ns im Ruhezustand und berechnet anschließend das nächsten Symbols des Frames. Das Verhalten von B1 ist konstant, es gibt keine Varianzen oder Abweichungen in der Zeit oder im Verhalten.

Es spielt für die Untersuchung des Receivers keine Rolle, ob B1 initial 1600 ns wartet oder nicht. Solange B2 im Modell alle 4000 ns ein Symbol zur Verfügung steht, ist das Auftreten der WLAN-Signale am Eingang des Receivers adäquat abgebildet. Als Umgebung modellieren wir demnach eine Komponente, die alle 4000 Zeiteinheiten ein externes Symbol an N_{B2} abgibt. Das entsprechende Netz, N_{Env} , ist in Abb. 3.5 dargestellt. Transition t_1 produziert alle 4000 Zeiteinheiten 80 Marken auf die Schnittstellenplätze von N_{B2} . Dies geschieht R mal, wobei R die Anzahl der Symbole pro Frame ist.

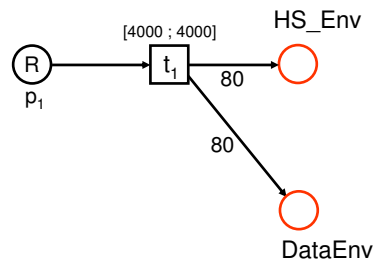


Abbildung 3.5.: Das Netz N_{Env} modelliert die Umgebung und das Verhalten des GALS-Blocks B1.

3.2.4. Das Netz N_{B2}

Block B2 nimmt die Symbole von der Umgebung auf. Sobald B1 die Daten freigibt, findet in B2 die Aufnahme und Ausgabe von Symbolen nebenläufig und mit einer Frequenz von 20 MHz statt. Während der Aufnahme des ersten bis zur Abgabe des letzten Symbols eines Frames ist B2 ständig aktiv, es sei denn, am Eingang des Blocks stehen keine Daten mehr zur Verfügung. Dabei generiert B2 alle 4000 ns für ein externes Symbol ein internes, das aus 48 Token besteht und gibt es an B3 weiter. Für die Aufnahme oder Abgabe eines Tokens benötigt B2 50 ns pro Handshake. Für die Ausgabe eines internen Symbols mit einer Taktrate von 20 MHz sind dementsprechend 2400 ns ($48 \cdot 50$ ns) erforderlich. Die Transitionen t_1 , t_5 und t_7 im Petrinetz N_{B2} , das in Abb. 3.6 dargestellt ist, modellieren die Request-Vorgänge am Eingang von B2, die für jeden Handshake vollzogen werden. Um der Datenaufnahme mit 20 MHz im Modell gerecht zu werden, schalten die jeweiligen Request-Transitionen am Ein- und Ausgang von N_{B2} genau nach 50 Zeiteinheiten. Die Transitionen, die die Bestätigung modellieren (t_2 , t_6 und t_8), schalten dagegen instantan. Somit bilden wir das Zeitverhalten an den Schnittstellen der Wrapper entsprechend im Modell ab.

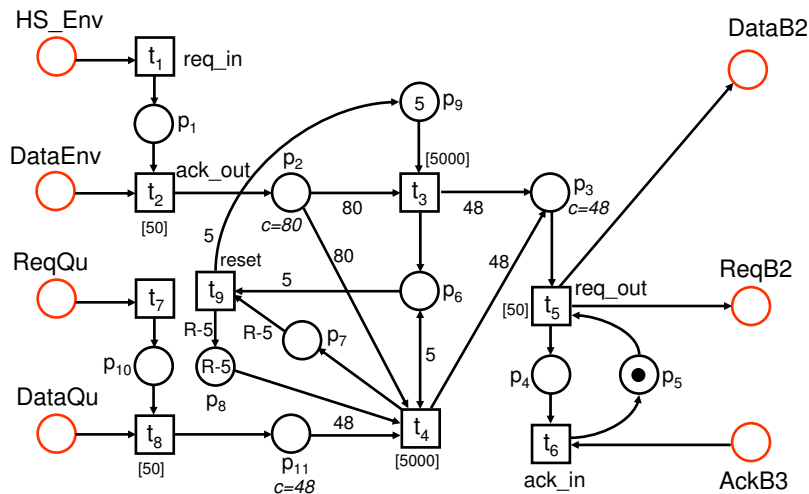


Abbildung 3.6.: Das Netz N_{B2} modelliert den GAL S-Block B2.

Anmerkung: In Abb. 3.6 tragen die Plätze p_2 , p_3 und p_{11} die Beschriftung $c = 80$ bzw. $c = 48$. Diese Plätze haben eine Kapazität c . Liegt im Nachbereich einer Transition t ein Platz mit der Kapazität c , so unterliegt diese Transition einer zusätzlichen Aktivierungs-

bedingung. Wir verändern jedoch nicht die Schaltregel sondern gehen für einen Platz p mit der Kapazität c implizit von der Existenz eines *Komplementärplatzes* \bar{p} aus mit $m_0(\bar{p}) = c$ und für alle $[t, p] \in F$ existiert eine Kante $[\bar{p}, t] \in F$ mit $V([\bar{p}, t]) = V([t, p])$ und für alle $[p, t] \in F$ existiert eine Kante $[t, \bar{p}] \in F$ mit $V([t, \bar{p}]) = V([p, t])$. Diese Konstruktion ist in Abb. 3.7(b) veranschaulicht. Der Übersichtlichkeit halber verwenden wir für dieses Muster im Folgenden stets die Darstellung aus Abb. 3.7(a). Beispielsweise schaltet die Transition t_8 nur, wenn in ihrem Nachbereich genügend Kapazitäten vorhanden sind. Ansonsten ist sie nicht aktiviert. Mit dieser Modellierung sichern wir für Block B2 zu, dass dieser seine Berechnungen einstellt, sollten keine Daten von der Asynchronen Queue zur Verfügung stehen bzw. B2 ein Symbol i nicht von der Asynchronen Queue entgegen nimmt, solange dieser das Symbol $i - 1$ noch berechnet.

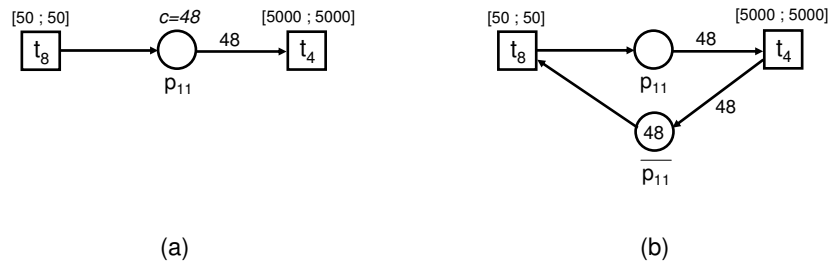


Abbildung 3.7.: Die Kapazität c des Platzes p_{11} in (a) impliziert die in (b) dargestellte zusätzliche Aktivierungsbedingung für t_8 , die durch den Platz \bar{p}_{11} modelliert wird.

Die in B2 angewendeten Funktionen spielen für die Modellierung keine Rolle, wir halten lediglich das zeitliche Verhalten und die resultierende Datenmenge im Modell fest. Auf Grund der komplexen Berechnungen in B2, beträgt die Bearbeitungsdauer für jedes Symbol 5000 ns. Analog zu den Designentscheidungen der Entwickler berücksichtigen wir für B2 keine Time-out Phase im Petrietz. Die Transitionen, die die Dauer der Funktionsberechnungen in N_{B2} abbilden (t_3 und t_4 in Abb. 3.6) schalten genau nach 5000 Zeiteinheiten. Des Weiteren konsumieren t_3 und t_4 jeweils 80 Marken und produzieren 48 Marken. Dieses Verhalten modelliert das Konzept der externen und internen Symbole im Schaltkreis.

Initial nimmt Block B2 Symbole aus der Umgebung auf und leitet sie an B3 weiter. Sollte B3 während des Betriebs nicht unterbrochen werden, liegt nach einiger Zeit das erste Symbol eines Frames in der Rückführung bereit, sobald das sechste Symbol am Eingang von B2 anliegt. Ab diesem Zeitpunkt kann B2 ein Symbol aus der Umgebung nur dann verarbeiten, wenn ein Symbol in der Asynchronen Queue verfügbar ist. Während der 4000 ns, in denen B2 ein Symbol aus der Umgebung aufnimmt, muss es eine Zeitspanne

von 2400 ns geben, in der B2 alle 50 ns ein Token aus der Queue empfängt. Ansonsten unterbricht der Datenfluss und die Funktion des Receivers ist nicht mehr korrekt.

Der Schaltkreis synchronisiert die Datenströme aus zwei Richtungen wie folgt: Ist ein Token der Umgebung am Eingang von B2 gültig, muss auch ein gültiges Token aus der asynchronen Queue verfügbar sein, damit beide Token verarbeitet werden können. Die Vereinigung des Datenstroms steuert in der Schaltung ein spezieller JOIN-Schaltkreis, der vor B2 liegt und mit Hilfe eines Kontrollsignals (EN) den Datenstrom regelt. Erreichen B2 nur Token aus der Umgebung, vollzieht der Schaltkreis gewöhnliche Handshakes nur mit der Umgebung ($EN-$). Sobald das ersten Token aus der Rückführung an B2 anliegt und mittels Handshake aufgenommen werden soll, wird ein Zähler aktiviert. Mittels der Request-Signale zählt der Zähler die von B2 aufgenommenen Token. Der Zähler setzt $EN+$, sobald alle 48 Token übermittelt wurden. Erst dann werden beide Datenströme in B2 vereinigt. Es ist allerdings nicht nötig, diesen speziellen Schaltkreis explizit im Petrinetz widerzuspiegeln, da das Modell dessen Verhalten bereits darstellt: t_4 kann nur schalten, wenn von beiden Seiten Marken verfügbar sind.

Im Modell stellen sich die Abläufe wie folgt dar. Sobald die Umgebung 80 Marken an das Netz N_{B2} abgibt, hat Transition t_3 Konzession. Transition t_3 produziert (nach der vorgegebenen Zeit) 48 Marken auf den Platz p_3 und stellt damit ein Symbol für B3 bereit. Pro Symbol produziert t_3 eine Marke auf den Platz p_6 (Kontrollfluss). Insgesamt enthält ein Frame R Symbole. Sobald die ersten fünf Symbole eines Frames berechnet sind, liegen auf p_6 fünf Marken. Ab diesem Zeitpunkt nimmt N_{B2} Daten von der Umgebung *und* der Asynchronen Queue entgegen. In dem Moment, in dem t_4 konzessioniert ist, schaltet t_3 nicht mehr. Die Vereinigung des Markenflusses aus der Umgebung und der Asynchronen Queue regelt ausschließlich t_4 . Da B2 ab diesem Zeitpunkt ein Symbol aus der Umgebung nur berechnen kann, wenn gleichzeitig ein Symbol aus der Rückführung bereit liegt, schaltet t_4 nur, wenn 80 Marken aus der Umgebung auf p_2 und 48 aus der Asynchronen Queue auf p_{11} liegen. Bei der Zusammenführung des Datenstroms besteht im Schaltkreis das Problem, die Rechengeschwindigkeit von B2 nicht zu mindern, da sonst die korrekte Datenübertragung des Receivers nicht mehr gewährleistet ist. Diese Problematik können wir mit dem beschriebenen Petrinetz adäquat untersuchen (siehe Kapitel 4).

Das Schalten der Transition t_9 modelliert schließlich das Zurücksetzen von B2, nachdem alle R Symbole eines Frames berechnet wurden. Für jedes Symbol, das t_4 berechnet, wird eine Marke auf den Platz p_7 produziert (Kontrollfluss). Sobald sich auf p_7 $R - 5$ Marken angesammelt haben (das heißt, alle Symbole eines Frames berechnet wurden), ist Transition t_9 aktiviert und schaltet sofort. Dabei wird das Netz N_{B2} in seine initiale

Markierung zurückgeführt. Diese spiegelt den Ruhezustand wider, den B2 einnimmt, nachdem ein vollständiger Frame berechnet wurde.

3.2.5. Das Netz N_{B3}

Das LS-Modul M3 des Blocks B3 bettet eine synchrone Queue mit 48 Stufen ein. B3 fungiert im Schaltkreis als Beschleuniger der Daten. Diese werden in M3 nicht weiter modifiziert, sondern lediglich für kurze Zeit zwischengespeichert. B3 nimmt Daten während der Request-getriebenen Phase mit einer Frequenz von 20 MHz von B2 entgegen und gibt sie während der Local-Clock Phase mit 80 MHz an B4 weiter. Die Abläufe in B3 finden sequenziell statt und lassen sich in die drei in Kapitel 2.1 (Abb. 2.2) beschriebenen Phasen untergliedern. Für alle anderen GALS-Blöcke des Receivers trifft dies nicht zu: B2 und B4 unterbrechen ihre Prozesse nur, wenn es ihnen nicht möglich ist, Daten vom Vorgänger entgegenzunehmen bzw. Daten an den Nachfolger weiterzugeben. In Abb. 3.8 ist das Petrinetz dargestellt, das B3 modelliert.

Während der Request-getriebenen Phase benötigt B3 2400 ns, um 48 Token aufzunehmen. Initial ist t_1 aktiviert, sobald der Schnittstellenplatz $ReqB2$ markiert ist (Kontrollfluss). Wird im Schaltkreis ein Symbol im 20 MHz-Takt von B2 nach B3 transportiert, so beträgt die Verzögerung für jedes Token 50 ns. Das Netz N_{B2} produziert alle 50 Zeiteinheiten jeweils eine Marke auf die Schnittstellenplätze von N_{B3} (siehe Transition t_5 in Abb 3.6). Da wir im Rahmen der Analyse alle Netze, die den Receiver modellieren, komponieren, ist gewährleistet, dass das zeitliche Ausgabeverhalten von N_{B2} das zeitliche Aufnahmeverhalten von N_{B3} bestimmt. Somit sind die Transitionen, die in N_{B3} die eingehenden Handshakes modellieren (t_1 , t_2 , t_3 , t_5 und t_9) instantan.

Geht der Block B3 in die Request-gesteuerte Phase über, schaltet t_1 im Netz N_{B3} . Sind die Daten am Eingang des Wrappers gültig, liegen in N_{B3} Marken auf $DataB2$. Während der anschließenden Abfolge von Handshakes schalten die Transitionen t_2 und t_3 abwechselnd (die Nummerierung der Transitionen impliziert eine mögliche Schaltreihenfolge). Um einen Handshake mit N_{B3} zu bestätigen, produziert t_2 eine Marke auf den Platz $AckB3$. Bei jeder Bestätigung eines Handshakes produziert t_2 zusätzlich eine Marke in den Vorbereich der Transitionen t_6 und t_8 (auf den Platz p_4). Diese Transitionen regeln die Handshakes am Ausgang von N_{B3} . Sobald sie konzessioniert sind, schalten sie so oft, wie zuvor t_2 am Eingang schaltet. Damit entsprechen wir der Schaltung, die genau so viele Token ausgibt wie sie zuvor aufgenommen hat. Die Transitionen t_6 und t_8 produzieren für jeden Handshake außerdem eine Marke auf den Platz $DataB3$, dies modelliert den eigentlichen Transport der Daten an den GALS-Block B4. Sobald die synchrone Queue

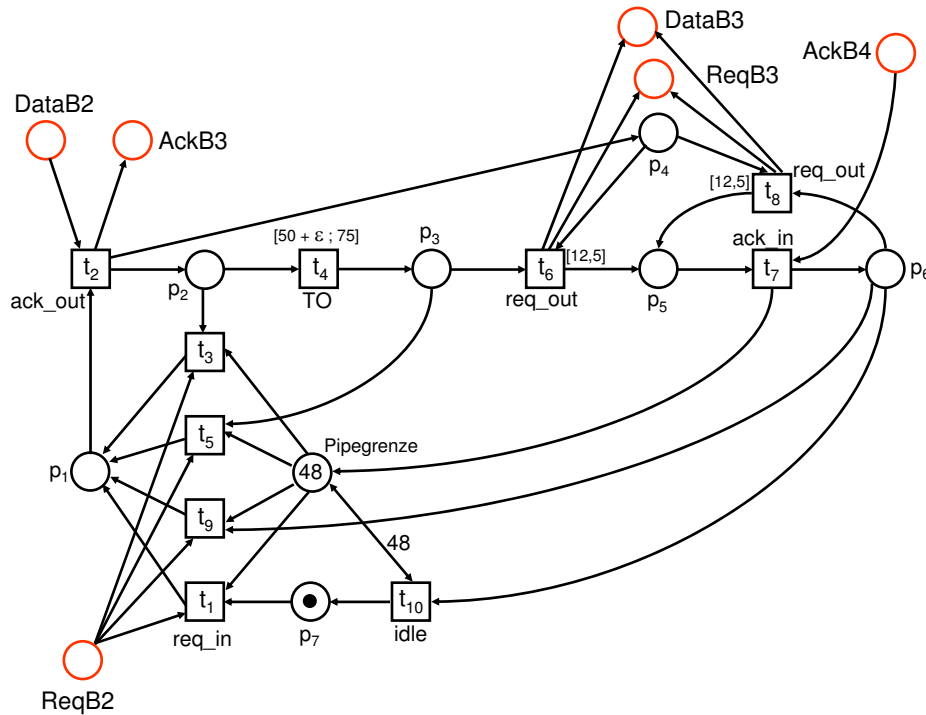


Abbildung 3.8.: Das Netz N_{B3} modelliert den GALs-Block B3.

in M3 keine Token mehr speichert, sind t_6 und t_8 deaktiviert. Die Transition t_7 modelliert die Bestätigung der Handshakes am Ausgang von B3 und produziert bei jedem schalten eine Marke auf den Platz Pipegrenze. Sobald B3 ein Token ausgibt, ist in M3 erneut eine Stufe frei und die Aufnahme eines weiteren Token möglich. Falls in dieser Situation weitere Token am Eingang von B3 (in N_{B3} auf den Plätzen ReqB2 und DataB2) anliegen, wird die Local-Clock Phase in B3 unterbrochen.

Wie bereits in Kapitel 2 erläutert, geht B3, nachdem innerhalb der vorgesehenen Zeitspanne von 50 ns kein Requestsignal von B2 anliegt, automatisch in die Time-out Phase über. In dieser Phase wird der lokale Takt in das LS-Modul geleitet, um mit dessen Hilfe die aufgenommenen Daten zu verarbeiten und auszugeben. Die Entscheidung, ob nach einem Handshake die Time-out Phase beginnt oder nicht, modellieren wir mit Hilfe der Transitionen t_3 und t_4 . Nachdem im Schaltkreis ein Handshake am Eingang von B3 abgeschlossen ist (im Netz N_{B3} der Platz p_2 markiert ist), ist der Beginn der Time-out Phase möglich (t_4 ist konzessioniert). So entsprechen wir der Schaltung, in der B3 nach jedem

Handshake potentiell in die Time-out Phase wechseln kann. Sind jedoch in der synchronen Queue noch Stufen frei (ist im Modell der Platz **Pipegrenze** markiert) und kündigt N_{B2} innerhalb von 50 ns einen weiteren Handshake an (ist der Schnittstellenplatz **ReqB2** markiert), so ist auch t_3 aktiviert. Die Transitionen t_3 und t_4 stehen unter diesen Bedingungen in Konflikt um eine Marke. Beide Transitionen unterliegen einem Schaltzwang, auf Grund des kleineren Zeitinveralls schaltet jedoch t_3 , wenn N_{B2} in der vorgegebenen Zeit weitere Marken produziert. Die Transition t_3 entzieht t_4 die Konzession und vollzieht den Handshake mit N_{B2} : Der Block B2 befindet sich weiterhin in der Request-getriebenen Phase. Sollte nach einem Handshake binnen 50 Zeiteinheiten keine neue Marke auf **ReqB2** liegen, schaltet t_4 und die Time-out Phase beginnt. Der Schaltkreis führt dann vier bis sechs Taktzyklen lang die entsprechenden Umschaltprozesse durch. Da die im Block B3 eingebettete Pausable Clock auf eine Taktrate von 80 MHz eingestellt ist, beträgt ein Taktzyklus in B3 12,5 ns, die Time-out Phase dauert also mindestens 50 und höchstens 75 ns an ($4 \cdot 12,5$ ns bis $6 \cdot 12,5$ ns). Da jedoch B2 bereits alle 50 ns Token an B3 abgibt, fallen hier Beginn der Time-out Phase und Aufnahme eines Tokens zusammen ($eft(t_4) = 50$ in N_{B3} und $eft(t_5) = lft(t_5) = 50$ in N_{B2}). Da die Time-out Phase jedoch erst beginnt, nachdem vier Taktzyklen nach der letzten Aufnahme eines Tokens vergangen sind, setzen wir $I(t_4) = [50 + \varepsilon; 75]$ mit $\varepsilon > 0$ in N_{B3} ³.

Nach Abschluss der Time-out Phase beginnt die Local-Clock Phase, in der B3 Daten an B4 weitergibt. Abbildung 2.2 zeigt, dass die Local-Clock Phase eines GALS-Blocks unterbrochen werden kann, sollten in dieser Zeit Daten an dessen Eingang anliegen. Sobald während der Ausgabe ein Requestsignal am Eingang des Wrappers anliegt und noch freie Stufen in der synchronen Pipeline vorhanden sind, geht der Schaltkreis nach Beenden des aktuellen Handshakes in die Request-gesteuerte Phase über. Die Ausgabebetätigkeit wird unterbrochen, der lokale Takt pausiert, und B3 nimmt weitere Token von B2 entgegen. Eine Möglichkeit den Schaltkreis auf diese Art zu unterbrechen, modelliert die Transition t_5 . Diese Transition ist aktiviert, nachdem in N_{B3} die Time-out Phase beendet ist (t_4 hat geschaltet) und noch kein Handshake am Ausgang von B3 stattgefunden hat. Sollte in der Pipeline eine Stufe frei sein (ist der Platz **Pipegrenze** markiert) und sollte zudem ein weiteres Request am Eingang anliegen (ist der Schnittstellenplatz **ReqB2** markiert) so schaltet t_5 sofort und entzieht t_6 die Konzession.

Die Transitionen t_6 , t_7 und t_8 modellieren die Datenausgabe von B3. Ein Handshake ist abgeschlossen, wenn t_7 geschaltet hat. Dann wird an dieser Stelle entweder ein weiterer Handshake mit dem Schalten von t_8 eingeleitet oder der Ausgabefluss wird erneut un-

³Im Rahmen der Analyse des Modells ist ε abhängig vom verwendeten Verifikationswerkzeug zu wählen (z.B. $\varepsilon = 0,1$, falls 0,1 die höchste Zeitaufösung ist).

terbrochen: Sind in N_{B3} die Plätze `ReqB2` und `Pipegrenze` markiert, so entzieht t_9 der Transition t_8 die Konzession und das Petrinetz konsumiert wieder Marken von N_{B2} .

Entsprechend der lokalen Taktrate von 80 MHz, gibt B3 ein Symbol binnen 600 ns an B4 weiter ($48 \cdot 12,5$ ns). Pro Handshake entsteht eine Verzögerung von 12,5 ns. Dementsprechend sind die Transitionen t_6 und t_8 mit dem Zeitintervall $eft = lft = 12,5$ ns versehen.

Der Block B3 geht in den Ruhezustand über, sobald keine Daten mehr aufzunehmen oder abzugeben sind. Im Petrinetz schaltet an dieser Stelle t_{10} , woraufhin die Transition t_1 bereit ist, weitere Handshakes zu vollziehen. So können jederzeit neue Marken von N_{B2} aufgenommen werden. Um zu gewährleisten, dass t_{10} nur dann schaltet, wenn die synchrone Queue leer ist, hat t_{10} nur dann Konzession, wenn es möglich ist, 48 Marken von dem Platz `Pipegrenze` zu konsumieren und zu produzieren. Das Modell entspricht dem Schaltkreis, da die Queue in M3 nur dann zurückgesetzt wird, wenn sie tatsächlich leer ist.

3.2.6. Das Netz N_{B4}

Abbildung 3.9 stellt das Modell des Blocks B4 dar. Das zeitliche Verhalten während der Aufnahme eines Symbols ist (analog zu den anderen Blöcken) für B4 durch dessen Vorgänger B3 bestimmt. Demzufolge schalten im Netz N_{B4} die Transitionen t_1 und t_2 sofort, wenn die Schnittstellenplätze `ReqB3` und `DataB3` markiert sind. Block B4 benötigt insgesamt 600 ns für die Aufnahme eines gesamten Symbols ($48 \cdot 12,5$ ns).

Um Symbol i verarbeiten zu können, muss B4 das Symbol $i + 1$ eines Frames vollständig empfangen haben. Darum werden im Petrinetz N_{B4} Marken zunächst auf einem Pufferplatz (p_2) gesammelt. Sobald p_2 48 Marken trägt, schaltet t_3 und N_{B4} hat das Symbol i aufgenommen. Die Transition t_4 modelliert die Time-out Phase von B4. Diese Transition hat erst dann Konzession, wenn das Symbol $i + 1$ vollständig auf dem Platz p_2 und das Symbol i auf dem Platz p_4 verfügbar ist. In diesem Fall, tauscht t_4 Symbol i gegen Symbol $i + 1$ aus, produziert gleichzeitig Symbol i auf den Platz p_5 und lässt dabei die Zeit vergehen, die die Time-out Phase in Block B4 andauert ($4 \cdot 12,5$ ns bis $6 \cdot 12,5$ ns). Sobald t_5 aktiviert ist, vergeht die Zeit, die Block B4 benötigt, um ein Symbol zu verarbeiten: $I(t_5) = [6000; 6000]$. Im Schaltkreis des Receivers befinden sich während der Verarbeitung eines Frames stets maximal fünf Symbole. Die letzten fünf Symbole eines Frames werden nicht durch die Asynchrone Queue zurückgeführt, sondern lediglich zum Ausgang des Receivers transportiert. Dementsprechend modelliert die Transition t_5 die Berechnung der ersten $R - 5$ Symbole eines Frames, t_{11} modelliert die Berechnung der

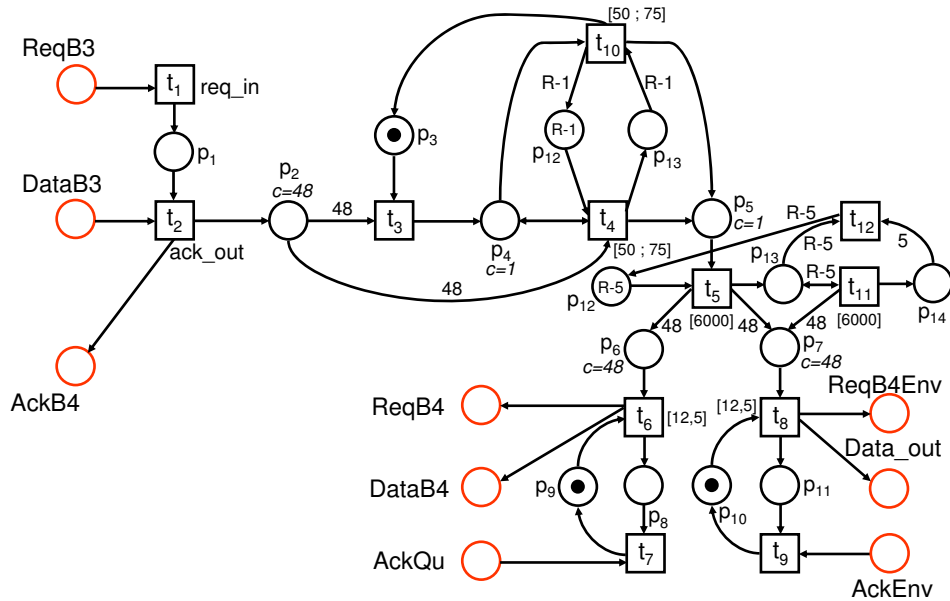


Abbildung 3.9.: Das Netz N_{B4} modelliert den GAL S-Block B4.

letzten fünf Symbole. Die Transition t_{11} gibt diese Symbole anschließend direkt an die Transition t_8 weiter, das heißt die Marken werden nicht zurückgeführt.

Das letzte, das heißt, das R -te Symbol eines Frames verarbeitet Block B4 direkt, ohne auf das nächste Symbol zu warten. Im Petrinetz modelliert die Transition t_{10} die Aufnahme des letzten Symbols. Die Transition t_{10} ist aktiviert, sobald das letzte Symbol auf dem Platz p_4 liegt, da Transition t_4 zu diesem Zeitpunkt alle $R - 1$ Marken von p_{12} nach p_{13} produziert hat. Während der Berechnung eines Frames ist die Transition t_{10} lediglich einmal aktiviert. Innerhalb der für die Time-out Phase vorgesehenen Zeit schaltet t_{10} , konsumiert das R -te Symbol und produziert es zur Verarbeitung auf den Vorplatz von t_5 .

Wie in Kapitel 2.3 beschrieben, wird der Ausgabestrom in B4 zweigeteilt: Die Transitionen t_6 und t_7 modellieren die Datenausgabe an die Asynchrone Queue, die Transitionen t_8 und t_9 bilden die Datenausgabe an die Schnittstelle des Receivers ab. Um einer Ausgabefrequenz von 80 MHz zu entsprechen, sind die Transitionen t_6 und t_8 mit dem Zeitintervall $[12,5; 12,5]$ versehen. Nachdem das letzte Symbol eines Frames ausgegeben ist, setzt die Transition t_{12} das Netz N_{B4} in die Anfangsmarkierung zurück.

3.2.7. Das Netz N_{aQu}

Die Asynchrone Queue realisiert im Receiver den asynchronen und unabhängigen Rückwärtsfluss der Daten von B4 nach B2. Die Struktur besteht aus 48 Stufen, die mittels Handshakes kommunizieren. Dabei entsteht im Schaltkreis pro Stufe eine Verzögerung von 3 bis 5 Nanosekunden für jedes Token. Das entsprechende Petrinetz N_{aQu} ist in Abb. 3.10 dargestellt. Analog zu den Schnittstellen der Petrinetze N_{B2} , N_{B3} und N_{B4} unterscheiden wir auch für das Netz N_{aQu} zwischen Kontroll- und Datenfluss. Liegt am Eingang der Queue ein Requestsignal an, so ist im Petrinetz der Platz $ReqB4$ markiert. Sobald die von B4 bereitgestellten Daten gültig sind, nimmt die Asynchrone Queue diese auf (Transition add schaltet) und bestätigt diesen Vorgang (Marke auf Platz $AckQu$). Alle Transitionen t dieses Netzes modellieren die jeweiligen Handshakeaktivitäten pro Stufe, 48 der Transitionen sind mit den Intervallgrenzen $I(t) = [3; 5]$ versehen.

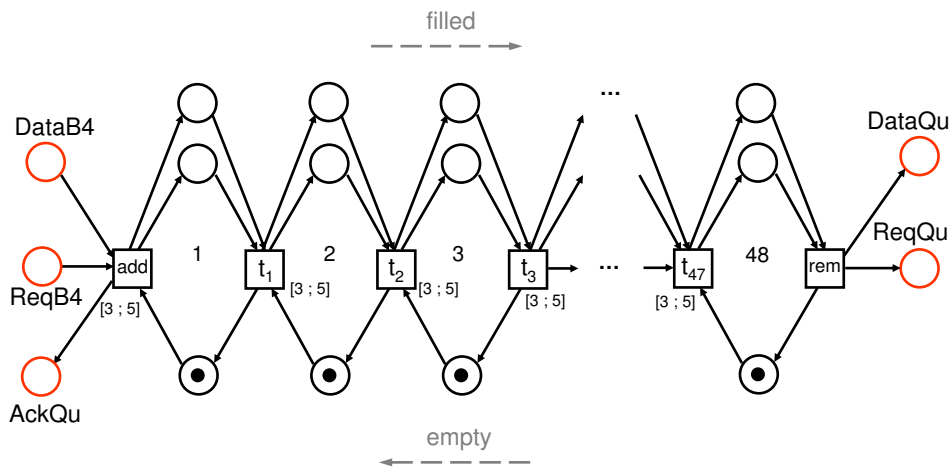


Abbildung 3.10.: Das Netz N_{aQu} modelliert die Asynchrone Queue.

3.3. Verifikationsaufgabe am Modell

Wie in Kapitel 2.3 beschrieben, ist es nicht klar, ob der Schaltkreis unter den gegebenen Bedingungen jederzeit korrekt funktioniert. Alle in diesem Kapitel vorgestellten Netze wurden in Zusammenarbeit mit den Entwicklern des Schaltkreises validiert. Das heißt, das Modell bildet das Verhalten des Schaltkreises adäquat ab. Wir können somit im Folgenden die in Kapitel 2.4 aufgezählten vier Fragestellungen auf Verifikationsaufgaben

abbilden, die am Petrinetzmodell untersucht werden. Konkret bilden wir die zu untersuchenden nicht-funktionalen Eigenschaften auf Petrinetz-Eigenschaften (beispielsweise die Erreichbarkeit von Markierungen und die Lebendigkeit von Transitionen) ab. Erweisen sich unsere Annahmen als korrekt, ist die Schaltung bezüglich der in dieser Arbeit beschriebenen Eigenschaften fehlerfrei.

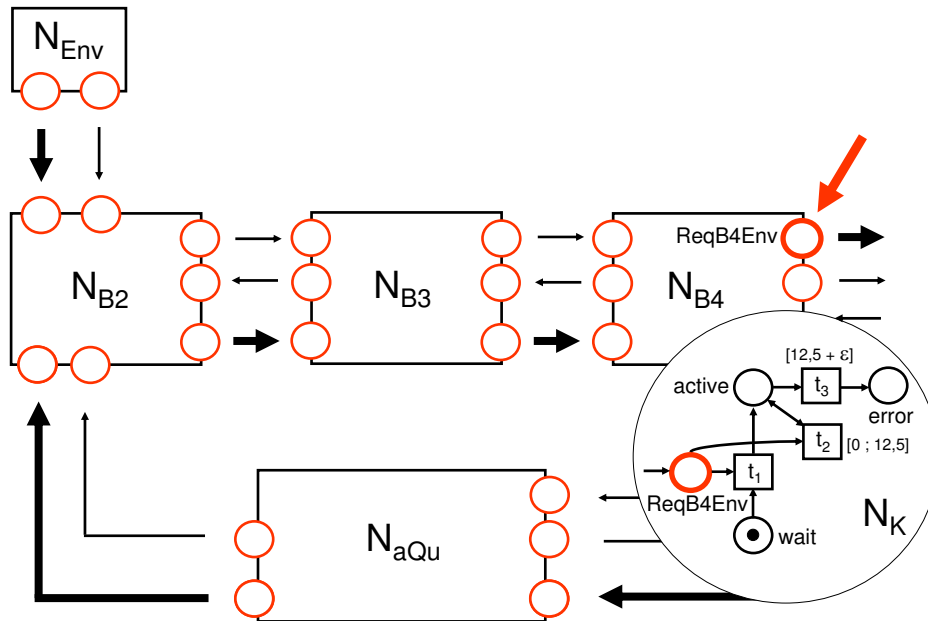
1. Ist es möglich, dass der Receiver nach einer gewissen Initialisierungsphase Daten konstant mit 80 MHz zur Verfügung stellt, das heißt B4 Daten mit 80 MHz ausgibt?

Wir untersuchen das Problem am Ausgang des Netzes N_{B4} . Hier wirken sich die möglichen Verzögerungen der Rückführung aus. Sollten für B2 Verzögerungen entstehen ist es letztendlich für B4 nicht möglich, Token mit einer Rate von 80 MHz zur Verfügung zu stellen und die Datenübertragung wird unterbrochen.

Die von uns gewählte Herangehensweise an das Problem besteht darin, N_{B4} um ein zusätzliches Petrinetz zu erweitern, das wir *Konsument* (N_K in Abb. 3.11) nennen. Wie in Abb. 3.11 dargestellt, verschmelzen wir den Schnittstellenplatz *DataB4* mit dem Platz *DataB4* des Konsumenten-Netzes. Sobald N_K Marken von N_{B4} konsumiert, sind die Transitionen t_2 und t_3 in Abb. 3.11 aktiviert (t_1 hat geschaltet und der Platz *active* ist markiert). Produziert dann das Netz N_{B4} binnen 12,5 Zeiteinheiten eine weitere Marke auf den Platz *DataB4*, schaltet Transition t_2 , sonst Transition t_3 . Sollte jemals t_3 in einem Ablauf schalten, bedeutet dies, dass es im Verhalten der Schaltung einen Ablauf gibt, in dem der Tokenfluss zu langsam erfolgt. Existiert eine erreichbare Markierung m mit $m(\text{error}) > 0$, so ist belegt, dass die Schaltung die vorgegebene zeitlichen Bedingungen verletzt. Alternativ ist es möglich, die Lebendigkeit von t_3 zu untersuchen: Wenn es keinen Ablauf gibt, in dem t_3 schaltet, so erfüllt der Schaltkreis die vorausgesetzten Kriterien.

2. Ist es möglich, den Receiver mit einer niedrigeren Taktrate zu betreiben?

Um dieses Problem zu untersuchen, vergrößern wir in den Petrinetzen N_{B3} und N_{B4} die Zeitintervalle an den Transitionen, die die Taktrate modellieren. Dies sind für N_{B3} die Transitionen t_4 (hier ist die Dauer der Time-out Phase modelliert), sowie t_6 und t_7 in Abb. 3.8 (diese Transitionen modellieren die Dauer des Ausgabestroms). Für N_{B4} sind es die Transitionen t_4 und t_{10} (Dauer der Time-out Phase), t_5 (hier vergeht die Zeit, in der die Daten berechnet werden) und t_6 und t_8 (Dauer des Ausgabestroms) in Abb. 3.9. Vergrößern wir die Zeitintervalle dieser Transitionen, entsprechen wir damit einer langsameren Reaktionszeit des Schaltkreises. Wir modellieren so den Schaltkreis mit einer beliebigen niedrigeren Taktrate. Um zu untersuchen, ob das gesamte Petrinetz daraufhin

Abbildung 3.11.: Der Konsument N_K am Ausgang von N_{B4} .

den Ansprüchen an die Schaltung gerecht wird, fügen wir wieder das aus der ersten Fragestellung bekannte Netz N_K an den Ausgang des Netzes N_{B4} an, passen die Zeitintervalle der Transitionen t_2 und t_3 in N_K an und fragen erneut, ob der Platz **error** jemals markiert wird. Ist dies der Fall, so erfüllt das System die vorausgesetzten Kriterien trotz verminderter Taktrate und kann dementsprechend mit weniger Energie betrieben werden.

3. Ist der Receiver tolerant gegenüber Verzögerungen des Datenflusses in der Asynchronen Queue?

Das exakte zeitliche Verhalten der Asynchronen Queue abzuschätzen ist sehr schwer, da die Verzögerung je nach aktueller Füllung der Queue variieren kann. Um zu untersuchen, ob die Latenz des Rückflusses der Daten in jedem Ablauf niedrig genug ist, um das korrekte Übertragungsverhalten des Receivers zu gewährleisten, betrachten wir den Ein- und Ausgang der Queue und ermitteln ob es möglich ist, alle 4000 ns ein Symbol aufzunehmen bzw. auszugeben. Dementsprechend unterteilen wir die 3. Frage in die Fragen 3.1 und 3.2.

3.1 Ist es möglich, dass am Eingang der Asynchronen Queue alle Token rechtzeitig angenommen werden können?

Nachdem der Receiver die ersten vier Symbole eines Frames aufgenommen und Block B4 die Berechnung des ersten Symbols dieses Frames vollzogen hat, nimmt die Asynchrone Queue Token auf. Wir wollen hier konkret ermitteln, ob der Schaltkreis gewährleistet, dass die Queue ab diesem Moment alle 4000 ns 48 Token aufnimmt. Um diese Frage untersuchen zu können, stehen in diesem Kapitel nicht ausreichend Informationen zur Verfügung. In Kapitel 4 dieser Arbeit verkleinern wir das Modell des Receivers, wobei sich mehr Analysemöglichkeiten für das zu untersuchende Modell ergeben, als in diesem Kapitel vorgestellt. Darum verweisen wir hier auf Kapitel 4, in dem wir näher auf die Analyse der Fragestellung 3.1 eingehen.

3.2 Steht am Ausgang der Asynchronen Queue stets ein Token für B2 zur Verfügung, wenn B2 dies zur Verarbeitung von Daten benötigt?

Hier verwenden wir wieder das Netz N_K der 1. Frage. Wir verschmelzen den Platz `DataQu` der Asynchronen Queue (aus Abb. 3.10) mit dem Platz `Data` des Konsumenten N_K und passen die Zeitintervalle der Transitionen t_2 und t_3 dahingehend an, dass sie der Aufnahme Frequenz des Netzes N_{B2} entsprechen. Anschließend überprüfen wir erneut, ob der Platz `error` jemals markiert wird. Ist dies nicht der Fall, so entspricht die Größe der Asynchronen Queue den zeitlichen Ansprüchen an die Schaltung und das System erfüllt die vorausgesetzten Kriterien. Ist die Markierung des Platzes `error` in einem Ablauf erreichbar, muss gegebenenfalls die Anzahl der FIFO-Stufen reduziert werden.

4. Ist es möglich, den Receiver zu betreiben, wenn die Asynchrone Queue weniger Stufen enthält?

Wieder betrachten wir dazu Ein- und Ausgang der Asynchronen Queue. Konkret stellen wir die Fragen 3.1 und 3.2, reduzieren jedoch die Anzahl der FIFO-Stufen. Können die Fragen für eine kleinere Queue noch immer positiv beantwortet werden, ist dies der Nachweis für eine legitime Reduktion der Chipfläche an dieser Stelle. Während der Analyse vermindern wir die Anzahl der Stufen sukzessive bis ein Wert erreicht ist, für den die Fragestellungen 3.1 und 3.2 nicht mehr positiv beantwortet werden können.

Um nicht-funktionale Eigenschaften wie Latenz, Datendurchsatz, Toleranzgrenzen bezüglich der entstehenden Verzögerungen und des Flächenverbrauchs einer Schaltung zu analysieren, ist es sinnvoll, das gegebene System auf die oben beschriebene Art zu analy-

sieren. Wir reduzieren die schwierigen und sehr technischen Fragestellungen aus Kapitel 2 auf die Erreichbarkeit einer Teilmarkierung im Petrinetzmodell und fragen beispielsweise nach der Erreichbarkeit des Platzes `error` in Abb. 3.11.

Um die Fragen 1 bis 4 auf die oben beschriebene Art zu analysieren, muss ein Verifikationswerkzeug hinzugezogen werden. Dieses berechnet den Erreichbarkeitsgraph der gegebenen Petrinetze und untersucht währenddessen die zu analysierenden Eigenschaften. Für die Widerlegung einer geforderten Eigenschaft gibt ein solches Werkzeug einen Gegenbeispielpfad aus, mit dem es belegt, dass ein Ablauf existiert, der die zu überprüfende Eigenschaft nicht erfüllt. Dieser Ablauf wird in eine entsprechende Signalsequenz des Schaltkreises zurückübersetzt und im konkreten System überprüft. Bestätigt sich auf diese Weise, dass die geforderte Eigenschaft nicht erfüllt ist, werden alternative Strukturen für die Schaltung gesucht. Basierend auf den Ergebnissen der Analyse wird die Schaltung dann neu überdacht (neue zeitliche Einschränkungen werden eingefügt, logische Verknüpfungen von Gattern werden anders strukturiert), bis die Schaltung dem gewünschten Verhalten entspricht.

Für die Analyse des hier beschriebenen Modells ergibt sich allerdings ein zu großer Zustandsraum, das heißt, der Erreichbarkeitsgraph kann für dieses Modell nicht vollständig berechnet werden. Dies liegt vor allem an den großen Beträgen in den Zeitintervallen, an der nebenläufigen Abarbeitung der Prozesse im Modell, an der Anzahl der Plätze und Transitionen und der großen Anfangsmarkierung (siehe Kapitel 4.1). Es ist jedoch möglich, diese Größen zu reduzieren, ohne dabei die für die Analyse relevanten Eigenschaften zu verlieren. In Kapitel 4 gehen wir auf diese und weitere Reduktionen ein und zeigen deren Ergebnisse auf.

4. Abstraktion des Modells

In Kapitel 3 haben wir ein formales Modell des zu untersuchenden Schaltkreises aus Kapitel 2 vorgestellt. Dieses gibt Aufschluss über erste Analysemöglichkeiten der nicht-funktionalen Eigenschaften der Schaltung. Wir werden in Kapitel 4.1 zeigen, dass der Zustandsraum des vorliegenden Modells für eine Verifikation zu groß ist um die Fragen 1 bis 4 auf die in Kapitel 3.3 beschriebene Art mit bestehenden Werkzeugen zu beantworten.

Anschließend diskutieren wir in diesem Kapitel mehrere Möglichkeiten, das Modell des Receivers zu abstrahieren. Dazu wenden wir in Kapitel 4.2 zunächst verschiedene strukturelle Reduktionsregeln auf das Modell an. Weiterhin fassen wir in Kapitel 4.3 Strukturen der Netze zusammen, wobei wir zusätzliches Wissen über die Schaltung in die Reduktion einfließen lassen. Insbesondere abstrahieren wir an dieser Stelle von der konkreten Belegung der Stufen der Asynchronen Queue, die mit dem Netz N_{aQu} modelliert ist. In Kapitel 4.4 abstrahieren wir weiterhin von dem Verhalten, das für die Analyse der gefragten Eigenschaft nicht notwendig ist und zeigen, inwiefern die zu untersuchende Eigenschaft durch die Abstraktion bewahrt wird. Anschließend werten wir unsere Abstraktionsergebnisse aus.

4.1. Abschätzung des Zustandsraums

In Kapitel 3.2 ist es uns gelungen, den Schaltkreis des Receivers adäquat auf ein formales Modell abzubilden. Das Modell beschreibt die Prozesse und das zeitliche Verhalten der Schaltung und verdeutlicht, welche GALS-Blöcke Daten nebenläufig und sequentiell abarbeiten. Des Weiteren haben wir mögliche Analyseschritte vorgestellt, mit denen die nicht-funktionalen Eigenschaften der Schaltung überprüft werden können. Trotz der Beschränktheit der vorliegende Netze ist es nicht möglich, die 1. bis 4. Frage auf die in Kapitel 3.3 beschriebene Art mit bestehenden Werkzeugen zu beantworten, da der Zustandsraum der Komposition der in Kapitel 3.2 vorgestellten Intervall-Petrinetze für eine Analyse zu groß ist. Wir definieren diese Komposition als das Netz NR_1 , welches aus allen Netzen besteht, die die einzelnen GALS-Blöcke des Receivers modellieren.

Definition 17 (Netz NR_1)

Sei $NR_1 = (P, T, F, V, m_0, I)$ das Intervall-Petrinetz, das aus der Komposition der Netze in Abbildung 3.3 entsteht, das heißt $NR_1 = N_{Env} \oplus N_{B2} \oplus N_{B3} \oplus N_{B4} \oplus N_{aQu}$. \lrcorner

Die Komposition NR_1 ist geschlossen, das heißt sie hat keine Schnittstellenplätze. NR_1 besteht aus 84 Transitionen und 205 Plätzen. Mit dem Verifikationswerkzeug INA [Sta92] konnte der Erreichbarkeitsgraph des Netzes NR_1 nicht vollständig berechnet werden: Das Werkzeug brach die Berechnungen für eine Anfangsmarkierung von 560 Marken (entspricht 7 Symbolen) nach ca. einer Millionen Zustände wegen Speicherüberlaufs ab. Allerdings implementiert INA keine Reduktionstechniken für Intervall-Petrinetze.

Wir konnten jedoch die Skelette der Teilnetze von NR_1 mit dem Verifikationswerkzeug LoLA [Sch00] auf Verklemmungen überprüfen, da dieses Werkzeug Reduktionstechniken für Stellen-Transitions-Netze implementiert. Obwohl LoLA die Zeiteinheiten eines Modells nicht berücksichtigt, ist die Untersuchung von Verklemmungen mit diesem Werkzeug sinnvoll. Falls das Skelett einen Deadlock enthält, so auch das entsprechende Intervall-Petrinetz. Die Analyse ergab, dass der einzige Deadlock des Skeletts der erwünschte Endzustand ist. Allerdings brach auch LoLA den Aufbau des Erreichbarkeitsgraphen der Komposition NR_1 nach ca. 20 Millionen Zuständen wegen Speicherüberlaufs ab. Der von LoLA generierte Erreichbarkeitsgraph enthält jedoch auch Schaltsequenzen, die durch den Schaltzwang (Zeit impliziert eine Schaltreihenfolge) im Intervall-Petrinetz nicht möglich sind. Allerdings besteht ein Zustand im Intervallnetz nicht nur aus einer Markierung, sondern auch aus einer Transitionsmarkierung (siehe Definition 13). Somit hat der Erreichbarkeitsgraph eines Intervall-Petrinetzes im Vergleich zu seinem Skelett zwar in der Regel weniger erreichbare Markierungen, jedoch umso mehr Zwischenzustände, die sich durch die jeweiligen Transitionsmarkierung unterscheiden.

Weiterhin ausschlaggebend für die Größe des Zustandsraums ist neben der Zeit vor allem die Nebenläufigkeit im Modell. Gravierende Auswirkungen hat zudem die Größe der Anfangsmarkierung m_0 des Modells. Diese beträgt 80 Marken pro Symbol eines Frames, wobei ein Frame aus R Symbolen besteht. Mit $R \geq 100$ ergibt sich eine Anfangsmarkierung von mindestens 8000 Marken auf den Schnittstellenplätzen von N_{Env} . Für R Durchläufe mit jeweils 80 Token pro Symbol ist es unter der Berücksichtigung der Zeit für gängige Verifikationswerkzeuge nicht möglich, den Zustandsraum für die Komposition der vorliegenden Netze zu berechnen.

Um die Zustandsexplosion abzuschwächen, stellen wir in diesem Kapitel verschiedene Abstraktionstechniken vor. Unter Abstraktion verstehen wir die Zusammenfassung von Übergängen im Modell, die eine zu untersuchende Eigenschaft E nicht beeinflussen. Im Folgenden bezeichnen wir für jeden Abstraktionsschritt das Ausgangsnetz als *konkretes*

Modell, das abstrahierte Netz als *abstraktes* Modell. Unsere Zielsetzung ist die Verminderung der Zustandsanzahl des zu untersuchenden Erreichbarkeitsgraphen unter gleichzeitiger Bewahrung der zu verifizierenden Eigenschaft. Dabei unterscheiden wir zwischen Abstraktionen, die eine *schwache* und eine *starke* Bewahrung der zu untersuchenden Eigenschaft E realisieren. Von schwacher Bewahrung sprechen wir, wenn gilt: ist E gültig im abstrakten Modell, so gilt E ebenfalls im konkreten Modell. Von starker Bewahrung einer Eigenschaft E sprechen wir, wenn zusätzlich gilt: ist E im abstrakten Modell ungültig, so auch im konkreten Modell.

Das Netz NR_1 ist Ausgangspunkt für alle Reduktionen und Abstraktionen, die wir im Folgenden vornehmen. Tabelle 4.1 am Ende dieses Kapitels enthält die jeweilige Platz-, Transitions- und (wenn möglich) Zustandsanzahl der im Folgenden vorgestellten Netze.

Wir führen zunächst eine schrittweise strukturelle Reduktion des Netzes NR_1 durch und zeigen für jede Reduktionsregel, dass es noch immer möglich ist, eine relevante Eigenschaft (z.B. Lebendigkeit) im reduzierten Netz zu untersuchen, wenn dies für das unreduzierte Netz möglich ist. Anschließend nehmen wir eine schrittweise Abstraktion des resultierenden Netzes vor, wobei wir für jeden Abstraktionsschritt zeigen, ob die zu untersuchende Eigenschaft stark oder schwach bewahrt wird.

4.2. Strukturelle Reduktion

In diesem Kapitel stellen wir einige Regeln vor, die es ermöglichen, die Struktur des Netzes NR_1 unter Bewahrung der zu untersuchenden Eigenschaft zu verkleinern. Starke definiert in [Sta90] strukturelle Reduktionsregeln für Stellen-Transitions-Netze. Im Folgenden passen wir diese Regeln für Intervall-Petrinetze an und beweisen die Korrektheit unseres Vorgehens. Anschließend wenden wir die Regeln auf das Netz NR_1 an.

4.2.1. Strukturelle Reduktion (Reduktionsregel 1)

Um die Reduktionsregeln für Stellen-Transitions-Netze an die Semantik von Intervall-Petrinetzen anzupassen, definieren wir zunächst einige Begriffe.

Definition 18 (Reduktionsschritt)

Wir nennen das einmaligen Anwenden einer Reduktionsregel auf das Intervall-Petrinetz $N = (P, T, F, V, m_0, I)$ einen Reduktionsschritt. Das Ergebnis bezeichnen wir stets mit $N' = (P', T', F', V', m'_0, I')$. ┘

Beim Reduktionsschritt von N nach N' ist die Netzeigenschaft E stark bewahrt, wenn gilt: das Netz N' erfüllt die Eigenschaft E genau dann, wenn N die Eigenschaft E erfüllt. Die für uns interessante Eigenschaft ist Lebendigkeit und damit einhergehend das Zeitverhalten des strukturell reduzierten Netzes. Lebendigkeit ist für Intervall-Petrinetze über deren Zustände definiert, dies schließt Markierung und Transitionsmarkierung ein. Bleibt diese Eigenschaft durch den Reduktionsschritt stark bewahrt, so gibt es in N zwar Markierungen, die in N' nicht erreichbar sind. Da jedoch der Platz **error** des Netzes N_K in Abb. 3.11 durch die strukturelle Reduktion nicht betroffen sein wird, gilt die starke Bewahrung bezüglich der Erreichbarkeit aller Markierungen, die den Platz **error** überdecken.

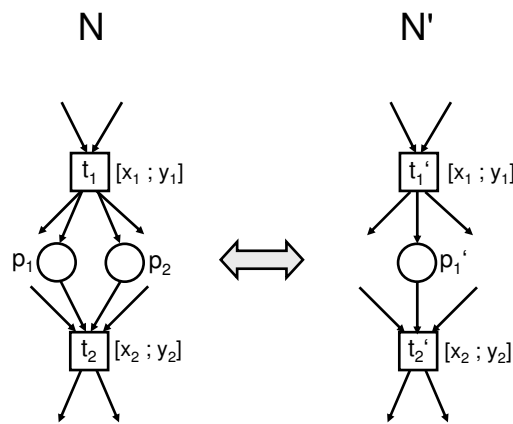


Abbildung 4.1.: Strukturelle Reduktionsregel für Intervall-Petrinetze: Reduktionsregel 1.

In Abb. 4.1 ist die erste von uns verwendete Reduktionsregel (*Reduktionsregel 1*) veranschaulicht. Die Plätze p_1 und p_2 im Netz N sind *parallel*, das heißt, sie haben die gleichen Vor- und Nachbereich und sind mit der gleichen Vielfachheit verbunden. Die Reduktionsregel 1 erlaubt, einen dieser Plätze zu streichen.

Definition 19 (Parallele Plätze)

Zwei Plätze p_1 und p_2 sind parallel im Netz N , wenn gilt:

1. $\bullet p_1 = \bullet p_2$,
2. $p_1 \bullet = p_2 \bullet$,
3. $\forall t \in \bullet p_1 : V([t, p_1]) = V([t, p_2])$,
4. $\forall t \in p_1 \bullet : V([p_1, t]) = V([p_2, t])$ und

5. $m_0(p_1) = m_0(p_2) = 0$.

┘

Reduktionsregel 1: Streichen paralleler Plätze

Voraussetzung: Seien p_1 und p_2 parallele Plätze im Netz N . Sei $I(t)$ mit $t \in \bullet p_1 \cup p_1 \bullet$ beliebig.

Anwendung: Für die Plätze p_1 und p_2 führen wir im Netz N' einen neuen Platz p'_1 ein mit

1. $\bullet p'_1 = \bullet p_1$,
2. $p'_1 \bullet = p_1 \bullet$ und
3. $m_0(p'_1) = m_0(p_1) + m_0(p_2)$.

Anschließend streichen wir die Plätze p_1 und p_2 und ihre ein- und ausgehenden Kanten im Netz N' .

Ein Kriterium für die Bewahrung einer Eigenschaft nach Anwendung einer Reduktionsregel ist die Konsistenz einer Reduktionsregel. Starke hat in [Sta90] bereits für alle in dieser Arbeit aufgeführten Reduktionsregeln gezeigt, dass sie in Stellen-Transitions-Netzen Lebendigkeit und Beschränktheit stark bewahren. Wir beschränken uns darauf zu zeigen, dass das Anwenden der Reduktionsregeln Lebendigkeit für Intervall-Petrinetze stark bewahrt.

Definition 20 (Konsistente Reduktionsregel)

Eine Reduktionsregel heißt konsistent, wenn bei ihrer Anwendung auf ein beliebiges Intervall-Petrinetz N Lebendigkeit des Netzes bewahrt bleiben, das heißt: N ist lebendig genau dann, wenn N' lebendig ist.

┘

Satz 4.1 *Die Reduktionsregel 1 ist konsistent.*

Beweis. Es genügt zu zeigen, dass die Reduktionsregel 1 das Zeitverhalten von N in N' stark bewahrt.

Sei t_1 eine Transition aus $\bullet p_1$ und t_2 eine Transition aus $p_1 \bullet$. Sei t_1 zum Zeitpunkt $h(t_1) = x$ aktiviert. Dann ist auch t'_1 in N' zum Zeitpunkt $h(t'_1) = x$ aktiviert. In N schaltet t_1 zum Zeitpunkt $h(t_1)$ mit $x_1 \leq h(t_1) \leq y_1$, in N' schaltet t'_1 zum Zeitpunkt $h(t'_1)$ mit $x_1 \leq h(t'_1) \leq y_1$. Daraufhin ist t_2 genau dann in N konzessioniert, wenn t'_2

in N' konzessioniert ist, da $I(t_2) = I'(t'_2) = x_2$. Offensichtlich ist die Reduktionsregel 1 konsistent. \square

Um das konkrete Modell strukturell zu reduzieren, eliminieren wir im Folgenden alle parallelen Plätze in NR_1 . Die Anwendung der Reduktionsregel 1 bewirkt, dass die Markierungsvektoren des Erreichbarkeitsgraphen kürzer sind und die resultierende Struktur des Netzes die Anwendung weiterer Reduktionsregeln ermöglicht. Exemplarisch stellen wir das strukturell reduzierte Netz N'_{aQu} in Abb. 4.2 dar. Dieses entsteht, indem wir die Reduktionsregel 1 auf alle Plätze anwenden, deren Vor- und Nachbereich gleich ist (siehe Abb. 4.2). So wenden wir beispielsweise an der Schnittstelle zwischen den Netzen N_{B4} und N_{aQu} die Reduktionsregel 1 auf den Platz **DataB4** an und streichen diesen (siehe Abb. 4.7 und Abb. 4.2). Die gestrichelten Kanten und Plätze in Abb. 4.2 berücksichtigen wir während der Analyse nicht.

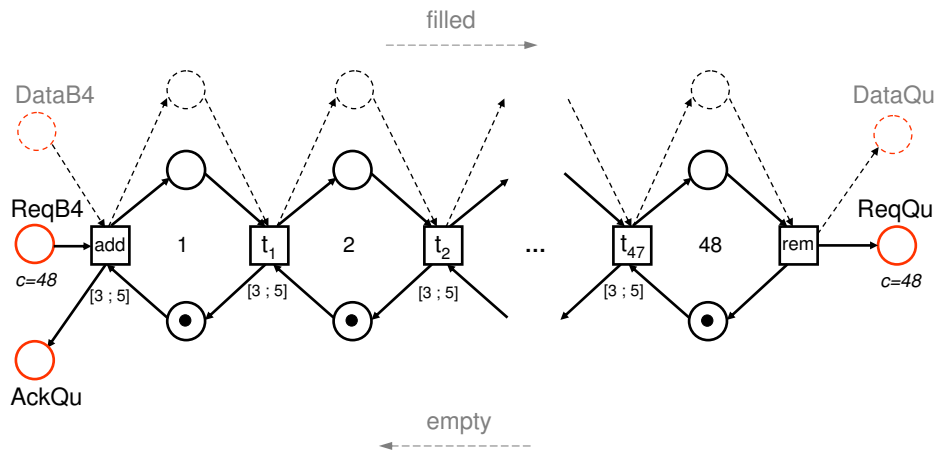


Abbildung 4.2.: Das strukturell reduzierte Netz N'_{aQu} (nach Anwenden der Reduktionsregel 1).

4.2.2. Strukturelle Reduktion (Reduktionsregel 2)

Nach der folgenden Reduktionsregel werden Vor- und Nachtransition eines Platzes p wie in Abb. 4.3 verschmolzen. Wichtige Voraussetzung dabei ist, dass die Transition im Nachbereich von p keine weiteren Vorplätze hat und so lebendig ist, sofern die Transition im Vorbereich von p lebendig ist.

Reduktionsregel 2: Zusammenfassen einer Transitionssequenz

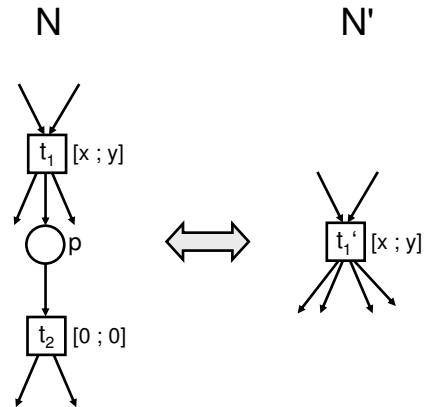


Abbildung 4.3.: Strukturelle Reduktionsregel für Intervall-Petrinetze: Reduktionsregel 2.

Voraussetzung: Sei p ein Platz in N mit

1. $\bullet p = \{t_1\}$ und $p\bullet = \{t_2\}$,
2. $p\bullet \cap \bullet p = \emptyset$,
3. $(p\bullet)\bullet \neq \emptyset$,
4. $\bullet(p\bullet) = \{p\}$,
5. $V([t_1, p]) = V([p, t_2])$.

Anwendung: Für die Transitionen t_1 und t_2 führen wir im Netz N' eine neue Transition t'_1 ein mit

1. $\bullet t'_1 = \bullet t_1$,
2. $t'_1\bullet = (t_1\bullet \setminus \{p\}) \cup (t_2\bullet)$ und
3. $I(t'_1) = I(t_1)$.

Anschließend streichen wir die Transitionen t_1 und t_2 , den Platz p und dessen ein- und ausgehende Kanten im Netz N' .

Satz 4.2 Die Reduktionsregel 2 ist konsistent.

Beweis. Es genügt zu zeigen, dass N und N' das gleiche Zeitverhalten haben. Sei t_1 zum Zeitpunkt $h(t_1) = x$ aktiviert. Dann ist t'_1 in N' zum Zeitpunkt $h(t'_1) = x$ aktiviert. t_1 schaltet in N zum Zeitpunkt $h(t_1)$ mit $x \leq h(t_1) \leq y$ und t'_1 schaltet in N' zum Zeitpunkt

$h(t'_1)$ mit $x \leq h(t'_1) \leq y$. In N ist anschließend t_2 aktiviert, in N' ist der Nachbereich von t'_1 markiert. Zum Zeitpunkt $h(t_2) = 0$ schaltet t_2 in N . Daraufhin ist der Nachbereich von t_2 markiert. Offensichtlich ist der Nachbereich von t_2 in N zum Zeitpunkt $h(t_2) = x + 0$ und der Nachbereich von t'_1 in N zum Zeitpunkt $h(t'_1) = x$ markiert. Die Reduktionsregel 2 ist konsistent. \square

Die Reduktionsregel 2 impliziert ebenfalls das Verschmelzen der Transitionen t_1 und t_2 , wenn gilt: $I(t_1) = [0; 0]$ und $I(t_2) = [0; 0]$. Außerdem gilt die Reduktionsregel 2 für den Fall $I(t_1) = [0; 0]$ und $I(t_2) = [x; y]$. Da der Beweis hierzu analog zu dem oben geführten Beweis ist, verzichten wir hier darauf.

Um das konkrete Modell strukturell zu reduzieren, verschmelzen wir im Folgenden alle Transitionen, die dem Muster der Reduktionsregel 2 entsprechen. Diese Vorgehensweise reduziert den Zustandsraum des zu untersuchenden Netzes, da Zwischenzustände vermieden werden.

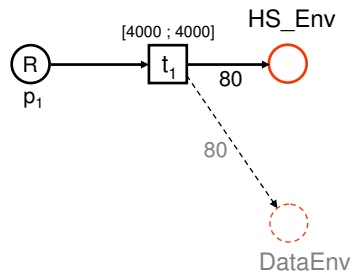


Abbildung 4.4.: Das strukturelle reduzierte Netz N'_{Env} (nach Anwenden der Reduktionsregeln 2 und 1).

Die Abbildungen 4.4 bis 4.5 zeigen die Netze N'_{Env} bis N'_{B2} nach Anwendung der strukturellen Reduktionsregeln 1 und 2 auf die Komposition NR_1 .

Wir wenden zunächst die Reduktionsregel 2 auf die Transitionen t_1 und t_2 in N_{B2} an. Sei die resultierende Transition t_2 mit $I(t_2) = [50; 50]$. Dann wenden wir die Reduktionsregel 1 auf den Platz $DataEnv$ in N_{Env} und N_{B2} an und streichen diesen. Auf die strukturelle Reduktion der Schnittstelle zwischen N_{B2} und dem Netz N_{aQu} gehen wir anschließend ein (siehe Strukturelle Reduktionsregel 3). Das Netz N'_{B2} ist in Abb. 4.5 dargestellt.

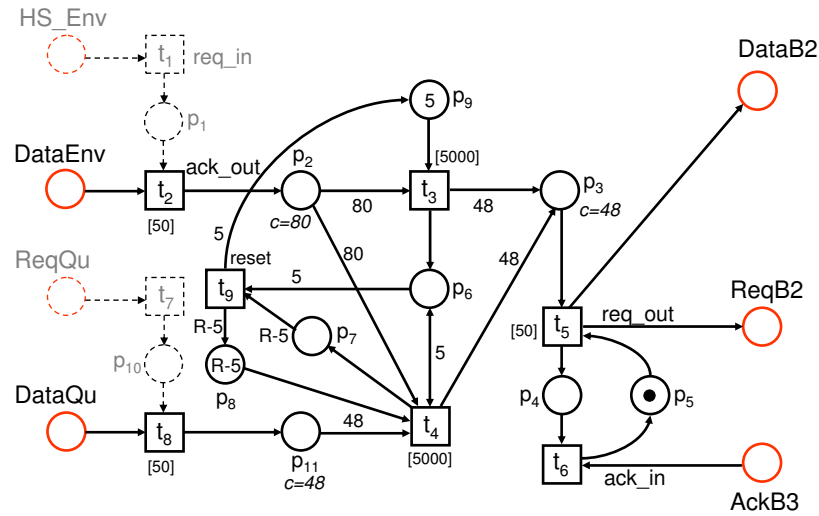


Abbildung 4.5.: Das strukturell reduzierte Netz N'_{B2} (nach Anwenden der Reduktionsregeln 1 und 2).

4.2.3. Strukturelle Reduktion (Reduktionsregel 3)

Ähnlich zu Reduktionsregel 2 sind ebenfalls Transitionen redundant, wenn ihre Vor- und Nachbereiche jeweils aus einelementigen Platzmengen bestehen. Abbildung 4.6 zeigt ein Intervall-Petrinetz N mit Transition t , der ein Zeitintervall von $I(t) = [0; 0]$ zugeordnet ist. Nach [Sta90] ist in diesem Fall für Stellen-Transitions-Netze eine Verschmelzung der Plätze konsistent.

Reduktionsregel 3: Zusammenfassen einer Platzsequenz

Voraussetzung: Sei t eine Transition im Netz N mit folgenden Eigenschaften:

1. $\bullet t = \{p_1\}$ und $t \bullet = \{p_2\}$ mit $p_1 \neq p_2$,
2. $V([p_1, t]) = V([t, p_2])$,
3. t ist instantan und
4. $\forall t' \in \bullet p_2$ gilt: t' ist instantan.

Anwendung: Für die Plätze p_1 und p_2 führen wir im Netz N' einen neuen Platz p'_1 ein mit

1. $\bullet p'_1 = \bullet p_1 \cup (\bullet p_2 \setminus \{t\})$,

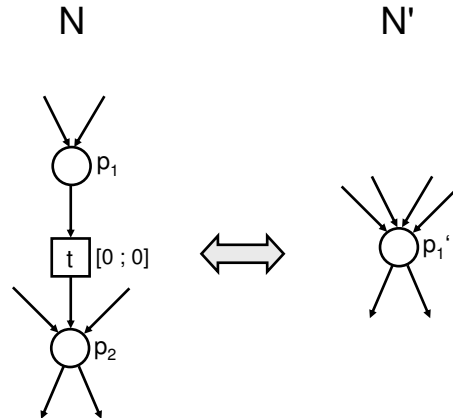


Abbildung 4.6.: Strukturelle Reduktionsregel für Intervall-Petrinetze: Reduktionsregel 3.

2. $p_1' \bullet = p_2 \bullet$ und
3. $m_0(p_1') = m_0(p_1) + m_0(p_2)$.

Anschließend streichen wir die Plätze p_1 und p_2 , die Transition t und ihre ein- und ausgehenden Kanten im Netz N' .

Satz 4.3 Die Reduktionsregel 3 ist konsistent.

Beweis. Es genügt zu zeigen, dass N und N' das gleiche Zeitverhalten haben. Sei p_1 zum Zeitpunkt τ markiert, das heißt t ist zum Zeitpunkt τ aktiviert (mit $h(t) = 0$). Dann ist p_1' in N' zum Zeitpunkt τ markiert. t schaltet in N zum Zeitpunkt $\tau + 0$ (mit $h(t) = 0$). Dann ist der Platz p_2 in N zum Zeitpunkt $\tau + 0$ markiert. Die Reduktionsregel 3 ist konsistent. \square

Um das konkrete Modell strukturell zu reduzieren, verschmelzen wir im Folgenden alle Plätze, die dem Muster der Reduktionsregel 3 entsprechen. Erneut wird so der Zustandsraum insofern reduziert, dass der zu untersuchende Erreichbarkeitsgraph weniger Zwischenzustände enthält.

Definition 21 (Netz NR_2)

Sei NR_2 das Netz, das entsteht, wenn wir Reduktionsregeln 1 bis 3 auf die Komposition NR_1 anwenden (mit $NR_2 = N'_{Env} \oplus N'_{B2} \oplus N'_{B3} \oplus N'_{B4} \oplus N'_{aQu}$). \lrcorner

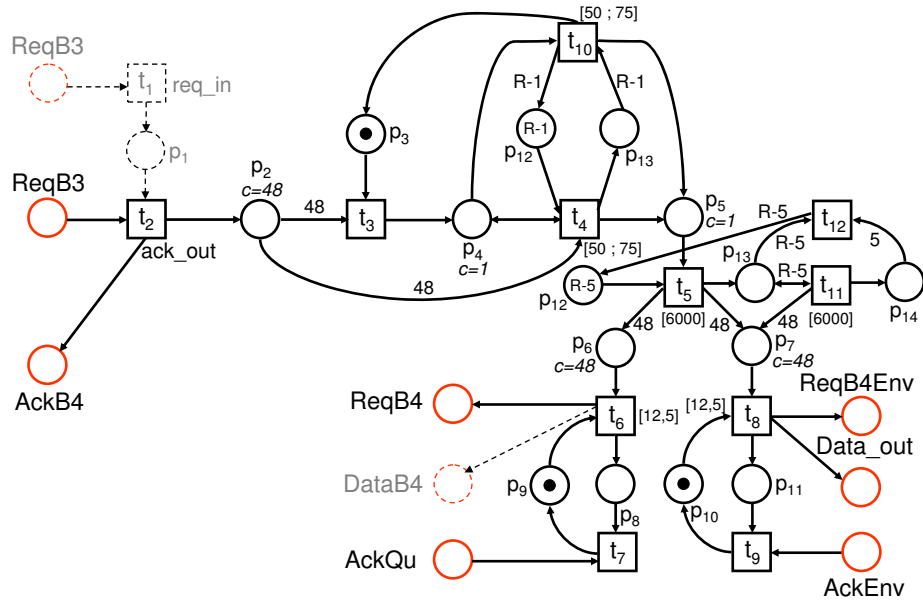


Abbildung 4.7.: Das strukturell reduzierte Netz N'_{B4} (nach Anwenden der Reduktionsregeln 1 und 3).

Die Abbildung 4.7 zeigt das Netz N'_{B4} nach Anwendung der strukturellen Reduktionsregeln auf die Komposition NR_1 .

An der Schnittstelle zwischen den Netzen N_{B3} und N_{B4} wenden wir die Reduktionsregel 3 auf die Transition t_1 in N_{B4} an und streichen diese. Anschließend haben die Plätze $DataB3$ und $ReqB3$ den gleichen Vor- und Nachbereich und erfüllen die Voraussetzung der Reduktionsregel 1. Demzufolge streichen wir den Platz $ReqB3$. Auf eine Abbildung von N'_{B3} verzichten wir an dieser Stelle und verweisen auf Abb. 4.13 in Kapitel 4.3.

An der Schnittstelle zwischen den Netzen N_{B2} und N_{aQu} wenden wir die Reduktionsregel 3 auf die Transition t_7 des Netzes N_{B2} an und streichen diese (siehe Abb. 4.5 und Abb. 4.2).

Das vorgestellte Verfahren der strukturellen Reduktion bietet den Vorteil, dass der Zustandsraum von NR_2 kleiner ist als der Zustandsraum von NR_1 . Die Struktur des Netzes NR_2 besteht aus 129 Plätzen und 72 Transitionen (vgl. Tabelle 4.1).

4.3. Reduktion durch Verhaltensbeobachtung

Neben strukturellen Reduktionen, die für beliebige Intervall-Petrinetze gelten, können wir die Struktur des konkreten Modells weiter zusammenfassen, indem wir auf unser Wissen über die Schaltung zurückgreifen. Dieses Wissen erlaubt uns, Reduktionen an Netz NR_2 vorzunehmen, die nicht aus der Struktur abgeleitet werden können. Im Folgenden überführen wir das Netz NR_2 in das Netz NR_3 . Wir beschreiben in diesem Kapitel die Abstraktion der Schnittstellen der Netze N'_{B_2} , N'_{B_3} , N'_{B_4} und N'_{aQu} und eine weitere Reduktion des Netzes N'_{aQu} unter Bewahrung des zu untersuchenden Verhaltens. Das resultierende Netz ist $NR_3 = N'_{Env} \oplus N''_{B_2} \oplus N''_{B_3} \oplus N''_{B_4} \oplus N''_{aQu}$. Die Struktur des Netzes N'_{Env} wird nicht weiter reduziert.

4.3.1. Schnittstellenreduktion

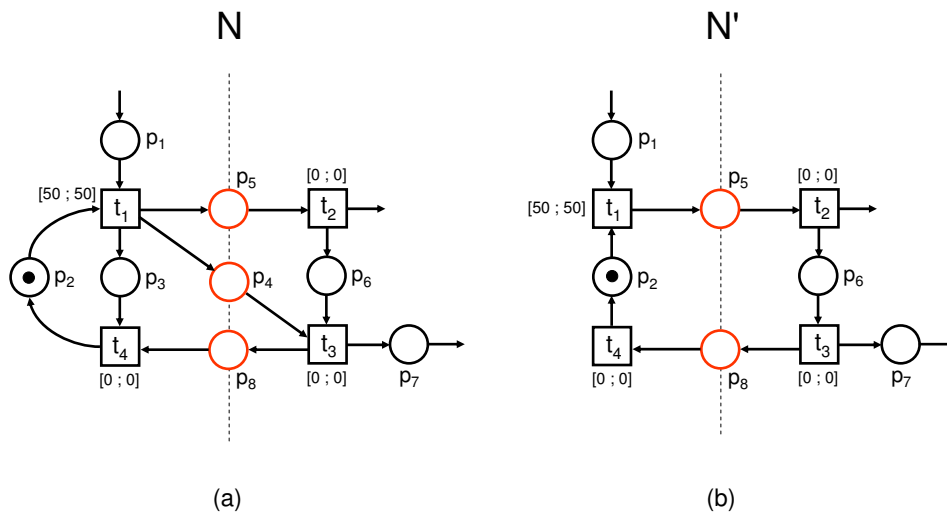


Abbildung 4.8.: N' ist die Abstraktion von N nach Anwendung der Schnittstellenreduktion.

Um eine Möglichkeit der *Schnittstellenreduktion* zu zeigen, betrachten wir exemplarisch die Netze N und N' in Abb. 4.8. Das Netz N stellt die Schnittstellen der Netze N'_{B_2} und N'_{B_4} (aus NR_2) dar. Es fällt auf, dass nur die Transition t_1 mit einem Zeitintervall ungleich Null versehen ist: Alle anderen Transitionen schalten instantan. Um alle 50 Zeiteinheiten eine Schaltsequenz von p_1 nach p_7 zu realisieren, sind einige Transitionen und Plätze in

N nicht notwendig, da sie keine Aspekte modellieren, die für die zu lösenden Probleme relevant sind.

Offensichtlich gelten in jedem erreichbaren Zustand $z = (m, h)$ von RG_N folgende Gleichungen:

- $m(p_2) = 1 - m(p_3)$
- $m(p_5) + m(p_6) = m(p_4)$

In jeder Markierung von N ist entweder der Platz p_2 oder der Platz p_3 markiert. Zudem ist der Platz p_4 stets markiert, wenn entweder p_5 oder p_6 markiert ist. Diese Beobachtungen ermöglichen es, die Zustände der Netze N und N' in Abb. 4.8 in eine starke Bisimulationsrelation zu setzen.

Definition 22 (Starke Bisimulation)

Seien N, N' Intervall-Petrinetze und $RG_N, RG_{N'}$ ihre Erreichbarkeitsgraphen. Eine Relation S über den Zuständen von RG_N und $RG_{N'}$ heißt eine starke Bisimulation, wenn gilt:

1. Wenn $z \xrightarrow{t} z_1$ in RG_N und für alle Paare $(z, z') \in S$ gilt, dann existieren die Zustände $z_1, z'_1 \in RG_{N'}$, so dass $z' \xrightarrow{t} z'_1$ und $(z_1, z'_1) \in S$.
2. Wenn $z' \xrightarrow{t} z'_1$ in $RG_{N'}$ und für alle Paare $(z', z) \in S$ gilt, dann existieren die Zustände $z'_1, z_1 \in RG_N$, so dass $z \xrightarrow{t} z_1$ und $(z'_1, z_1) \in S$.

┘

Besteht zwischen den Netzen N und N' eine starke Bisimulation, so gilt jede Erreichbarkeitseigenschaft in N ebenfalls in N'. Natürlich sind die Zwischenzustände, von denen in $RG_{N'}$ abstrahiert wurde, nicht mehr erreichbar.

Des Weiteren definieren wir zwei Relationen R_α und R_γ .

Definition 23 (Relationen R_α und R_γ)

Sei $N = (P, T, F, V, m_0, I)$ das Intervall-Petrinetz aus Abb. 4.8(a). Sei $N' = (P', T', F', V, m_0, I)$ das Intervall-Petrinetz aus Abb. 4.8(b), sowie Z die Zustandsmenge von N und Z' die Zustandsmenge von N'.

$R_\alpha \subseteq Z \times Z'$ ist die Relation mit $((m, h), (m', h')) \in R_\alpha$, wenn:

- $m' = (m(p_1), m(p_2), m(p_5), m(p_6), m(p_7), m(p_8))$ und
- $h' = h$.

$R_\gamma \subseteq Z' \times Z$ ist die Relation mit $((m', h'), (m, h)) \in R_\gamma$, wenn:

- $m = (m'(p_1), 1 - m'(p_3), m'(p_3), m'(p_5) + m'(p_6), m'(p_5), m'(p_6), m'(p_7), m'(p_8))$ und
- $h = h'$.

┘

Lemma 4.4

Offensichtlich gilt $R_\alpha^{-1} = R_\gamma$ und $R_\gamma^{-1} = R_\alpha$.

┘

Das heißt, R_γ ist die Umkehrrelation von R_α und R_α ist die Umkehrrelation von R_γ .

Satz 4.5 R_α ist eine starke Bisimulationsrelation zwischen den Zuständen von RG_N und $RG_{N'}$.

Wir führen den Beweis für alle möglichen Schaltfolgen der Netze N und N' .

Beweis. Für alle Transitionen t von N und N' ist zu zeigen: t ist in N konzessioniert bzw. aktiviert gdw. t ist in N' konzessioniert bzw. aktiviert. Die Anfangsmarkierung für den Platz p_1 sei $m_0(p_1) = n$ mit $1 \leq n \leq 48$.

- 1a) Sei $z = (m, h)$ ein Zustand in RG_N mit $m = [n'p_1, p_2]$ und $h(t_1) = 0$ (die Transition t_1 ist in N konzessioniert). Dann gibt es einen Zustand $R_\alpha(z)$, der t_1 in N' konzessioniert mit $R_\alpha(z) = ([n'p_1, p_2], 0)$. Nach einem Zeitverlauf von 50 Zeiteinheiten ist t_1 in N aktiviert mit $h(t_1) = 50$. Dann ist t_1 mit $R_\alpha(z) = ([n'p_1, p_2], 50)$ in N' aktiviert.
- 1b) Sei $z' = (m', h')$ ein Zustand in $RG_{N'}$ mit $m' = [n'p_1, p_2]$ und $h'(t_1) = 0$ (die Transition t_1 ist in N' konzessioniert). Dann gibt es einen Zustand $R_\alpha^{-1}(z')$, der t_1 in N konzessioniert mit $R_\alpha^{-1}(z') = ([n'p_1, p_2], 0)$. Nach einem Zeitverlauf von 50 Zeiteinheiten ist t_1 in N' aktiviert mit $h'(t_1) = 50$. Dann ist t_1 mit $R_\alpha^{-1}(z') = ([n'p_1, p_2], 50)$ in N aktiviert.
- 2a) Sei $z = (m, h)$ ein Zustand in RG_N mit $m = [n'p_1, p_2]$ und $h(t_1) = 50$, der t_1 in N aktiviert. Dann gibt es einen Zustand $R_\alpha(z) = z'$, der t_1 in N' aktiviert mit $R_\alpha(z) = ([n'p_1, p_2], 50)$. Schaltet t_1 in N , so gilt $z \xrightarrow{t_1} z'$.
- 2b) Sei $z' = (m', h')$ ein Zustand in $RG_{N'}$ mit $m' = [n'p_1, p_2]$ und $h'(t_1) = 50$, der t_1 in N' aktiviert. Dann gibt es einen Zustand $R_\alpha^{-1}(z') = z$, der t_1 in N aktiviert mit $R_\alpha^{-1}(z') = ([n'p_1, p_2], 50)$.

- 3a) Sei $z = (m, h)$ ein Zustand in RG_N mit $m = [n'p_1, p_3, p_4, p_5]$ und $h(t_2) = 0$, der t_2 in N aktiviert. Dann gibt es einen Zustand $R_\alpha(z) = z'$, der t_2 in N' aktiviert mit $R_\alpha(z) = ([n'p_1, p_5], 0)$. Schaltet t_2 in N , so gilt $z \xrightarrow{t_2} z'$.
- 3b) Sei $z' = (m', h')$ ein Zustand in $RG_{N'}$ mit $m' = [n'p_1, p_5]$ und $h'(t_2) = 0$, der t_2 in N' aktiviert. Dann gibt es einen Zustand $R_\alpha^{-1}(z') = z$, der t_2 in N aktiviert mit $R_\alpha^{-1}(z') = ([n'p_1, p_3, p_4, p_5], 0)$.
- 4a) Sei $z = (m, h)$ ein Zustand in RG_N mit $m = [n'p_1, p_3, p_4, p_6]$ und $h(t_3) = 0$, der t_3 in N aktiviert. Dann gibt es einen Zustand $R_\alpha(z) = z'$, der t_3 in N' aktiviert mit $R_\alpha(z) = ([n'p_1, p_6], 0)$. Schaltet t_3 in N , so gilt $z \xrightarrow{t_3} z'$.
- 4b) Sei $z' = (m', h')$ ein Zustand in $RG_{N'}$ mit $m' = [n'p_1, p_6]$ und $h'(t_3) = 0$, der t_3 in N' aktiviert. Dann gibt es einen Zustand $R_\alpha^{-1}(z') = z$, der t_3 in N aktiviert mit $R_\alpha^{-1}(z') = ([n'p_1, p_3, p_4, p_6], 0)$.
- 5a) Sei $z = (m, h)$ ein Zustand in RG_N mit $m = [n'p_1, p_3, p_7, p_8]$ und $h(t_4) = 0$, der t_4 in N aktiviert. Dann gibt es einen Zustand $R_\alpha(z) = z'$, der t_4 in N' aktiviert mit $R_\alpha(z) = ([n'p_1, p_7, p_8], 0)$. Schaltet t_4 in N , so gilt $z \xrightarrow{t_4} z'$.
- 5b) Sei $z' = (m', h')$ ein Zustand in $RG_{N'}$ mit $m' = [n'p_1, p_7, p_8]$ und $h'(t_4) = 0$, der t_4 in N' aktiviert. Dann gibt es einen Zustand $R_\alpha^{-1}(z') = z$, der t_4 in N aktiviert mit $R_\alpha^{-1}(z') = ([n'p_1, p_3, p_7, p_8], 0)$.

□

Abbildung 4.9 veranschaulicht den Zustandsraum der Netze N und N' . Offensichtlich gilt für jeden Übergang in N und N' , das R_α eine starke Bisimulationsrelation ist.

Definition 24 (Schnittstellenreduktion)

Sei R_α die Relation aus Definition 23. Dann nennen wir die Anwendung von R_α auf ein Intervall-Petrimetz Schnittstellenreduktion. ┘

Wir haben gezeigt, dass die Schnittstellenreduktion einer starken Bisimulationsrelation entspricht. Insbesondere gilt, dass starke Bisimulation CTL*-Eigenschaften bewahrt [Mil71]. Die Eigenschaften aus Kapitel 3.3 können als CTL*-Eigenschaften beschrieben werden. Somit ist es uns möglich, diese auch nach Anwendung der Schnittstellenreduktion zu untersuchen. Die obige Argumentation zeigt, dass sich das zeitliche Verhalten der Schnittstellen der Netze durch die Schnittstellenreduktion nicht verändert. Die Reihenfolge der Ereignisse im Receiver ist nach Anwendung der Schnittstellenreduktion die gleiche.

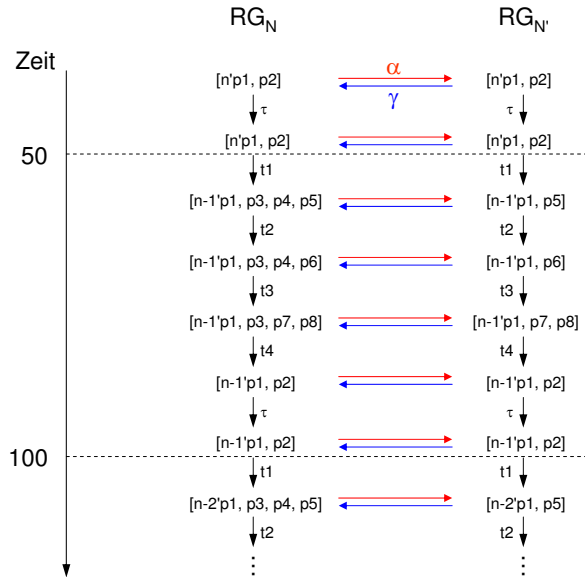


Abbildung 4.9.: Illustration zu Satz 4.5.

Betrachten wir weiterhin die Netze N und N' in Abb. 4.10. Offensichtlich entsprechen die Plätze p_8 und p_2 den Voraussetzungen der strukturellen Reduktionsregel 3. Wir verschmelzen demzufolge die Plätze p_8 und p_2 nach Anwendung der Schnittstellenreduktion. Sei p_2 der resultierende Platz in Abb. 4.10.

Wir wenden im Folgenden die Schnittstellenreduktion auf die Komposition NR_2 an. Alle hier aufgezählten Abbildungen sind in Kapitel 4.3.2 zu finden. Zunächst reduzieren wir das Teilnetz in N'_{B_2} , das die Datenausgabe modelliert. Um Marken an N'_{B_3} abzugeben und gleichzeitig das zeitliche Verhalten während der Handshakeaktivitäten zu bewahren, ist lediglich die Transition t_5 notwendig. Das Netz N''_{B_2} nach Anwendung der Schnittstellenreduktion ist in Abb. 4.12 dargestellt. Analog verfahren wir an den Ausgabeschnittstellen von N'_{B_4} (siehe Abb. 4.14).

Ebenso wie die Reduktionsregeln 2 und 3 reduziert die Schnittstellenreduktion den Zustandsraum des zu untersuchenden Netzes insofern, dass der Zustandsgraph weniger Zwischenzustände enthält und, wie gezeigt, die Anwendung weiterer struktureller Reduktionsregeln erlaubt. In jedem Teilnetz der Komposition NR_2 , in dem wir mittels struktureller Reduktion und Schnittstellenreduktion die Handshakes vereinfachen, sparen wir im Zustandsgraphen pro Handshake einen Zwischenzustand ein. Die Handshakes der Teil-

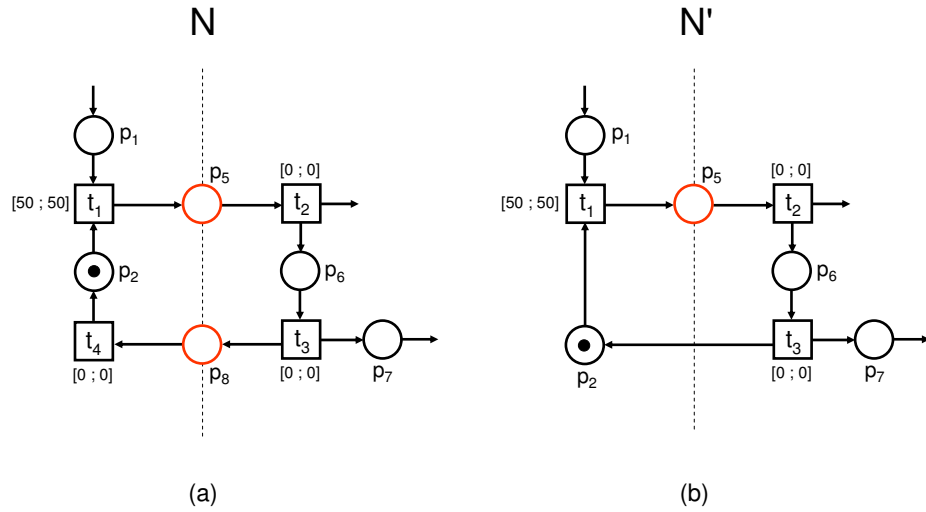


Abbildung 4.10.: Das Netz N' nach Anwendung der strukturellen Reduktionsregel 3.

netze N'_{B_2} und N'_{B_4} finden nebenläufig statt. Durch die Reduktion der Zustandsräume der einzelnen Netze fällt die Zustandsexplosion insgesamt geringer aus.

4.3.2. Reduktion der Asynchronen Queue

Im Folgenden betrachten wir zunächst das Netz N'_{aQu} und untersuchen dessen Verhalten an den Ein- und Ausgabepunkten. Anschließend reduzieren wir die Struktur des Netzes N'_{aQu} und stützen uns dabei auf Fakten, die offensichtlich in NR_2 gelten. Auf diese Fakten gehen wir im Folgenden ein.

Die Länge der Asynchronen Queue entspricht der Größe eines Symbols: Die Struktur kann genau ein Symbol aufnehmen. Zudem ist das Aufnahmeverhalten des Netzes N'_{aQu} durch das Ausgabeverhalten von N'_{B_4} bestimmt. Insbesondere hat der Platz p_6 in N'_{B_4} (Abb. 4.7) die Kapazität $c = 48$. Die Asynchrone Queue muss also höchstens ein Symbol speichern. Das Netz N'_{aQu} modelliert für 48 Übergänge eine Verzögerung von jeweils $eft(t) = 3$ und $lft(t) = 5$, wobei jede Transition t , die eine FIFO-Stufe modelliert, einem Schaltzwang unterliegt. Somit ist das Weiterleiten eines Tokens von einer Stufe in die nächste schneller möglich als die Aufnahme eines Tokens mit $I(t_6) = [12,5; 12,5]$ in Abb. 4.7. Das Netz N'_{aQu} leitet ein Token stets rechtzeitig von der ersten in die zweite Stufe weiter, bevor

ein neues Token aufgenommen wird. Dementsprechend ist es für das Netz N'_{aQu} stets möglich, eingehende Daten aufzunehmen.

Das Ausgabeverhalten der Asynchronen Queue ist durch das Aufnahmeverhalten von B2 bestimmt. Nachdem B2 die ersten fünf Symbole eines Frames aufgenommen und weitergeleitet hat, ist es für diesen Block nur möglich, dass sechste Symbol entgegenzunehmen, wenn das erste Symbol in der Asynchronen Queue bereit liegt. Ab diesem Zeitpunkt nimmt N'_{B2} alle 50 ns ein Token von N'_{aQu} entgegen. Sei τ der Zeitpunkt zu dem die erste Marke eines Symbols am Ausgang von N'_{B4} liegt. Wenn diese Marke anschließend die Queue passiert hat, so liegt sie frühestens nach $\tau + 206,5$ ns (mit $\tau + 12,5$ ns + $48 \cdot 3$ ns + 50 ns) und spätestens nach $\tau + 302,5$ ns (mit $\tau + 12,5$ ns + $48 \cdot 5$ ns + 50 ns) in N'_{B2} . Ab diesem frühesten bzw. spätesten Zeitpunkt tritt alle 50 ns eine Marke aus N'_{aQu} aus. Dementsprechend hat die Queue ein komplettes Symbol frühestens nach 2556,5 ns ($206,5$ ns + $47 \cdot 50$ ns) und spätestens nach 2652,5 ns ($302,5$ ns + $47 \cdot 50$ ns) ausgegeben. Das Netz N'_{B4} gibt jedoch lediglich alle 6000 ns ein weiteres Symbol an N'_{aQu} ab ($I(t_5) = [6000; 6000]$ in Abb. 4.7). Auf Grund dieser Annahmen ist es uns möglich, folgendes Korollar zu formulieren.

Korollar 4.6 Für das Netz N'_{aQu} ist es stets möglich

- alle 12,5 ns eine Marke aufzunehmen, wenn ein Symbol an dessen Eingang liegt,
- frühestens nach 206,5 ns und spätestens nach 302,5 ns alle 50 ns eine Marke auszugeben, und
- frühestens nach 2556,5 ns und spätestens nach 2652,5 ns das gesamte Symbol (48 Marken) auszugeben.

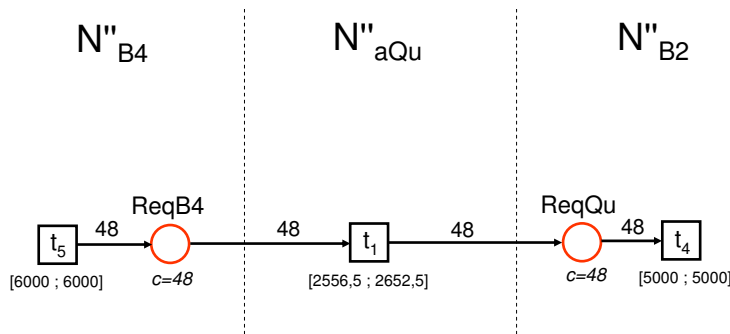


Abbildung 4.11.: Das zusammengefasste Netz N''_{aQu} mit den Schnittstellen zu N''_{B2} und N''_{B3} .

4. Abstraktion des Modells

Wir abstrahieren im Folgenden von der konkreten Belegung des Netzes N'_{aQu} und fassen dessen Struktur zu einer einzigen Transition zusammen. Gleichzeitig ist es möglich, die Ausgabeschnittstellen in N'_{B4} und die Eingabeschnittstelle in N'_{B2} zu reduzieren. Anhand des Korollars reduzieren wir das Netz N'_{aQu} entsprechend der Abb. 4.11 zu N''_{aQu} . Das beschriebene Zeitverhalten ist durch das Intervall von t_1 , $I(t_1) = [2556,5; 2652,5]$, gewährleistet.

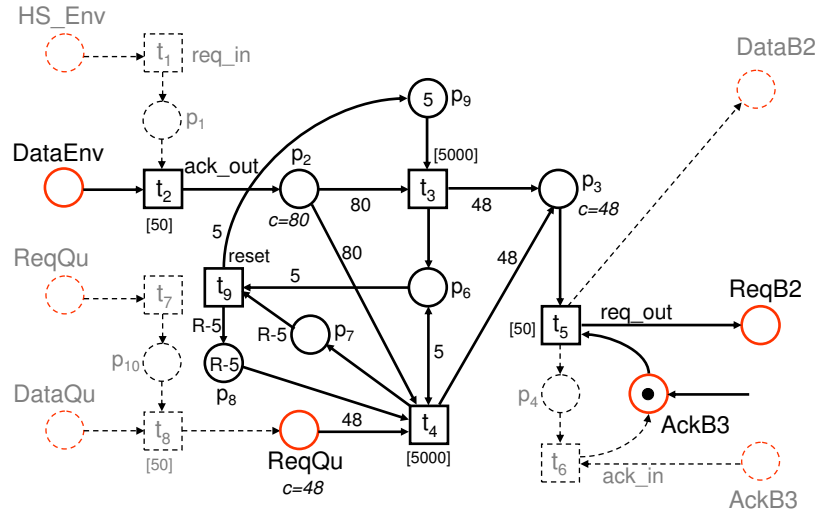


Abbildung 4.12.: N''_{B2} nach Anwendung der Schnittstellenreduktion und der Abstraktion von N'_{aQu} .

Definition 25 (Netz NR_3)

Sei NR_3 das Netz, das nach Anwendung der Schnittstellenreduktion und der Reduktion des Netzes N'_{aQu} auf die Komposition NR_2 entsteht (mit $NR_3 = N'_{Env} \oplus N''_{B2} \oplus N''_{B3} \oplus N''_{B4} \oplus N''_{aQu}$).

Es ist schwierig, den Zustandsraum für die Asynchrone Queue pro Symbol abzuschätzen. Simulationen haben ergeben, dass der Erreichbarkeitsgraph bereits 383 unterschiedliche Markierungen enthält, wenn die Asynchrone Queue die erste Marke ausgibt. Sobald zehn Marken ausgegeben sind, enthält der Zustandsraum bereits 787 unterschiedliche Markierungen. Zudem sind für diese Simulation die jeweiligen Transitionsmarkierungen im Erreichbarkeitsgraph nicht berücksichtigt. Es ist offensichtlich, dass die Abstraktion des Netzes N'_{aQu} eine große Einsparung von Zwischenzuständen im zu untersuchenden Erreichbarkeitsgraph und somit eine effizientere Erreichbarkeitsanalyse realisiert. Der Zustandsraum von NR_3 enthält für das Netz N''_{aQu} pro Symbol lediglich einen Zustand.

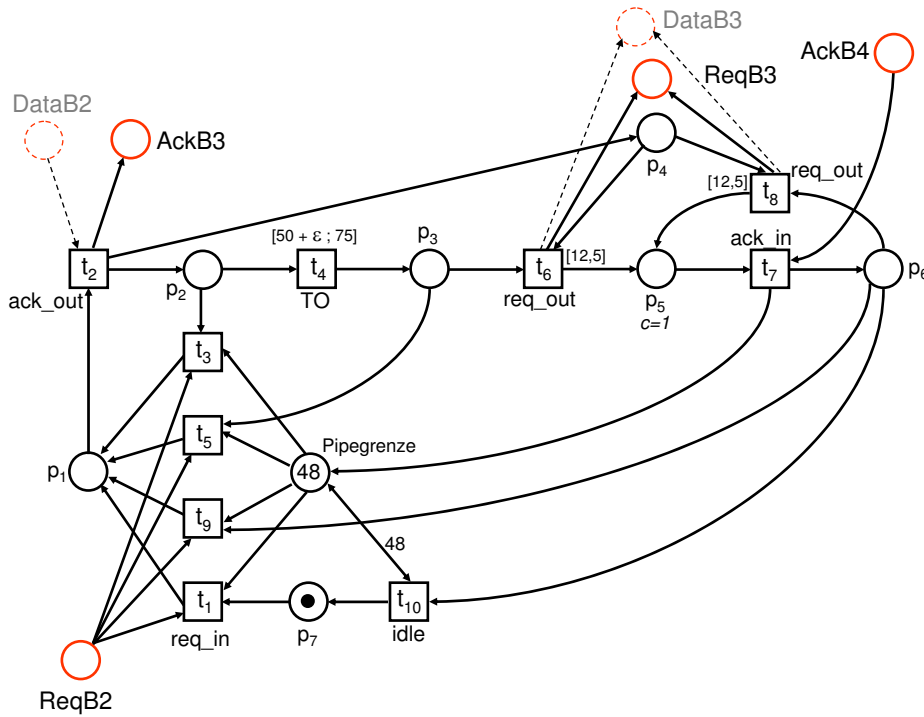


Abbildung 4.13.: N''_{B3} nach Anwendung der Schnittstellenreduktion.

Durch Anwendung der Schnittstellenreduktion und der Abstraktion der Asynchronen Queue enthält die Komposition NR_3 insgesamt 34 Plätze und 29 Transitionen. Die Abbildungen 4.11 bis 4.14 zeigen die Teilnetze von NR_3 nach Anwendung der Schnittstellenreduktion und Abstraktion des Netzes N'_{aQu} .

4.3.3. Weiterführende Beobachtungen

An dieser Stelle greifen wir Fragen 1 bis 4 aus Kapitel 3.3 auf und diskutieren die resultierenden Analysemöglichkeiten nach der Überführung von NR_2 nach NR_3 .

1. Ist es möglich, dass der Receiver nach einer gewissen Initialisierungsphase Daten konstant mit 80 MHz zur Verfügung stellt, das heißt B4 Daten mit 80 MHz ausgibt?

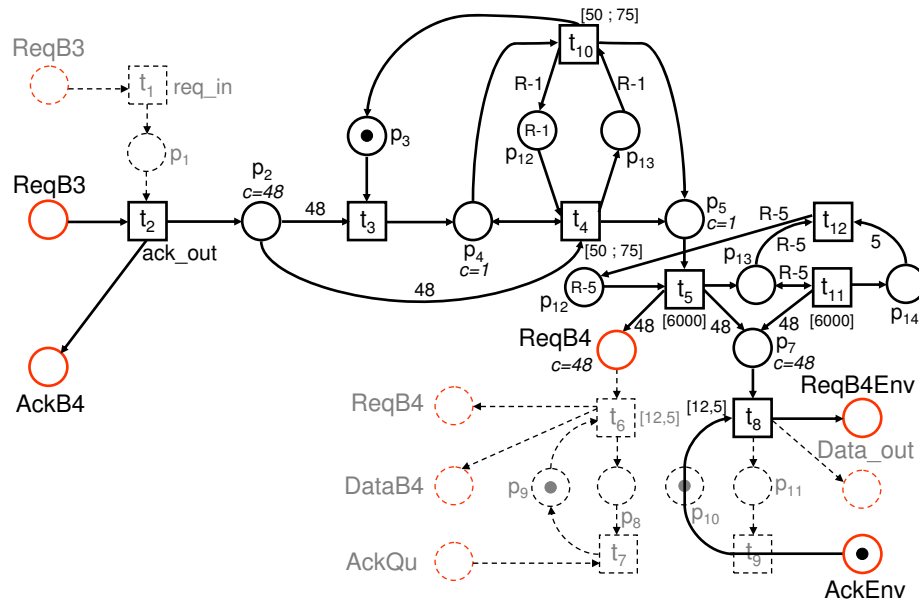


Abbildung 4.14.: N''_{B4} nach Anwendung der Schnittstellenreduktion und der Abstraktion von N'_{aQu} .

Durch die Reduktion der Struktur des Netzes N''_{aQu} ist der Zustandsraum des Netzes NR_3 kleiner als der des Netzes NR_2 . Die Erreichbarkeitsanalyse kann somit für die 1. Frage effizienter durchgeführt werden.

2. Ist es möglich, den Receiver mit einer niedrigeren Taktrate zu betreiben?

Da wir diese Frage analog zur ersten Frage untersuchen, sind die Vorteile hier die gleichen.

3. Ist der Receiver tolerant gegenüber Verzögerungen des Datenflusses in der Asynchronen Queue?

Reagiert die Asynchrone Queue während der Berechnung eines Frames innerhalb der für diese Struktur vorgesehenen minimalen und maximalen Reaktionszeiten, so ist der Receiver tolerant gegenüber allen möglichen Verzögerungen des Datenflusses in der Queue. Entsprechend der Aussage in Korollar 4.6 lassen sich die Fragen 3.1 und 3.2 aus Kapitel 3.3 positiv beantworten.

4. Ist es möglich, den Receiver zu betreiben, wenn die Asynchrone Queue weniger Stufen enthält?

Es ist nicht möglich, die Frage 4 auf die in Kapitel 3.3 beschriebene Art, am Netz NR_3 zu untersuchen. Die Reduktion des Netzes N'_{aQu} basiert auf den Beobachtungen aus Korollar 4.6. Diese Aussage wiederum ist nur wahr, wenn die Asynchrone Queue tatsächlich 48 Stufen implementiert. Somit können wir mittels NR_3 nicht untersuchen, ob der Platz error des Netzes N_K in Abb. 3.11) noch immer erreichbar ist, wenn der Transition t_1 im Netz N''_{aQu} (siehe Abb. 4.11) ein kleineres oder größeres Zeitintervall zugewiesen ist. Untersuchungen bezüglich der 4. Fragestellung lassen sich mit Hilfe des Netzes NR_2 analysieren.

4.4. Abstraktion der Anfangsmarkierung

Relevant für die Größe des Zustandsraums ist im konkreten Modell vor allem die Anfangsmarkierung. Je mehr Marken die Anfangsmarkierung des Netzes NR_3 enthält, desto mehr Zwischenzustände enthält der entsprechende Erreichbarkeitsgraph. Ein externes Symbol entspricht im Modell 80 Marken, ein internes 48. Da wir den Schaltkreis während der Abarbeitung eines Frames untersuchen und dieser einige hundert Symbole enthalten kann, ergibt sich am Eingang des konkreten Modells eine Markierung von mindestens 8000 Marken (mit $R = 100$ in Abb. 4.15). Diese Anzahl ist jedoch für die Analyse der vier Verifikationsaufgaben nicht notwendig und vergrößert darüber hinaus den Zustandsraum unnötig.

Im Folgenden betrachten wir die Netze NR_3 aus Definition 25 und NR_4 , welches wir anschließend definieren. NR_4 ist das Netz, das durch Abstraktion von NR_3 entsteht. Wir abstrahieren von der Anfangsmarkierung des konkreten Netzes und lassen im abstrakten Modell pro Symbol lediglich 6 statt 48 Handshakes zu. Somit besteht ein internes Symbol im abstrakten Modell aus 6 Marken, ein externes Symbol besteht aus 10 Marken. Entsprechend beträgt die Vielfachheiten der Bögen, die in N'_{Env} die Datenausgabe an N''_{B2} modellieren, im abstrakten Modell 10 statt 80 (siehe Abb. 4.15). Gleichzeitig verachtfachen wir die Schaltdauer der Transitionen, die die Handshakes modellieren (siehe beispielsweise $I(t_2)$ in Abb. 4.18).

Um die Erreichbarkeitsanalyse der 1. und 2. Frage weiterhin durchzuführen, vergrößern wir ebenfalls das Zeitintervall der Transition t_2 des Konsumenten-Netzes um das Achtfache (siehe Abb. 4.16). Entsprechend ist der Platz error in N'_K mit $I(t_3) = [100 + \varepsilon; 100 + \varepsilon]$ erreichbar.

Im Rahmen der Abstraktion passen wir lediglich die Zeitintervalle der Transitionen an, die Symbole tokenweise transportieren. Transitionen, die ein Symbol vollständig konsumieren und produzieren bleiben unverändert. Da bei der folgenden Argumentation einige

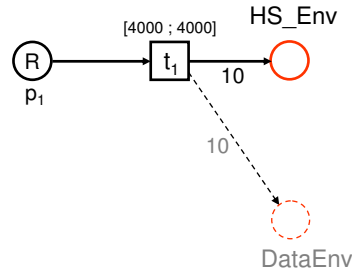


Abbildung 4.15.: N''_{Env} im Netz NR_4 .

Sonderfälle bezüglich des Verhaltens der Transitionen zu beachten sind, partitionieren wir zunächst die Menge der Transitionen in NR_3 . Dabei berücksichtigen wir die jeweiligen Sonderfälle.

Der Übersichtlichkeit halber betrachten wir im Folgenden zwei Netze C und A , wobei wir A in Definition 28 definieren. A ist das Netz, das durch Abstraktion von C entsteht. Zunächst partitionieren wir die Menge der Transitionen in C in die Mengen T_1 und T_2 mit $T_1 \cup T_2 = T_C$ und $T_1 \cap T_2 = \emptyset$. Transitionen innerhalb einer Partition sind äquivalent bezüglich ihres Verhaltens:

T_1 : Alle Transitionen t , mit $V([p, t]) = 80$ oder $V([p, t]) = 48$ oder $V([t, p]) = 80$ oder $V([t, p]) = 48$.

Sonderfall T_{1a} : t_4 in N''_{B3} (mit $V([p, t_4]) = V([t_4, p]) = 1$).

T_2 : Alle Transitionen t , die in C 80 bzw. 48 mal pro Symbol schalten mit $V([p, t]) = V([t, p]) = 1$ für ein $p \in \bullet t$ bzw. $p \in t \bullet$.

Sonderfälle T_{2a} bis T_{2c} : Alle Transitionen t , die in C 80 bzw. 48 mal pro Symbol schalten (mit $V([p, t]) = V([t, p]) = 1$), denen währenddessen jedoch die Konzession entzogen werden kann, mit

$$I(t) = \begin{cases} [50; 50], & \text{falls } t \in T_{2a} \text{ (} t_5 \text{ in } N''_{B2}\text{),} \\ [12,5; 12,5], & \text{falls } t \in T_{2b} \text{ (} t_6 \text{ und } t_8 \text{ am Ausgang von } N''_{B3}\text{),} \\ [12,5 + \varepsilon; 12,5 + \varepsilon], & \text{falls } t \in T_{2c} \text{ (} t_3 \text{ in } N'_K \text{ in Abb. 4.16).} \end{cases}$$

Weiterhin teilen wir alle Plätze des Netzes C in Partitionen P_1 und P_2 auf mit $P_1 \cup P_2 = P_C$ und $P_1 \cap P_2 = \emptyset$.

P_1 : Alle Plätze p mit $V([p, t]) = 80$ oder $V([p, t]) = 48$ oder $V([t, p]) = 80$ oder $V([t, p]) = 48$.

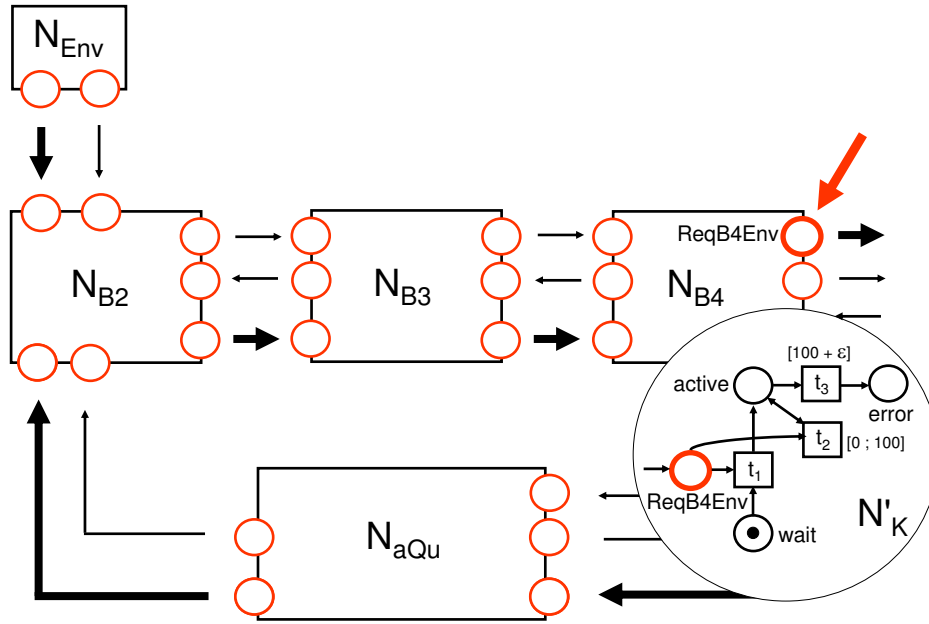


Abbildung 4.16.: N'_K im Netz NR_4 .

P_2 : Alle Plätze p , die nicht in P_1 enthalten sind.

Die Konstruktion der Partitionen T_i und P_i ermöglicht es uns, die Abstraktionsfunktion α zu definieren, mit der wir alle Zustände von $RG_C(m_0)$ in Zustände von $RG_A(m_0)$ überführen. Des Weiteren definieren wir die dazugehörige Konkretisierungsfunktion γ .

Definition 26 (Abstraktionsfunktion α)

Sei $C = (P, T, F, V, m_0, I)$ ein Intervall-Petrietz. Dann ist die Abstraktionsfunktion α die Funktion, die jeden Zustand aus $RG_C(m_0)$ auf einen Zustand aus $RG_A(m_0)$ abbildet mit $\alpha(z_c) = \alpha((m_c, h_c)) = (\alpha_1(m_c), \alpha_2(h_c))$. Im Folgenden bezeichnen wir einen Zustand aus $RG_A(m_0)$ mit $z_a = (m_a, h_a)$. α_1 und α_2 sind wie folgt definiert:

- $\forall p \in P_1: \alpha_1(m_c(p)) = \left\lfloor \frac{m_c(p)}{8} \right\rfloor$
- $\forall p \in P_2: \alpha_1(m_c(p)) = m_c(p)$
- $\forall t \in T_1: \alpha_2(h_c(t)) = h_c(t)$

$$\bullet \forall t \in T_2: \alpha_2(h_c(t)) = \begin{cases} \#, & \text{falls } h_c(t) = \#, \\ x \text{ mit } x \in [50; 400], & \text{falls } t \in T_{2a}, \\ x \text{ mit } x \in [12,5; 100], & \text{falls } t \in T_{2b}, \\ x \text{ mit } x \in [100 + \varepsilon; 100 + \varepsilon], & \text{falls } t \in T_{2c}, \\ 8 \cdot h_c(t), & \text{sonst.} \end{cases}$$

┘

Definition 27 (Konkretisierungsfunktion γ)

Sei $A = (P, T, F, V, m_0, I)$ ein Intervall-Petrinetz. Dann ist die Konkretisierungsfunktion γ die Funktion, die jeden Zustand aus $RG_A(m_0)$ in eine nichtleere Menge von Zuständen aus $RG_C(m_0)$ überführt mit $\gamma(z_a) = \gamma((m_a, h_a)) = \gamma_1(m_a) \times \gamma_2(h_a)$. Im Folgenden bezeichnen wir einen Zustand aus $RG_C(m_0)$ mit $z_c = (m_c, h_c)$. γ_1 und γ_2 sind wie folgt definiert:

- $\forall p \in P_1: \gamma_1(m_a(p)) = \{m_a(p) \cdot 8 + i \mid i = 0, \dots, 7\}$
- $\forall p \in P_2: \gamma_1(m_a(p)) = \{m_a(p)\}$
- $\forall t \in T_1: \gamma_2(h_a(t)) = \{h_a(t)\}$

$$\bullet \forall t \in T_2: \gamma_2(h_a(t)) = \begin{cases} \{\#\}, & \text{falls } h_a(t) = \#, \\ \{0\}, & \text{falls } h_a(t) = 0, \\ \{50\}, & \text{falls } t \in T_{2a}, \\ \{12,5\}, & \text{falls } t \in T_{2b}, \\ \{12,5 + \varepsilon\}, & \text{falls } t \in T_{2c}, \\ \{\frac{1}{8} \cdot h_a(t)\}, & \text{sonst.} \end{cases}$$

┘

Mit der Abstraktionsfunktionen α ist es möglich, Mengen von Zuständen in C auf einen Zustand in A abzubilden. Im Rahmen der Abstraktion lassen wir während der entsprechenden Zustandsübergänge im abstrakten Modell mehr Zeit vergehen, indem wir den jeweiligen Transitionen größere Intervalle zuordnen. Das entstehende Netz ist wie folgt definiert:

Definition 28 (Netz A)

Sei $C = NR_3 = (P, T, F, V, m_0, I)$ ein Intervall-Petrinetz. Sei $A = (P, T, F, V', m'_0, I')$ das Intervall-Petrinetz mit

- $\forall f \in F : V'(f) = \begin{cases} \frac{1}{8} \cdot V(f), & \text{falls } V(f) = 80 \text{ oder } V(f) = 48, \\ V(f), & \text{sonst.} \end{cases}$
- $\forall t \in T : I'(t) = \begin{cases} [8 \cdot I_1(t); 8 \cdot I_2(t)], & \text{falls } t \in T_2 \text{ in } C, \\ [50; 400], & \text{falls } t \in T_{2a} \text{ in } C, \\ [12,5; 100], & \text{falls } t \in T_{2b} \text{ in } C, \\ [100 + \varepsilon; 100 + \varepsilon], & \text{falls } t \in T_{2c} \text{ in } C, \\ I(t), & \text{sonst.} \end{cases}$
- $m'_0 = \alpha(m_0)$.

□

Um die Lesbarkeit im folgenden Beweis zu erhöhen, benutzen wir $C = (P_C, T_C, F_C, V_C, m_0, I)$ und $A = (P_A, T_A, F_A, V_A, m_0, I)$.

Wir zeigen nun, dass wir durch den Abstraktionsschritt von C nach A die Analyse des Receivers zwar einschränken, die zu untersuchende Eigenschaft aus Kapitel 3 in A jedoch noch immer untersuchen können. Die Konkretisierungsfunktion γ bildet jeden Zustand $z_a = (m_a, h_a)$ aus $RG_A(m_0)$ auf eine Menge von Zuständen in $RG_C(m_0)$ ab, so dass gilt: für jeden Zustand $z_a \in RG_A(m_0)$ existiert mindestens ein Zustand $m_c \in RG_C(m_0)$ (vgl. Abb. 4.17).

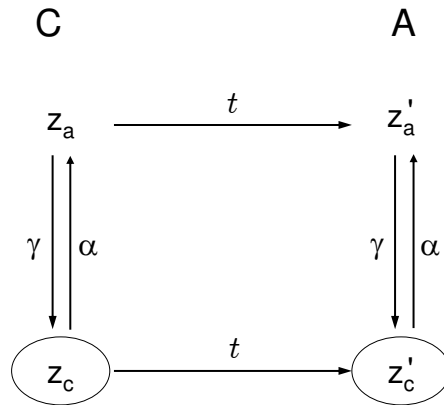


Abbildung 4.17.: Veranschaulichung zu Satz 4.7.

Satz 4.7 Sei $z_a, z'_a \in RG_A(m_0)$ und $z_a \xrightarrow{t} z'_a$. Dann gibt es Zustände $z_c, z'_c \in RG_C(m_0)$ mit:

- $z_c \in \gamma(z_a)$,
- $z_c \xrightarrow{t} z'_c$ und
- $\alpha(z'_c) = z'_a$.

Beweis. Seien z_a und z'_a Zustände wie in Satz 4.7. Wir argumentieren anhand der Partitionen T_1 und T_2 unter Berücksichtigung der jeweiligen Sonderfälle.

- 1.) Sei $t \in T_1$. Aus Definition 28 folgt, dass t in A und C mit den gleichen Zeitintervallen versehen ist. Sei $z_a = (m_a, h_a)$. Da t aktiviert ist, gilt

$$\forall p \in \bullet t: m_a(p) \geq V_A([p, t]) \text{ und}$$

$$\forall p_i \in t\bullet: m_a(p_i) = n_i \text{ (mit } n_i \in \mathbb{N}) \text{ und } eft(t) \leq h_a(t) \leq lft(t).$$

Somit gilt $z_a \xrightarrow{t} z'_a$ mit $z'_a = (m'_a, h'_a)$, wobei

$$\forall p \in P_A: m'_a(p) = m_a(p) - V([p, t]) + V([t, p]) \text{ und}$$

$$h'_a(t) = \begin{cases} 0, & \text{falls } t \text{ in } m'_a \text{ konzessioniert ist,} \\ \#, & \text{sonst.} \end{cases}$$

- Sei $z_c = (m_c, h_c) \in \gamma(z_a)$. Wir wählen z_c mit

$$\forall p \in P_C: m_c(p) = \begin{cases} 8 \cdot m_a(p) + 7, & \text{falls } p \in P_1, \\ m_a(p), & \text{sonst.} \end{cases}$$

$$\text{und } eft(t) \leq \gamma_2(h_a(t)) \leq lft(t).$$

- Aus Definition 28 folgt, dass die Bögen, deren Vielfachheit in A $V_A([p, t]) = 10$ oder $V_A([p, t]) = 6$ oder $V_A([t, p]) = 10$ oder $V_A([t, p]) = 6$ (mit $p \in P_1$) beträgt, in C das Achtfache beträgt. Die restlichen Transitionen bleiben unverändert. Aus Definition 27 folgt, dass zudem die Markierungen der Plätze p aus P_1 verachtacht werden. Somit ist t in z_c aktiviert. Dann gilt $z_c \xrightarrow{t} z'_c$ mit $z'_c = (m'_c, h'_c)$, wobei

$$\forall p: m'_c(p) = \begin{cases} 8 \cdot m'_a(p) + 7 - V_C([p, t]) + V_C([t, p]), & \text{falls } p \in P_1, \\ m'_a(p), & \text{sonst.} \end{cases}$$

und $h'_c(t) = \begin{cases} 0, & \text{falls } t \text{ in } m'_c \text{ konzessioniert ist,} \\ \#, & \text{sonst.} \end{cases}$

- Es folgt aus Definition 26 für alle $p \in P_C$:

$$\alpha_1(m'_c(p)) = \begin{cases} m'_a(p) + \lfloor \frac{7}{8} \rfloor - \frac{1}{8}V_C([p, t]) + \frac{1}{8}V_C([t, p]) \\ = m'_a(p) + 0 - V_A([p, t]) + V_A([t, p]), & \text{falls } p \in P_1, \\ m'_a(p), & \text{sonst.} \end{cases}$$

1a.) Der Beweis für den Sonderfall T_{1a} ist analog zu 1.).

2.) Sei $t \in T_2$. Es gilt $\bullet t = \{p_1\}$ und $t \bullet = \{p_2\}$ und $p_1, p_2 \in P_2$. Sei $z_a = (m_a, h_a)$ mit $m_a(p_1) = n_1$ und $m_a(p_2) = n_2$ (mit $n_1, n_2 \in \mathbb{N}$). Da t in m_a aktiviert ist, gilt $h_a(t) = 400$. Somit gilt $z_a \xrightarrow{t} z'_a$ mit $z'_a = (m'_a, h'_a)$, wobei $m'_a(p_1) = n_1 - 1$, $m'_a(p_2) = n_2 + 1$ und $h'_a(t) = 0$. Für alle anderen $p \in P_A$ gilt $m'_a(p) = m_a(p)$. Diese werden jedoch durch das Schalten von t nicht beeinflusst, darum erwähnen wir sie im Weiteren nicht.

- Dann sei $z_c = (m_c, h_c) \in \gamma(z_a) = \gamma((m_a, h_a))$. Wir wählen m_c aus $\gamma_1(m_a)$ mit $m_c(p_1) = 8 \cdot n_1 - 7$ und $m_c(p_2) = 8 \cdot n_2 + 7$ und $h_c(t) = \gamma_2(h_a(t)) = \frac{400}{8} = 50$.
- Dann ist t in z_c aktiviert, da die Markierung von p_1 in C durch die Anwendung von γ verachtacht wird und die Vielfachheit von t gleich bleibt. Aus der Monotonie des Schaltens folgt, dass t in C aktiviert ist, wenn t in A aktiviert ist. Dann gilt $z_c \xrightarrow{t} z'_c$ mit $z'_c = (m'_c, h'_c)$, wobei $m'_c(p_1) = 8 \cdot n_1 - 7 - 1 = 8n_1 - 8$, $m'_c(p_2) = 8 \cdot n_2 + 7 + 1 = 8 \cdot n_2 + 8$ und $h'_c(t) = 0$.
- Dann gilt $\alpha_1(m'_c(p_1)) = \frac{8}{8}n_1 - \lfloor \frac{8}{8} \rfloor = n_1 - 1$ und $\alpha_1(m'_c(p_2)) = \frac{8}{8}n_2 + \lfloor \frac{8}{8} \rfloor = n_2 + 1$ und $\alpha_2(h'_c(t)) = \frac{0}{8} = 0$.

2a.) Sei $t \in T_{2a}$. Der Beweis für den Sonderfall T_{2a} ist analog zu 2.) mit: $50 \leq h_a(t) \leq 400$ und $h'_a(t) = 0$.

- Dann ist $h_c(t) = 50$,
- $h'_c(t) = 0$ und
- $\alpha_2(h'_c(t)) = 0$.

2b.) Sei $t \in T_{2b}$. Der Beweis für den Sonderfall T_{2b} ist analog zu 2.) mit: $12,5 \leq h_a(t) \leq 100$ und $h'_a(t) = 0$.

- Dann ist $h_c(t) = 12,5$,
- $h'_c(t) = 0$ und

4. Abstraktion des Modells

- $\alpha_2(h'_a(t)) = 0$.

2c.) Sei $t \in T_{2c}$. Der Beweis für den Sonderfall T_{2c} ist analog zu 2.) mit: $h_a(t) = 100 + \varepsilon$ und $h'_a(t) = 0$.

- Dann ist $h_c(t) = 100 + \varepsilon$,
- $h'_c(t) = 0$ und
- $\alpha_2(h'_a(t)) = 0$.

□

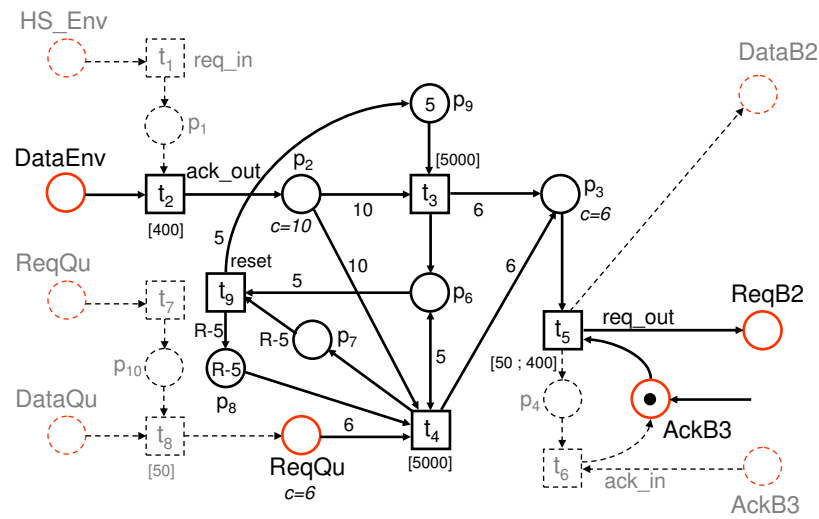


Abbildung 4.18.: Das abstrahierte Netz N'''_{B2} .

Es ist offensichtlich, dass es im Netz A die gleichen Transitionen und Plätze wie in C gibt. Jeder Ablauf, der in $RG_A(m_0)$ möglich ist, ist auch in $RG_C(m_0)$ möglich. Somit haben wir gezeigt, dass es für jeden Zustand im abstrakten Modell A mindestens einen Zustand im konkreten Modell C gibt. Mit Hilfe der Abstraktion realisieren wir eine schwache Bewahrung der Erreichbarkeit in A . Ermitteln wir bei der Analyse von $RG_A(m_0)$ einen Zustand, in dem $m(\text{error}) > 0$ ist, so ist dieser Zustand ebenfalls in $RG_C(m_0)$ erreichbar. Die Abstraktion durch α erlaubt uns jedoch nicht Aussagen über C zu treffen, wenn wir keinen Zustand mit $m(\text{error}) > 0$ in $RG_A(m_0)$ ermitteln. Die Anwendung der Abstraktionsfunktion α entspricht einer Unterapproximation des zu untersuchenden Erreichbarkeitsgraphen $RG_C(m_0)$. $RG_A(m_0)$ enthält nicht mehr alle Abläufe, die in $RG_C(m_0)$

möglich sind. Jedoch ist jede Schaltsequenz, die im abstrakten Modell möglich ist, auch im konkreten Modell möglich.

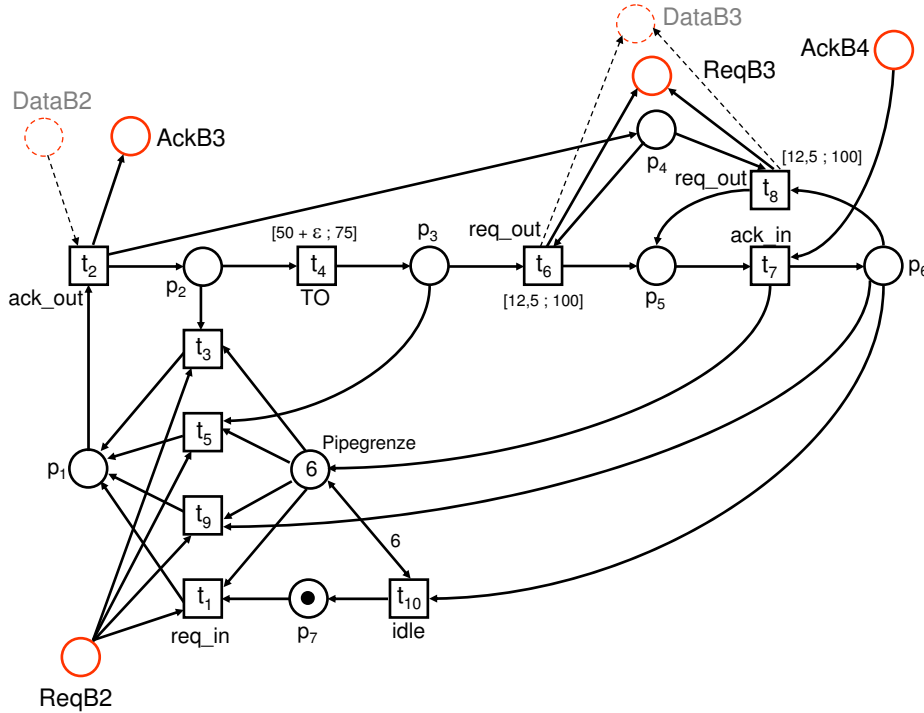


Abbildung 4.19.: Das abstrahierte Netz N'''_{B3} .

Definition 29 (Netz NR_4)

Sei NR_4 das Netz, das nach Anwendung der Abstraktionsfunktion α auf die Komposition NR_3 entsteht (mit $NR_4 = N''_{Env} \oplus N'''_{B2} \oplus N'''_{B3} \oplus N'''_{B4} \oplus N'''_{aQu}$). \lrcorner

Die obige Argumentation zeigt, dass sich das zeitliche Verhalten des abstrakten Modells nicht von dem des konkreten unterscheidet. Die Reihenfolge der Ereignisse im Receiver ist nach Anwendung der Abstraktionsfunktion α ebenfalls die gleiche. Allerdings ergibt sich in NR_4 ein unschärferes Verhalten für die Handshakeaktivitäten der GALS-Blöcke. Schaltet eine Handshake-Transition in NR_4 , so schaltet die gleiche Transition in NR_3 acht mal. Daraus folgt, dass wir für die Schaltung mittels NR_4 nur Aussagen über jedes achte Ereignis treffen können. Die Analyse des Modells NR_4 lässt dementsprechend lediglich gröbere Aussagen über die Schaltung zu. Allerdings ist der Zustandsraum dieser

Generieren weiterer Zwischenzuständen. Das Netz NR_1 besteht insgesamt aus 205 Plätzen und 84 Transitionen. Der Erreichbarkeitsgraph dieses Netzes ist für die computergestützte Verifikation zu groß.

Nach Anwendung der in Kapitel 4.2 vorgestellten strukturellen Reduktionsregeln besteht das resultierende Netz NR_2 aus 129 Plätzen und 72 Transitionen. So ergeben sich erste Einsparung von Zwischenzuständen im Erreichbarkeitsgraph des Modells. Allerdings brachen sowohl INA als auch LoLA die Berechnung des Zustandsraums für die vorgestellten Modelle NR_1 und NR_2 bzw. deren Skelette wegen Speicherüberlaufs ab. Tabelle 4.1 vergleicht die Größen aller in dieser Arbeit vorgestellten Netze bezüglich der jeweiligen Anzahl ihrer Plätze, Transitionen und Zustände.

Durch die in Kapitel 4.3 vorgestellte Schnittstellenreduktion und die Reduktion des Netzes N'_{aQu} ergaben sich weitere Einsparung von Zwischenzuständen im Erreichbarkeitsgraph des Modells. Das Netz N'_{aQu} bestand zuvor aus 98 Plätzen und 49 Transitionen. Mit Hilfe der Reduktion konnten wir das Netz N''_{aQu} in NR_3 durch eine einzige Transition modellieren. Offensichtlich erreichen wir auf diese Weise weitere Einsparungen von Zwischenzuständen im zu untersuchenden Erreichbarkeitsgraph und ermöglichen somit eine effizientere Erreichbarkeitsanalyse für die Verifikationsaufgaben 1 und 2. Das Netz NR_3 besteht insgesamt aus 34 Plätzen und 29 Transitionen. Tabelle 4.1 stellt die Anzahl der Zustände des jeweiligen Erreichbarkeitsgraphen aller in dieser Arbeit vorgestellten Netze dar. Für LoLA war es lediglich unter Anwendung der Reduktionstechnik Partial Order Reduction (POR) möglich, einen Erreichbarkeitsgraph mit 191713 Zuständen für das Skelett der Komposition NR_3 zu berechnen. INA brach die Berechnungen für NR_3 wegen Speicherüberlaufs frühzeitig ab. Darüber hinaus können wir die 3. Frage aus Kapitel 2.4 mittels der Analyse von NR_3 positiv beantworten. Allerdings ist es nicht möglich, die 4. Frage anhand dieses Modells zu untersuchen. Ob der Schaltkreis tolerant gegenüber Verzögerungen in der Asynchronen Queue ist, kann lediglich mit dem Modell NR_2 aus Kapitel 4.2 untersucht werden.

Die in Kapitel 4.4 vorgestellte Abstraktionsfunktion α verkleinert die Anfangsmarkierung der Komposition NR_4 . Auf diese Art erreichen wir eine effizientere Erreichbarkeitsanalyse für NR_4 , da in dem Zustandsgraph dieses Modells lediglich jeder achte Handshake modelliert ist. Somit sparen wir einen erheblichen Betrag an Zwischenzuständen im Erreichbarkeitsgraph ein. Entsprechend generierte LoLA ohne die Anwendung jeglicher Reduktionstechniken einen Erreichbarkeitsgraph mit 209008 Zuständen. Mittels Partial Order Reduction berechnet das Werkzeug einen Erreichbarkeitsgraph von nur 3099 Zuständen.

Tabelle 4.1.: Vergleich der Modelle bzgl. Anzahl der Transitionen und Plätze.

Netz	Plätze	Transitionen	Zustandsraum LoLA
NR_1	205	84	Abbruch nach ca. 14 Mio. Zuständen
NR_2	129	72	Abbruch nach ca. 15 Mio. Zuständen
NR_3	34	29	Abbruch nach ca. 20 Mio. Zuständen 191713 Zustände (mit POR)
NR_4	34	29	209008 Zustände 3099 Zustände (mit POR)

Trotz aller in dieser Arbeit vorgestellten Reduktions- und Abstraktionsmethoden ist es uns nicht möglich, den Zustandsraum des Modells derart zu reduzieren, dass Erreichbarkeitsanalysen mit INA möglich sind. Dieses Werkzeug bricht die Berechnung des Zustandsraums für alle in dieser Arbeit vorgestellten Modelle wegen Speicherüberlaufs ab.

5. Zusammenfassung und Ausblick

5.1. Zusammenfassung

Ziel dieser Arbeit war es, ein formales Modell der in [KG04] vorgestellten GALS-Schaltung zu erstellen und dieses mittels Abstraktion zu verkleinern, um insbesondere die Analyse nicht-funktionaler Eigenschaften des Receivers zu ermöglichen. In der vorliegenden Arbeit haben wir den GALS-Schaltkreis auf ein Intervall-Petrinetzmodell abgebildet, anhand dessen wir Verifikationsaufgaben formulieren konnten. Des Weiteren haben wir mehrere Reduktions- und Abstraktionsmöglichkeiten vorgestellt, um die Struktur des Modells zu verkleinern und die Größe des entsprechenden Zustandsraums zu reduzieren.

Zunächst erläuterten wir in Kapitel 2.1 die grundlegenden Prinzipien und Eigenschaften einer GALS-Schaltung und gingen auf deren Funktionsweise ein. Anschließend stellten wir in Kapitel 2.2 und 2.3 eine Anwendung des beschriebenen GALS-Schaltkreises vor: einen WLAN-Basisbandprozessor. Dieser Schaltkreis wurde bereits implementiert und getestet. Der Datenfluss dieser GALS-Schaltung war Gegenstand der Untersuchungen dieser Arbeit. In Kapitel 2.4 gingen wir auf die eigentliche Aufgabenstellung ein und zählten vier Fragen an den Schaltkreis auf, die durch reines Testen nicht beantwortet werden konnten. Im Mittelpunkt der Analyse nicht-funktionaler Eigenschaften standen Untersuchungen zu Zeitverhalten der Schaltung, Latenz im Datendurchsatz, Speichertiefen der Pipelines und Toleranzen gegenüber der Änderung der einzelnen Taktraten.

In Kapitel 3.1 gaben wir eine Einführung in die Petrinetz- und Intervall-Petrinetztheorie und definierten die formalen Grundlagen unseres Modells. In Kapitel 3.2 haben wir den Schaltkreis des Receivers auf ein Intervall-Petrinetzmodell abgebildet. Anhand dieses Modells konnten wir die vier Fragen aus Kapitel 2.4 als Verifikationsaufgaben am Modell formulieren. Insbesondere ist es uns in Kapitel 3.3 gelungen, die ursprünglichen Fragen auf Erreichbarkeitseigenschaften eines Petrinetzes zurückzuführen. Allerdings schränkte die Größe des Zustandsraums des vorgestellten Modells die computergestützte Verifikation des Schaltkreises erheblich ein. Ursache dafür ist neben der Nebenläufigkeit im Modell das explizite Modellieren der Zeit. Der Erreichbarkeitsgraph eines Intervall-Petrinetzes hat im Vergleich zu seinem Skelett in der Regel mehr Zwischenzustände, die sich durch die Transitionsmarkierung unterscheiden. Allein die Modellierung der Rückführung in

der Schaltung resultierte in einer zu großen Anzahl von Zwischenzuständen im Erreichbarkeitsgraph.

Dementsprechend diskutierten wir in Kapitel 4 einige Möglichkeiten, das vorliegende Modell zu verkleinern. Dazu spezifizierten wir in Kapitel 4.2 zunächst unabhängig vom Modell drei strukturelle Reduktioneregeln für Intervall-Petrinetze und wandten diese anschließend auf das Modell aus Kapitel 3.2 an. In Kapitel 4.3 und 4.4 konnten wir implizites Wissen über den GALS-Schaltkreis ausnutzen: Durch Beobachtung des regelmäßigen Verhaltens im Modell gelang es uns mittels Abstraktion weitere Teilnetze zu reduzieren. Die daraus resultierende Einsparung von Zwischenzuständen im Erreichbarkeitsgraph des Modells ermöglichte uns eine effizientere Analyse einiger Verifikationsaufgaben aus Kapitel 3.3. Insgesamt stellten wir mehrere Möglichkeiten vor, nicht-funktionale Eigenschaften einer GALS-Schaltung mittels Modellierung, Abstraktion und Verifikation zu analysieren.

5.2. Ausblick

Wir haben in dieser Arbeit lediglich einige Möglichkeiten aufgezählt, ein konkretes Modell zu abstrahieren. Ein weiterer Ansatz, das Modell aus Kapitel 4.4 zu untersuchen, ist die Abstraktion durch das Modellieren von Runden: Eine *Runde* beginnt bei Eintritt des ersten Symbols in den Schaltkreis und ist beendet, wenn das nächste Symbol eines Frames am Eingang anliegt. Insgesamt führt der Schaltkreis R Runden aus, wobei R die Anzahl der Symbole pro Frame ist. Mittels dieser Betrachtungsweise ist es möglich, eine rundenweise Verifikation auf kleineren Teilnetzen durchzuführen. Da das erste Symbol eines Frames erst zu Beginn der sechsten Runde am Eingang des Blocks B2 anliegt, schlagen wir vor, das Modell für die Runde 1 bis 5 (mit einem Netz R_1), 6 bis $R - 5$ (mit einem Netz R_2) und die letzten fünf Runden (mit einem Netz R_3) zu betrachten. Dabei sind die Endmarkierungen des Netzes R_1 die Anfangsmarkierungen des Netzes R_2 und die Endmarkierungen des Netzes R_2 die Anfangsmarkierungen des Netzes R_3 . Erfüllen alle Modelle in allen Runden alle zeitlichen Bedingungen (gilt beispielsweise in allen Abläufen $m(\text{error}) = 0$), so ist für Runde 1 bis R gezeigt, dass der Schaltkreis die zu untersuchende Eigenschaft erfüllt.

Wie oben beschrieben erlaubte uns die in Kapitel 4.3 vorgeschlagene Reduktion der Rückführung nicht, die Analyse der 4. Frage aus Kapitel 2.3 am reduzierten Modell vorzunehmen. Um zu ermitteln, ob die Schaltung insgesamt tolerant gegenüber allen möglichen Verzögerungen der Asynchronen Queue ist, fassen wir die Netze, die die Blöcke B2, B3 und B4 modellieren zu einer einzigen Transition t zusammen. Wir wählen die best

case-Verzögerung eines Tokens, das diese Blöcke einmal passiert, als $eft(t)$ und die worst case-Verzögerung als $lft(t)$. Wir beginnen die Analyse zu dem Zeitpunkt, zu dem das erste Symbol eines Frames in der Rückführung bereit liegt, also den Moment, in dem B2 als nächstes Token aus der Asynchronen Queue entgegen nimmt. Wir fragen wieder mittels eines Konsumenten-Netzes, ob es für die Transition t stets möglich ist, 2400 ns lang alle 50 ns eine Marke entgegenzunehmen, anschließend 1600 ns weder Marken produziert noch konsumiert und anschließend wieder 2400 ns lang alle 50 ns eine Marke zu konsumieren, usw. Der Vorteil dieser Methode ist, dass wir in der Anfangsmarkierung lediglich 48 Marken für einen gesamten Frame benötigen und während der Analyse stets die gleichen Marken für neue Symbole aus der Umgebung nutzen. Daraus ergibt sich eine signifikante Verkleinerung des Zustandsraums.

Weiterhin sind wir genaue Fallstudien mit Hilfe eines Verifikationswerkzeugs schuldig geblieben. Für das Modell in Kapitel 3.2 sind diese nicht möglich, da für dieses Netz ein zu großer Erreichbarkeitsgraph berechnet werden muss. Auch für die abstrakten Modell in Kapitel 3 ist dies nicht möglich. Insbesondere sind Verifikationstechniken (z.B. Partial Order Reduktion) die bei der Verifikation von Stellen-Transitions-Netzen sehr erfolgreich eingesetzt werden, für die Verifikation von Zeitpetrinetzen nicht verfügbar. Allerdings zeigt Popova-Zeugmann in [PZ06] neue Möglichkeiten auf, den Zustandsraum für Intervall-Petrinetze zu berechnen. Diese Reduktionen können die computergestützte Verifikation des vorgestellten Modells ermöglichen.

A. Glossar und Abkürzungen

- GALS-Block (Global-asynchroner lokal-synchroner Block bzw. asynchroner Wrapper mit synchronem Modul)
- WLAN (Wireless Local Area Network) bezeichnet ein „drahtloses“ lokales Funknetz, wobei ein IEEE 802.11-Standard gemeint ist.
- 80 MHz (Megahertz) entsprechen 80 Millionen Schwingungen pro Sekunde bzw. einer Schwingung pro 12,5 Nanosekunden. 20 MHz entsprechen einer Schwingung pro 50 Nanosekunden.
- 1 Msps (Mega Samples per Second) entspricht einer Millionen Abtastwerte pro Sekunde bzw. einer Millionen Schwingungen pro Sekunde.
- ns (Nanosekunde)
- ein Token = Datensignal (Bitvektor) in der Schaltung entspricht einer Marke im Petrinetz
- \mathbb{Q}_0^+ sind die positiven rationalen Zahlen einschließlich der Null
- LoLA (Low Level Analyzer) Verifikationswerkzeug für Stellen-Transitions-Netze, siehe [Sch00].
- INA (Integrated Net Analyzer) Verifikationswerkzeug für Intervall-Netze, siehe [Sta92].

Abbildungsverzeichnis

2.1. Aufbau und Struktur eines GALS-Blocks.	10
2.2. Übergänge der Phasen eines Wrappers.	11
2.3. Komponenten des WLAN-Basisbandprozessors	12
2.4. Datenfluss im Receiver in Mega Samples per Second (Msps).	14
3.1. N_1 ist ein Petrinetz.	23
3.2. Das Intervall-Petrinetz Z_1	29
3.3. Schnittstellenmodell der GALS-Blöcke des Receivers.	32
3.4. Schnittstellenmodell zweier GALS-Blöcke A und B.	33
3.5. Das Netz N_{Env}	34
3.6. Das Netz N_{B2}	35
3.7. Die Kapazität c	36
3.8. Das Netz N_{B3}	39
3.9. Das Netz N_{B4}	42
3.10. Das Netz N_{aQu}	43
3.11. Der Konsument N_K am Ausgang von N_{B4}	45
4.1. Strukturelle Reduktionsregel 1.	51
4.2. Das Netz N'_{aQu}	53
4.3. Strukturelle Reduktionsregel 2.	54
4.4. Das Netz N'_{Env}	55
4.5. Das Netz N'_{B2}	56
4.6. Strukturelle Reduktionsregel 3.	57
4.7. Das Netz N'_{B4}	58
4.8. Die Schnittstellenreduktion.	59
4.9. Illustration zu Satz 4.5.	63
4.10. Das Netz N' nach Anwendung der strukturellen Reduktionsregel 3.	64
4.11. Das Netz N''_{aQu}	65
4.12. Das Netz N''_{B2}	66
4.13. Das Netz N''_{B3}	67
4.14. Das Netz N''_{B4}	68
4.15. N''_{Env} im Netz NR_4	70

4.16. N'_k im Netz NR_4	71
4.17. Veranschaulichung zu Satz 4.7.	73
4.18. Das abstrahierte Netz N'''_{B2}	76
4.19. Das abstrahierte Netz N'''_{B3}	77
4.20. Das abstrahierte Netz N'''_{B4}	78

Tabellenverzeichnis

4.1. Vergleich der Modelle bzgl. Anzahl der Transitionen und Plätze.	80
--	----

Literaturverzeichnis

- [CES86] CLARKE, Edmund M. ; EMERSON, E. A. ; SISTLA, A. P.: Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 8 (1986), April, Nr. 2, S. 244–263. – ISSN 0164–0925
- [Cha84] CHAPIRO, Daniel M.: *Globally-asynchronous locally-synchronous systems*, Stanford University, Ph.D. Thesis, Oktober 1984
- [FD96] FURBER, Stephen B. ; DAY, Paul: Four-phase micropipeline latch control circuits. In: *IEEE Trans. Very Large Scale Integr. Syst.* 4 (1996), Nr. 2, S. 247–253. – ISSN 1063–8210
- [Jul05] JULIUS, Alexandra: *Entwurf und VHDL-Modellierung von mesochronen GALS-Schaltungen*, Humboldt-Universität zu Berlin, Studienarbeit, Dezember 2005
- [KG04] KRSTIĆ, Miloš ; GRASS, Eckhard: GALSification of IEEE 802.11a Baseband Processor. In: MACH, Enrico (Hrsg.) ; KOUFOPAVLOU, Odysseas G. (Hrsg.) ; PALIOURAS, Vassilis (Hrsg.): *Proceedings of the 14th International Workshop on Integrated Circuit and System Design, Power and Timing Modeling, Optimization and Simulation (PATMOS 2004)*. Santorini, Greece : Springer-Verlag, 2004 (LNCS 3254), S. 258–267
- [KGS05] KRSTIC, Milos ; GRASS, Eckhard ; STAHL, Christian: Request-Driven GALS Technique for Wireless Communication System. In: *Proceedings of the 11th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC 2005)*. New York, NY, USA : IEEE Computer Society, März 2005. – ISBN 0–7695–2305–6, S. 76–85
- [KGSP06] KRSTIĆ, Miloš ; GRASS, Eckhard ; STAHL, Christian ; PIZ, Maxim: System Integration by Request-driven GALS Design. In: *IEE Proc. Computers & Digital Techniques* 153 (2006), September, Nr. 5, S. 362–372
- [LMW07] LOHMANN, Niels ; MASSUTHE, Peter ; WOLF, Karsten: Operating Guidelines for Finite-State Services. In: *Petri Nets and Other Models of Concurrency - ICATPN 2007, 28th International Conference on Applications and Theory of*

- Petri Nets and Other Models of Concurrency, Siedlce, Poland, June 25-29, 2007, Proceedings, 2007.* – accepted
- [Mil71] MILNER, R.: An Algebraic Definition of Simulation between Programs. In: *Proc. of the 2nd IJCAI*. London, UK : William Kaufmann, 1971, S. 481–489
- [Pee96] PEETERS, Ad M. G.: *Single-Rail Handshake Circuits*, Technische Universiteit Eindhoven, Ph.D. Thesis, Juni 1996
- [PZ06] POPOVA-ZEUGMANN, Louchka: *Zeit und Petrinetze*, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Habilitationsschrift, 2006. – unveröffentlicht
- [Rei85] REISIG, Wolfgang: *Petrinetze, Eine Einführung*. Springer-Verlag, 1985
- [Sch00] SCHMIDT, Karsten: LoLA: A Low Level Analyser. In: NIELSEN, Mogens (Hrsg.) ; SIMPSON, Dan (Hrsg.): *Application and Theory of Petri Nets, 21st International Conference (ICATPN 2000)*, Springer-Verlag, Juni 2000 (LNCS 1825), S. 465–474
- [SRK05] STAHL, Christian ; REISIG, Wolfgang ; KRSTIĆ, Miloš: Hazard Detection in a GALS Wrapper: a Case Study. In: DESEL, Jörg (Hrsg.) ; WATANABE, Y. (Hrsg.): *Proceedings of the Fifth International Conference on Application of Concurrency to System Design (ACSD'05)*. St. Malo, France : IEEE Computer Society, Juni 2005, S. 234–243
- [Sta90] STARKE, Peter H.: *Analyse von Petri-Netz-Modellen*. Stuttgart : B.G. Teubner Verlag, 1990
- [Sta92] STARKE, Peter H.: *INA Integrated Net Analyzer*. Berlin : Handbuch, 1992
- [VHD87] *IEEE Standard VHDL Language Reference Manual*, März 1987. – IEEE Standard 1076-1987

Danksagung

Ich danke dem gesamten Lehrstuhl für Theorie der Programmierung für die vielen Anregungen und die konstruktive Kritik. Miloš Krstić danke ich für seine Hilfe beim Verständnis des Receivers.

Ganz besonders möchte ich meiner Familie danken für alle nur denkbare Unterstützung.

Vielen Dank!

Erklärung

Hiermit erkläre ich, die vorliegende Arbeit „*Abstrakte Datenflussmodelle für GALIS-Schaltungen zum Nachweis nicht-funktionaler Eigenschaften*“ selbstständig und ohne fremde Hilfe verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet zu haben.

Weiterhin erkläre ich hiermit mein Einverständnis, dass die vorliegende Arbeit in der Bibliothek des Instituts für Informatik der Humboldt-Universität zu Berlin ausgestellt werden darf.

Berlin, den 14. Mai 2007

