# Partitionierungsalgorithmen für Modelldatenstrukturen zur parallelen compilergesteuerten Logiksimulation (Projekt)

*Klaus Hering*

Universität Leipzig
Institut für Informatik
Augustusplatz 10-11
04109 Leipzig (Deutschland)
khering@informatik.uni-leipzig.de

Report Nr. 5

Juli 1994

## Zusammenfassung

Die enorme Komplexität in naher Zukunft absehbarer VLSI-Schaltkreisentwicklungen bedingt auf allen Entwurfsebenen sehr zeitintensive Simulationsprozesse. Eine Antwort auf diese Herausforderung besteht in der Parallelisierung dieser Prozesse. Es wird ein Forschungsvorhaben vorgestellt, welches auf eine effektive Partitionierung von Modelldatenstrukturen im Vorfeld compilergesteuerter Logiksimulationen auf Parallelrechnern mit lose gekoppelten Prozessoren gerichtet ist. Im Rahmen dieses Projekts sollen ausgehend von einem Graphen-Modell Partitionierungsalgorithmen entwickelt, theoretisch untersucht sowie Kriterien für ihren Einsatz in Abhängigkeit von anwendungstypischen Modelleigenschaften abgeleitet werden. Zur experimentellen Unterstützung ist die Entwicklung einer parallel arbeitenden Testumgebung für die Analyse relevanter Modelldatenstrukturen vorgesehen. Eine Erweiterung dieser Umgebung zu einer Softwarekomponente, welche im Ergebnis eines Präprocessing von Modelldatenstrukturen Partitionierungsalgorithmen auswählt und realisiert, soll schließlich in ein System zur Logiksimulation auf der Basis parallel arbeitender Instanzen eines der führenden heute kommerziell verfügbaren funktionellen Logiksimulatoren eingebunden werden.
Für dieses Projekt wurde bei der DFG ein Antrag auf Förderung im Rahmen des Schwerpunktprogrammes *Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen* gestellt.

**Schlüsselwörter:** VLSI-Design, Parallele Logiksimulation, Modellpartitionierung

# 1 Einleitung

VLSI-Entwürfe mit 50 bis 100 Millionen Transistoren pro Chip werden in nicht allzu ferner Zukunft möglich sein; Entwürfe dieser Komplexität erfordern zur Gewährleistung kurzer Entwicklungszeiten und hoher Zuverlässigkeit der resultierenden Schaltkreise eine durchgängige Unterstützung des Designprozesses durch leistungsfähige, gut aufeinander abgestimmte Programmsysteme. Das Zusammenspiel der verschiedenen Designphasen bei der Entwicklung eines Mikroprozessors ist anhand eines konkreten Projekts (*Capitol Chip Set*) in [Spr89] dargestellt. Zur Vermeidung kosten- und zeitintensiver Arbeiten mit fehlerbehafteten Zwischenstrukturen ist eine Begleitung sämtlicher Entwurfsphasen durch Verifikationsprozesse unumgänglich.

Die Hauptform der VLSI-Designverifikation ist die Logiksimulation. Auf der Basis von Modellen verschiedener Abstraktionsstufen (*circuit-, switch-, gate-, register transfer-, functional-, instruction level*) wird das Verhalten zu entwerfender Schaltkreise in Abhängigkeit von vorgegebenen Stimuli nachgebildet und ausgewertet. Häufig sind Elemente mehrerer Abstraktionsstufen im Rahmen von Modellen kombiniert. Aspekte der Mehrebenensimulation sind in [PoS92] zu finden. Hinsichtlich der Strategie der Evaluierung funktioneller Elemente lassen sich im wesentlichen zwei Arten der Logiksimulation unterscheiden [Pos89]: compilergesteuerte (*time driven, compiled mode*) und ereignisgesteuerte (*event driven*) Simulation. Im ersten Fall werden in der Regel bei äquidistant fortschreitender Simulationszeit sämtliche funktionellen Elemente des betrachteten Modells evaluiert. Dagegen ist im zweiten Fall eine Elementevaluierung an die Änderung relevanter Eingangssignale (*events*) gebunden, womit wiederholte Elementevaluierungen bei unveränderter Eingangssignalsituation ausbleiben. Dem Vorteil der zweiten Strategie in Form vergleichsweise geringerer Anzahlen von Elementevaluierungen steht als Vorteil der erstgenannten Strategie der Wegfall des Scheduling-Overheads für potentiell evaluierbare Elemente gegenüber.

Die zunehmende Komplexität der Simulationsmodelle einerseits (insbesondere bei niedrigem Abstraktionsgrad) und der Stimuli andererseits (z.B. bei Simulation eines kompletten *Initial Program Load* auf Register-Transfer-Level für einen Prozessorentwurf) führen bei Verwendung sequentiell arbeitender Simulationssysteme zu Ausführungszeiten, die unter dem Gesichtspunkt vernünftiger Entwicklungszeiten für VLSI-Chips nicht mehr vertretbar sind. Untersuchungen zur Abhängigkeit von Simulationszeiten vom Abstraktionsgrad des betrachteten Modells sind für eine geringe Zahl von Beispielentwürfen in [SoB87] dargestellt. Eine Antwort auf das starke Anwachsen von Simulationszeiten besteht in der Parallelisierung der Simulation.

In Kapitel 2 dieses Reports wird ausgehend von einer Skizze des Standes der Forschung auf dem Gebiet der Parallelen Logiksimulation in kurzer Form das Konzept eines parallelen Logiksimulationssystems auf der Basis des funktionellen compilergesteuerten Simulators *MaxSim* (IBM) beschrieben. Dieses System ist als unmittelbares praktisches Umfeld unseres Projekts zur Modellpartitionierung anzusehen, welches nach einführenden Bemerkungen zu bestehenden Partitionierungsstrategien in Kapitel 3 im Detail vorgestellt wird.

# 2 Parallele Logiksimulation

## 2.1 Stand der Forschung

In den letzten Jahren hat es eine Reihe von Ansätzen zur Parallelisierung der Logiksimulation gegeben. Eine diesbezügliche Übersicht ist in [Mei93] zu finden. Folgt man [MSD93], sind für die Logiksimulation folgende Parallelitätsaspekte von Bedeutung:

- **Algorithmeninhärente Parallelität**
  Diese Form der Parallelität hat ihren Ursprung darin, daß unabhängig von der betrachteten Abstraktionsebene immer wiederkehrende Grundelemente von Simulationsalgorithmen (Behandlung von globalen Ein- und Ausgaben, Berechnung von Ausgabewerten, Scheduling von fan-out Elementen) überlappend innerhalb von Pipelines realisiert werden können.

- **Dateninhärente Parallelität**
  Die Durchführung der Logiksimulation für mehrere Stimuli bezüglich eines Modells beinhaltet ein (trivial zu nutzendes) Parallelitätspotential in dem Sinn, daß verschiedene Läufe eines Simulationsprogramms voneinander unabhängig auf unterschiedlichen Prozessoren/Rechnern durchgeführt werden können. In [Bry88] wird die Möglichkeit der Ausnutzung der Bitparallelität logischer Operationen auf Maschinenebene bei konventionellen Prozessoren sowie der Einsatz eines massiv-parallelen SIMD-Systems (*Connection Machine*) zur simultanen Bearbeitung von Stimuli behandelt. Unter den hier betrachteten Parallelitätsaspekt fällt auch das Zerlegen umfangreicher Stimuli und die parallele Bearbeitung der resultierenden Teile durch verschiedene Instanzen eines Simulationssystems auf unterschiedlichen Prozessoren. Bei dieser Vorgehensweise ergibt sich das Problem, an Schnittstellen der Simulation Modellzustände vorwegnehmen zu müssen.

- **Modellinhärente Parallelität**
  Die Parallelität von Signalpfaden im zugrundeliegenden Modell bietet einen Parallelisierungsansatz für die Logiksimulation durch Partitionierung des betreffenden Modells und Kooperation mehrerer Instanzen eines Simulationssystems über den erhaltenen Modellbestandteilen. Der Effekt dieses Vorgehens hängt wesentlich von der Lastverteilung auf die beteiligten Prozessoren (Load Balancing) und dem Overhead für erforderliche Synchronisationsmaßnahmen ab. Nach [MSD93] gewinnt die Modellpartitionierung nicht nur unter dem Zeitaspekt, sondern auch unter dem räumlichen Aspekt der Speicherung extrem umfangreicher Schaltkreismodelle stark an Bedeutung.

Die algorithmeninhärente Parallelität verkörpert innerhalb der drei betrachteten Fälle den geringsten Ansatzpunkt zur Senkung der Simulationszeiten für komplexe Entwürfe. Hinsichtlich des Einsatzes von SIMD-Architekturen zur simultanen Bearbeitung einer großen Zahl von Stimuli ist zu bemerken, daß dabei zwar ein hoher Parallelitätsgrad erreichbar ist, dafür aber starke Einschränkungen bezüglich der einsetzbaren Stimuli bestehen. Unser Forschungsvorhaben ist auf die Ausnutzung der modellinhärenten Parallelität bei Einsatz lose gekoppelter universeller Prozessoren gerichtet.

Eine Reihe von Ansätzen zur Parallelisierung der Logiksimulation basiert auf der Entwicklung speziell dafür vorgesehener Hardware (*Hardware-Simulatoren*). Dazu gehört beispielsweise die *Yorktown Simulation Engine (YSE)* [Pfi82]. Die industriellen Erfahrungen mit Hardware-Simulatoren sind jedoch nicht überzeugend. Man erreicht bei ihrem Einsatz zwar

gute Speed-Up-Werte, aber in Bezug auf rasch erforderliche Änderungen der Simulationstechnik sind Hardware-Simulatoren sehr unflexibel. Sie veralten schnell und sind aufgrund ihres hohen Entwicklungsaufwandes sehr teuer. Die IBM-Laboratorien in Böblingen haben bei ihren VLSI-Entwicklungen seit Beginn der 80-iger Jahre auf den Einsatz von Hardware-Simulatoren verzichtet. Software-Simulationssysteme als Gegenstück bieten den Vorteil, ohne aufwendige Anpassungen innerhalb kurzer Zeit auf neuester und schnellster Prozessortechnologie lauffähig zu sein.

Bei den Arbeiten zur Parallelisierung ereignisgesteuerter Logiksimulationssysteme scheinen insbesondere die auf der von *Jefferson* eingeführten *Time-Warp*-Methode basierenden Ansätze erfolgversprechend [BSK91]. Dabei arbeiten Simulatorinstanzen ohne direkte Synchronisation auf der Basis einer lokalen Zeit parallel über Modellpartitionen. Partitionsgrenzen überschreitende Signale werden, mit Zeitmarken versehen, zwischen den entsprechenden Simulatorinstanzen vermittelt und können, sofern ihre Zeitmarke den aktuellen Wert der lokalen Zeit in der Zielinstanz unterschreitet, zu einem Rücksetzen (*Roll Back*) des Simulationsprozesses in der Zielinstanz (und gegebenenfalls in weiteren Instanzen) führen. Die erforderliche Behandlung von *Roll Backs* führt allerdings zu einem erheblichen Overhead. Untersuchungsergebnisse zur aktivitätsbezogenen Parallelität von Modellen (etwa in Form der durchschnittlichen Anzahl während eines Zeitschrittes behandelter Ereignisse) bei ereignisgesteuerter Logiksimulation sind in [Bai92] zu finden.

Die Entwicklung führender Mikroprozessor-Architekturen (z.B. *RS/6000* (IBM,Motorola), *ESA/390* (IBM)) wurde und wird erfolgreich von compilergesteuerten Logiksimulationssystemen begleitet. Den konkreten Hintergrund für die Orientierung unserer Arbeit auf den compilergesteuerten Ansatz verkörpert der sehr leistungsfähige (sequentielle) Simulator *MaxSim* (IBM) zur funktionellen Logiksimulation (vorgestellt auf der Design Automation Conference 94 in San Diego). Dieser Simulator arbeitet über Modellen synchroner Schaltkreise nach dem *clock-cycle*-Algorithmus. Dabei werden auf Modellebene logische Elemente (verzögerungsfrei) im *zero-delay*-Modus und als Latches interpretierbare Elemente im *unit-delay*-Modus (Verzögerung der Ausgangswerte um 1 Zeiteinheit) betrachtet.

## 2.2 Parallelisierung der Logiksimulation auf *MaxSim* -Basis

Wir beabsichtigen die Implementierung eines parallelen Logiksimulationssystems, bei dem Simulationen über Modellen komplexer VLSI-Entwürfe durch kooperierende *MaxSim* -Instanzen realisiert werden, welche über Modellpartitionen auf verschiedenen Prozessoren eines lose gekoppelten Systems arbeiten. Als Zielhardware dienen switchgekoppelte RISC/6000-Prozessoren (SP1, SP2 von IBM). Das Vorhaben kann grob wie folgt untergliedert werden:

- **Realisierung der Modelldatenaufbereitung**
  Bei der Modellbildung zur Simulation mittels *MaxSim* wird von Schaltkreisbeschreibungen in den Sprachen BDL/S (Gate-Ebene) und DSL (Register-Transfer-Ebene) ausgegangen. Diese Beschreibungen werden in Datenstrukturen übersetzt, die von der Softwarekomponente *DA_DB* (IBM) manipuliert und analysiert werden können. Momentan wird für die Simulation mittels *MaxSim* ausgehend vom Übersetzungsergebnis jeweils ein Modell erzeugt.

Für die Parallelisierung der Simulation soll die im Ergebnis der erwähnten Übersetzung entstehende Zwischenstruktur unter Beachtung von Synchronisations- und Load-Balancing-Aspekten auf der Basis von *DA_DB* -Manipulationsprimitiven partitioniert werden. Ausgehend von den Partitionen sollen dann Modellteile für die Bearbeitung durch *MaxSim* -Instanzen auf einzelnen Prozessoren einschließlich erforderlicher Synchronisationsinformationen generiert werden.

Die Realisierung der Modelldatenaufbereitung erfolgt in zwei Schritten. Zunächst wird ein sich momentan in Entwicklung befindender Partitionierungsalgorithmus auf *fan-in cone* -Basis implementiert. Diese feste Implementierung soll später durch die im Ergebnis des Forschungsvorhabens *Partitionierungsalgorithmen für Modelldatenstrukturen zur parallelen compilergesteuerten Logiksimulation* entstehende Partitionierungskomponente mit Möglichkeiten der modellspezifischen Algorithmenwahl ersetzt werden.

- **Entwicklung prozessorbezogener Laufzeitrahmen für *MaxSim* -Instanzen**
  Zur Realisierung des Zusammenspiels parallel arbeitender *MaxSim* - Instanzen sollen in C zwei verschiedene Shellkomponenten *MPRF (MainProcessorRuntimeFrame)* und *SPRF (SecondaryProcessorRuntimeFrame)* entwickelt werden. Eine Modifikation von *MaxSim* selbst ist nicht vorgesehen. *MPRF* soll gegenüber *SPRF* eine erweiterte Funktionalität hinsichtlich der Initialisierung und Termination des Gesamtsystems und der Kommunikation des Systems mit seiner Umwelt aufweisen. Das *MaxSim* -Applikationsinterface wird für die parallele Implementierung erweitert.

- **Entwicklung eines Kommunikationsinterface**
  Wesentlichen Einfluß auf durch die Parallelisierung erreichbare Speed-Up-Werte hat die Realisierung der Kommunikation zwischen den beteiligten Prozessoren. Ausgehend von einer Bestandsaufnahme der verfügbaren Werkzeuge zur Prozeßsynchronisation und -kommunikation soll für *MPRF* und *SPRF* ein Kommunikationsinterface in Bibliotheksform bereitgestellt werden.

Für die Behandlung sehr komplexer Modelle werden substantielle Laufzeitverkürzungen erwartet, durch die für die VLSI-Entwurfsverifikation interessante, aber im Moment nicht in zufriedenstellender Zeit realisierbare Simulationen in den Bereich der Ausführbarkeit gelangen würden.

# 3 Modellpartitionierung im Vorfeld der parallelen Logiksimulation

## 3.1 Projektansatzpunkte

Die Modellpartitionierung im Vorfeld der parallelen Logiksimulation hat das Ziel, für die Bearbeitung durch einzelne Prozessoren vorgesehene Modellbestandteile zu fixieren. Maßgeblich für die Güte einer Partitionierung hinsichtlich des Speed-Up der darauf aufbauenden parallelen Simulation ist die Minimierung des aus dem Schneiden von Signalpfaden resultierenden Synchronisations-/Kommunikationsoverheads und eine ausgewogene Lastverteilung auf die beteiligten Prozessoren. [MSD93] zeigt auf der Grundlage bestimmter graphentheoretischer Modelle die Formulierung des Partitionierungsproblems als Optimierungsproblem und seine Rückführbarkeit auf das NP-vollständige allgemeine Graphen-Partitionierungsproblem. Bei sehr komplexen zugrundeliegenden Modellen kommen zum Erhalt suboptimaler Lösungen eines derartigen Problems im wesentlichen heuristische bzw. stochastische Verfahren in Frage.

Sehr einfache Partitionierungsverfahren gehen von einer gleichmäßigen Verteilung atomarer Modellbestandteile entsprechend einer vorliegenden Ordnung oder von einer zufälligen Elementverteilung auf Partitionen aus. Dabei haben bestehende Signalpfade keinen Einfluß auf die resultierenden Modellzerlegungen; letztere sind höchstens als initiale Lösungen für anschließende iterative Verfahren von Interesse.

Einer Reihe von Verfahren liegt das Prinzip zugrunde, innerhalb betrachteter Ausgangsmodelle zunächst auf bestimmte Weise über Signalpfade zusammenhängende atomare Elemente für die weitere Betrachtung zu neuen Objekten (Corollas [SpB93], Cones [MSD93]) zusammenzufassen und ausgehend von diesen komplexeren Objekten (eventuell über mehrere Stufen) Modellpartitionen zu bilden. Solchen Partitionierungsverfahren mit *bottom-up* -Vorgehensweise stehen zum Beispiel die dem *top-down* -Prinzip gehorchenden *Min-Cut* -Verfahren [FiM82] gegenüber.

Zu den für die Modellpartitionierung im Kontext der parallelen Logiksimulation interessanten stochastischen Verfahren zählen *Simulated Annealing* und *Genetische Algorithmen* [HoS92]. Diese Verfahren wurden bereits erfolgreich zur Lösung von Optimierungsproblemen im Layout-Design eingesetzt. Sie sind allerdings mit einem sehr hohen Zeitaufwand verbunden. Dieser Nachteil kann in gewissem Grad durch Parallelisierung der Verfahren gemindert werden.

Neben statischen Partitionierungsstrategien (nach der Partitionierung erfolgt ein entsprechendes Mapping der Modellbestandteile, welches über den gesamten Zeitraum einer betrachteten Simulation konstant bleibt) werden in der Literatur auch dynamische Strategien betrachtet [KrA88,SiO93]. Dabei können bestimmte während eines Simulationsprozesses eintretende Zustände (z.B. starke Abweichungen in der Rechenlast beteiligter Prozessoren oder hohes Kommunikationsaufkommen zwischen Prozessoren) zu einer Re-Partitionierung und entsprechendem Re-Mapping führen. Diese Vorgehensweise bringt unseres Erachtens aufgrund eines erheblichen Overheads insbesondere für unser Vorhaben auf der Basis compilergesteuerter Simulationsprozesse kaum Vorteile gegenüber statischen Strategien.

Ausgehend vom gegenwärtigen Stand der Forschung sehen wir folgende Ansatzpunkte für unser Vorgehen:

- In der Literatur ist eine Vielzahl von Arbeiten zur Modellpartitionierung unter dem Gesichtspunkt der Parallelisierung ereignisgesteuerter Logiksimulationen zu verzeichnen, während der compilergesteuerte Ansatz in diesem Kontext deutlich weniger Betrachtung findet. Gründliche Untersuchungen in der letztgenannten Richtung halten wir vor dem Hintergrund des erfolgreichen Einsatzes von compilergesteuerten Simulatoren in der Praxis für erforderlich.

- Ein wesentliches Potential bei der Behandlung von Partitionierungsproblemen liegt in der Ausnutzung anwendungsspezifischer Modelleigenschaften. Einer Fixierung in diesem Kontext relevanter Eigenschaften und ihrer Ausnutzung zur Auswahl bestehender bzw. Entwicklung neuer Partitionierungsalgorithmen sollte verstärkt Aufmerksamkeit geschenkt werden.

- Die Verwendbarkeit einer Reihe interessanter Partitionierungsverfahren bei der Bearbeitung umfangreicher Modelle ist angesichts hoher Zeitkomplexitäten eingeschränkt. Mit dem Ziel der Erweiterung der Praktikabilität derartiger Verfahren sollten Untersuchungen zu ihrer Parallelisierung durchgeführt werden.

## 3.2 Projektziele

Unser Projekt hat die *Untersuchung, Entwicklung und Realisierung von Partitionierungsalgorithmen* für Modelldatenstrukturen im Vorfeld paralleler compilergesteuerter Logiksimulationsprozesse zum Ziel. Die Ergebnisse sollen unmittelbar in die Entwicklung des unter Abschnitt 2.2 skizzierten parallelen Simulationssystems auf *MaxSim* -Basis einfließen. Dabei hat die Modellpartitionierung entscheidenden Einfluß auf den erreichbaren Speed-Up.

Die Ziele des Vorhabens werden wie folgt konkretisiert:

- **Untersuchung und Entwicklung von Partitionierungsalgorithmen im Kontext anwendungsspezifischer Modelleigenschaften**

  Auf der Grundlage eines Performance-Modells, in welches Evaluierungskosten für logische Elemente und Kosten für erforderliche Synchronisationsmaßnahmen wesentlich eingehen, sollen Gütekriterien für Partitionierungen von Graphen-Modellen komplexer VLSI-Entwürfe unter dem Aspekt der Parallelisierung compilergesteuerter Logiksimulationsprozesse abgeleitet werden. Vor dem Hintergrund dieser Kriterien sind vorhandene Partitionierungsstrategien sowie Wirkungen ihrer Kombination zu analysieren. Dabei wird angestrebt, Beziehungen zwischen Eigenschaften von Modellen relevanter VLSI-Entwürfe und der zu erwartenden Güte bei Anwendung bestimmter Algorithmen entstehender Partitionen zu fixieren. Die Ergebnisse dieser Untersuchungen sind als Grundlage für die Realisierung einer Partitionierungskomponente anzusehen, welche aufbauend auf einem Modell-Präprocessing unter bestimmten Voraussetzungen in der Lage ist, aus einer vorgegebenen Menge von Partitionierungsstrategien eine günstige zu wählen.

Darüber hinaus sollen anwendungstypische Modelleigenschaften als Ausgangspunkt für die Entwicklung neuer Partitionierungsalgorithmen betrachtet werden. Dabei ist insbesondere die Untersuchung hinsichtlich der funktionellen Logiksimulation invarianter Modelltransformationen als Vorstufe von Partitionierungsprozessen vorgesehen.

Hinsichtlich der Praktikabilität von Partitionierungsprozessen für komplexe Modelle sollen Untersuchungen zur Parallelisierung ausgewählter Algorithmen durchgeführt werden.

- **Entwicklung einer parallelen Testumgebung für Partitionierungsalgorithmen**

  Die Entwicklung der Testumgebung soll in 2 Stufen (sequentielle und parallele Version) auf der Basis des bereits verfügbaren Systems *DA_DB* (Design Automation DataBase, IBM) erfolgen. *DA_DB* dient der Manipulation und Analyse von Schaltkreismodellen auf Register-Transfer- bzw. Gate-Ebene . Die Testumgebung soll auf *SP1*-Systemen implementiert werden. Sie bietet für die im vorangehenden Punkt genannten Untersuchungen ein experimentelles Umfeld. In engem Zusammenhang zum geplanten Vorgehen bei der Realisierung des im Hintergrund stehenden parallelen Logiksimulationssystems auf *MaxSim* -Basis sollen für *DA_DB* Shellkomponenten und ein Kommunikationsinterface entwickelt werden. Eine Möglichkeit der Konvertierung verschiedener Modelldarstellungen ist vorgesehen.

- **Entwicklung einer Softwarekomponente zur Realisierung der Modellpartitionierung und deren Einbindung in ein paralleles Logiksimulationssystem**

  Die genannte Komponente ist als Erweiterung der oben genannten parallelen Testumgebung geplant. Sie soll Modellanalysen mit dem Ziel der Auswahl anzuwendender Partitionierungsalgorithmen gestatten und die eigentliche Ausführung der Modellpartitionierung übernehmen. Ergebnisse der Partitionierung sind einerseits für die Zuordnung zu einzelnen Prozessoren vorgesehene Teile des Ausgangsmodells und andererseits Informationen über erforderliche Synchronisationsbeziehungen in die Simulation einbezogener Prozessoren. Zur Bereitstellung dieser Ausgangsdaten für die parallele Simulation auf *MaxSim* -Basis ist die Partitionierungskomponente in das betreffende Logiksimulationssystem einzubinden. Im Rahmen dieses Systems sind Experimente unter Verwendung relevanter Schaltkreismodelle zum Vergleich vorangehender auf theoretischer Basis erhaltener Speed-Up-Abschätzungen mit tatsächlich erreichten Werten vorgesehen.

### 3.3 Aktivitäten zur Projektrealisierung

Nachfolgend soll unser Vorhaben in einzelne Aktivitäten unter Annahme der Einbeziehung von zwei wissenschaftlichen Mitarbeitern (M1,M2) und drei studentischen Hilfskräften (S1,S2, S3) untergliedert werden. Eine schematische Darstellung des Zusammenspiels der einzelnen Aktivitäten ist auf Seite 12 zu finden. Basiszeitraum für unsere Betrachtungen ist der Antragszeitraum (bezüglich Förderung durch die DFG) von 24 Monaten.

(1) Bestimmung des aktuellen Standes der Forschung
Ausführung: M1,M2

- ausgehend vom Stand unserer Voruntersuchungen Analyse der weiteren Entwicklung der Forschung auf dem Gebiet der Parallelen Logiksimulation und speziell der Modellpartitionierung in deren Vorfeld, Verschaffen einer sehr detaillierten Übersicht zu bestehenden Partitionierungsalgorithmen
- Kontaktaufnahme zwecks Koordination und Kooperation zu Forschergruppen, die auf eng verwandtem Gebiet arbeiten

(2) Aufbereitung des Parallel Programming Environment *PPE* der *SP1*
Ausführung: S1

- Verschaffen einer Übersicht zu Möglichkeiten der Kommunikation und Synchronisation paralleler Prozesse auf switchgekoppelten SP1 - Systemen
- Installation und Test verfügbarer Software zur Simulation von SP1 -Systemen auf beantragter *RISC/6000* -Workstation
- über Netz Vorbereitung späterer Sitzungen an dem gewählten *SP1* -Zielsystem (GMD St.Augustin oder DESY Zeuthen), S1 übernimmt Verantwortung für diesen Zugang

(3) Aufbereitung der Softwarekomponente *DA_DB*
Ausführung: S2

- Installation und Test des verfügbaren *DA_DB* -Systems auf beantragter *RISC/6000* Workstation
- Realisierung elementarer Operationen zur Modellanalyse und -manipulation auf der Basis von *DA_DB* -Primitiven

(4) Vorbereitungen für die Arbeit mit Schaltkreismodellen
Ausführung: S3

- Verschaffen einer Übersicht zu verfügbaren relevanten Modellen und ihrer internen Darstellung (Internet-Nutzung, Kontakte zu Dr. Roesner (IBM Austin, Texas) und anderen Forschungseinrichtungen)
- Erstellen von Konvertierungsprogrammen für unterschiedliche Modellformate mit dem *DA_DB* -Format als Zielformat

(5) Konzipierung der Testumgebung für die Untersuchung von Partitionierungsalgorithmen
Ausführung: M1, M2

- Festlegung einer (später) erweiterbaren Grundmenge von Modellanalyse- und Manipulationsprimitiven
- Spezifikation einer Shellkomponente für *DA_DB* zur Realisierung der sequentiellen Version der Testumgebung, (*DA_DB* verkörpert den Kern der Umgebung und wird selbst nicht modifiziert)
- Festlegung (auf der erwähnten Grundmenge basierender) Manipulations- und Analyseprimitiven für die parallele Version der Testumgebung (mehrere mit einer Shell versehene kooperierende *DA_DB* -Komponenten laufen auf verschiedenen Prozessoren eines *SP1* -Systems)

- Spezifikation verschiedener Shellkomponenten für *DA_DB* und eines Kommunikationsinterface zur Realisierung der parallelen Version der Testumgebung

(6) Realisierung der sequentiellen Version der Testumgebung

- Realisierung der für die sequentielle Version vorgesehenen Analyse- und Manipulationsprimitiven
  Ausführung: S2

- Realisierung der für die sequentielle Version vorgesehenen Shellkomponente
  Ausführung: S3

(7) Bereitstellung einer Simulationsumgebung für die Entwicklung der parallelen Version der Testumgebung
  Ausführung: S1

- Realisierung des Kommunikationsinterface (für Prozessorgrenzen überschreitende Kommunikation) auf der Basis stark abgerüsteter Shellrahmen als Simulation auf der *RISC/6000* -Workstation

- Test der Realisierung auf realem *SP1* -System über Netz

(8) Untersuchung bestehender Partitionierungsalgorithmen
  Ausführung: M1

- Fixieren des Begriffs Güte von Partitionierungen auf der Basis eines Performance-Modells, in welches Evaluierungskosten für logische Elemente und Synchronisations-/ Kommunikationskosten in Bezug auf Prozessoren eingehen, die bei der parallelen Logiksimulation über einzelnen Modellpartitionen arbeiten und miteinander kooperieren

- gezielte Untersuchung der bekannten Partitionierungsstrategien hinsichtlich besonderer Relevanz im Vorfeld der compilergesteuerten Logiksimulation, dabei spezielle Betrachtungen für die der Simulation mittels *MaxSim* zugrundeliegende Modellklasse

- spezielle Untersuchungen von *bottom-up* -Strategien mit *Cones* als erster Form der Zusammenfassung elementarer Modellbestandteile, Untersuchungen zur Minimierung der bei Coneclusterbildung entstehenden Überlappungsbereiche, Variation der Techniken der Clusterbildung, Betrachtung verschiedener Strategien der Zusammenfassung elementarer Modellbestandteile

- Betrachtung von Möglichkeiten, Elemente verschiedener Partitionierungsstrategien zu kombinieren

- Untersuchungen zur Zeitkomplexität von Partitionierungsalgorithmen

(9) Untersuchung von Beziehungen zwischen Modelleigenschaften und der Güte von Partitionierungen
  Ausführung: M1

- systematische Suche bestimmte Partitionierungsstrategien stark beeinflussender Modelleigenschaften, welche unter Einbeziehung schaltkreisspezifischen Wissens auch für eine große Zahl von Modellen gesichert sind (enger Kontakt zur Schaltkreisentwicklung erforderlich !)

- Abschätzung des für parallele Simulationen ausgehend von erhaltenen Partitionierungen zu erwartenden Speed-Up in Abhängigkeit von verschiedenen Parametern auf der Basis eines Performance-Modells (in dieser Phase bereits Nutzbarkeit der sequentiellen Version der Testumgebung)

- vor dem Hintergrund der funktionellen Logiksimulation auf *MaxSim* - Basis Auswahl einer Menge von Partitionierungsalgorithmen zur Realisierung in der geplanten Partitionierungskomponente

- Entwicklung eines Algorithmus für das Modellpräprocessing, dessen Resultate unter bestimmten Voraussetzungen eine Entscheidungsgrundlage für die Wahl eines der zu implementierenden Partitionierungsalgorithmen darstellen

- Betrachtung anwendungsspezifischer Modelleigenschaften als Ausgangspunkt für die Entwicklung neuer Partitionierungsalgorithmen

(10) Untersuchungen zur Parallelisierung von Partitionierungsalgorithmen
Ausführung: M2

- Komplexitätsuntersuchungen für Partitionierungsalgorithmen

- Betrachtung von Parallelisierungsansätzen, bei denen kooperierende Prozesse keine gemeinsamen Speicherbereiche besitzen (Kommunikation über *message passing*, lose gekoppelte Prozessoren als Zielhardware im Hintergrund)

- Entwicklung von Ansätzen auf der Basis statischer Prozeßkonzepte, Betrachtung unterschiedlicher Möglichkeiten der Modelldatenzuordnung zu Prozessen (entweder Gesamtmodell oder Modellpartitionen als Ergebnisse einer Vorpartitionierung, in letzterem Fall sind insbesondere *top down* -Partitionierungsstrategien von Interesse)

- Untersuchungen hinsichtlich des Einsatzes dynamischer Prozeßkonzepte

- Entwicklung von Kriterien für *lohnende* Parallelisierbarkeit von Partitionierungsalgorithmen, die im Vorfeld compilergesteuerter Logiksimulationen interessant sind

- Ableitung von Anforderungen an die Implementierung des Partitionierers ausgehend von betrachteten Parallelisierungsansätzen

(11) Realisierung der parallelen Version der Testumgebung

- Einbindung des Kommunikationsinterface in zu entwickelnde Shellkomponenten, vorbereitende Arbeiten für den Übergang von der Simulation der parallelen Version auf der Workstation zur Implementation auf *SP1* -Systemen
Ausführung: S1

- Realisierung der für die parallele Version vorgesehenen Manipulations- und Analyseprimitiven
Ausführung: S2

- Realisierung der für die parallele Version vorgesehenen Shellkomponenten
Ausführung: S3

(12) Konzipierung der Partitionierungskomponente
Ausführung: M1, M2

- M1 bringt in das Konzept des Partitionierers einen Auswahlmechanismus für eine Menge von Partitionierungsalgorithmen in Abhängigkeit von vorhergehendem Modellpräprocessing ein; die in Betracht kommenden Algorithmen einschließlich des Präprocessings sind zu realisieren

- M2 bringt eine Menge paralleler zu realisierender Partitionierungsalgorithmen und damit verbundener Anforderungen an die Realisierung des Partitionierers ein

(13) Untersuchungen zur Modelltransformation im Vorfeld der Logiksimulation
Ausführung: M1

- Ausbau der Untersuchungen unter (9), komplexere Modellanalysen auf der Basis der parallelen Testumgebung

- Untersuchung von Transformationsregeln, welche Modelle in solche aus Modellklassen mit bestimmten partitionierungsbezogenen Eigenschaften überführen, Betrachtung der Konsequenzen hinsichtlich erforderlicher Begleitprozesse im Kontext paralleler Simulationen über Partitionen, die ausgehend von transformierten Modellen entstanden sind

(14) Realisierung des Partitionierers

- Erweiterung der Funktionalität der parallelen Testumgebung
Ausführung: M2, S1, S2, S3

- Realisierung von Partitionierungsalgorithmen, wobei M2 parallele Algorithmen behandelt
Ausführung: M2, S2, S3

- Realisierung des auf M1 zurückgehenden Modellpräprocessings und des zugehörigen Auswahlmechanismus für Partitionierungsalgorithmen, Gestaltung der Schnittstelle zum *MaxSim-Simulationssystem* und Einbindung des Partitionierers
Ausführung: S1

- Testläufe (für paralleles Präprocessing und parallele Partitionierungsalgorithmen über Netz auf SP1)
Ausführung: S1, S2, S3

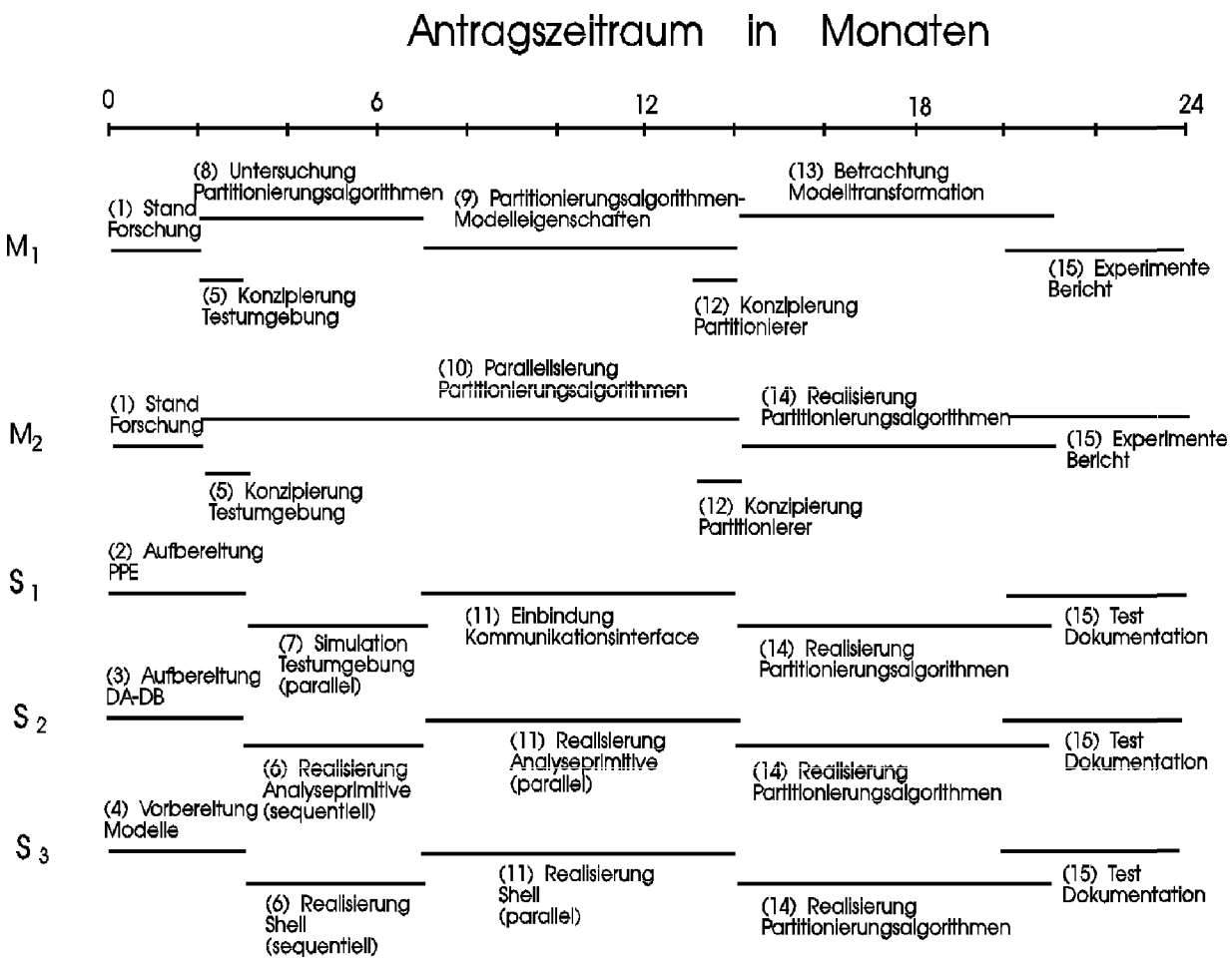
(15) Abschließende Arbeiten

- Tests und Dokumentationen zu den entstandenen Softwarekomponenten
Ausführung: S1, S2, S3

- Durchgehende Experimente vom Modell bis zur parallelen *MaxSim* - Simulation
Ausführung: M1, M2, S1, S2, S3

- Bericht über die Arbeit im Antragszeitraum mit Ausblick auf ihre Weiterführung
Ausführung: M1, M2

Bei der Realisierung der parallelen Testumgebung und der Partitionierungskomponente ist an eine zusätzliche Einbeziehung von Studenten in Form von Studienarbeiten gedacht.


Antragszeitraum in Monaten
0
6
12
18
24
M1
(1) Stand Forschung
(8) Untersuchung Partitionierungsalgorithmen
(9) Partitionierungsalgorithmen-Modelleigenschaften
(13) Betrachtung Modelltransformation
(15) Experimente Bericht
(5) Konzipierung Testumgebung
(12) Konzipierung Partitionierer
M2
(1) Stand Forschung
(10) Parallelisierung Partitionierungsalgorithmen
(14) Realisierung Partitionierungsalgorithmen
(15) Experimente Bericht
(5) Konzipierung Testumgebung
(12) Konzipierung Partitionierer
S1
(2) Aufbereitung PPE
(7) Simulation Testumgebung (parallel)
(11) Einbindung Kommunikationsinterface
(14) Realisierung Partitionierungsalgorithmen
(15) Test Dokumentation
S2
(3) Aufbereitung DA-DB
(6) Realisierung Analyseprimitive (sequentiell)
(11) Realisierung Analyseprimitive (parallel)
(14) Realisierung Partitionierungsalgorithmen
(15) Test Dokumentation
S3
(4) Vorbereitung Modelle
(6) Realisierung Shell (sequentiell)
(11) Realisierung Shell (parallel)
(14) Realisierung Partitionierungsalgorithmen
(15) Test Dokumentation

# 4 Literatur


[Bai92] BAILEY, M.: *How Circuit Size Affects Parallelism.* IEEE Transactions on Computer-Aided Design, vol. 11, pp. 208-215, 1992.

[Bry88] BRYANT, R.E.: *Data Parallel Switch-Level Simulation.* IEEE Int. Conference on Computer-Aided Design, ICCAD-89. Digest of Technical Papers, (IEEE Cat.No.88CH2657-5), pp. 354-357, 1988.

[BSK91] BAUER, H., SPORRER, C., KRODEL, T.H.: *On Distributed Logic Simulation Using Time Warp.* Proc. International Conference on Very Large Scale Integration VLSI 91, pp. 127-136, 1991.

[FiM82] FIDUCCIA, C.M., MATTHEYSES, R.M.: *A Linear-Time Heuristic for Improving Network Partitions.* Proc. 19th Design Automation Conference, ACM/IEEE, pp. 175-181, 1982.

[HoS92] HORROCKS, D.H., SPITTLE, M.C.: *Genetic Algorithms.* IEE Colloquium on Circuit Theory and DSP (Digest No.037), pp. 1-5, 1992.

[KrA88] KRAVITZ, S., ACKLAND, B.: *Static vs. Dynamic Partitioning of Circuits for MOS Timing Simulator on a Message-based Multiprocessor.* Proc. Distributed Simulation Conference, San Diego, pp. 136-140, 1988.

[Mei93] MEISTER, G.: *A Survey on Parallel Logic Simulation.* Universität Saarbrücken, SFB 124, Report 14/1993.

[MSD93] MUELLER-THUNS, R.B., SAAB, D.G., DAMIANO, R.F., ABRAHAM, J.A.: *VLSI Logic and Fault Simulation on General-Purpose Parallel Computers.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, pp. 446-460, 1993.

[Pfi82] PFISTER,G.: *The Yorktown Simulation Engine: Introduction.* Proc. 19th Design Automation Conference, ACM/IEEE, pp. 51-54, 1982.

[Pos89] POST, H.-U.: *Entwurf und Technologie hochintegrierter Schaltungen.* Teubner-Verlag, 1989.

[PoS92] POST, H.-U., SCHWARZ, P.: *Parallele ereignisgesteuerte Logiksimulation.* ASIM-Mitteilungen, Heft 29, 1992.

[SiO93] SIMIC, N., ORTNER, H.: *Partitioning Strategies Within a Distributed Multilevel Logic Simulator, Including Dynamic Repartitioning.* EURO-DAC 1993.

[SoB87] SOULE, L., BLANK, T.: *Statistics for Parallelism and Abstraction in Digital Simulation.* Proc. 24th Design Automation Conference, ACM/IEEE, pp. 588-591, 1987.

[SpB93] SPORRER, C., BAUER, H.: *Corolla Partitioning for Distributed Logic Simulation of VLSI-Circuits.* Proceedings of the 1992 SCS Western Simulation Multiconference on Parallel and Distributed Simulation (PADS92), pp. 205-209, 1993.

[Spr89] SPRUTH, W.G.: *The Design of a Microprocessor.* Springer-Verlag, Berlin, Heidelberg, 1989.