Beschleunigte Verifikation von analogen Schaltungen durch Multi-Thread und Multi-Job-Ansätze mit Simulatoren

Udo Sobe und Achim Graupner
ZMDI, Grenzstrasse 28, 01109 Dresden, {Udo.Sobe, Achim.Graupner}zmdi.com

Kurzfassung

Während des Schaltungsentwurfes werden analoge Schaltungen mit Hilfe von Simulationsverfahren verifiziert. Rechentechnisch ist die Verifikation analoger Schaltungen aufwendig. Der Aufwand äußert sich in der Simulationszeit, der Anzahl der Simulationen oder in der Kombination dieser Bestandteile. Hier bieten sich Parallelisierungstechniken an, um Hardware und Software Ressourcen effizient einzusetzen. Deren Anwendung wird beispielhaft an Hand des ZMDI hausinternen Verifikationswerkzeugs zmdAnalyser und dem kommerziellen Simulators von Cadence dargestellt.

1 Einführung

Damit analoge Schaltungen nicht bei Schwankungen von Fertigung oder Einsatzbedingungen unzureichende Eigenschaften zeigen oder ausfallen, wird die erforderliche Ausbeute bereits während des Schaltungsentwurfes untersucht und verbessert (Design for Yield (DfY)). Die Untersuchung, im weiteren als Verifikation bezeichnet, basiert auf Simulationen von Varianten und baut auf bekannte und qualifizierte Analysen des Simulators (z.B. AC, DC; TRAN, ...) auf.

Die Anzahl der Varianten V und die Simulationszeit t_{sim} bestimmen den Simulationsaufwand der Verifikation. Zur Erstellung der Varianten sind Methoden zur Versuchsplanung (Design of Experiments (DOE)) nutzbar. Die Simulationszeit t_{sim} der Einzelsimulation ist von vielen Faktoren abhängig (z.B. Schaltung, Testschaltung, Abstraktionslevel, Modelle, usw.). Im Beitrag wird davon ausgegangen, dass die Simulationszeit der Einzelsimulation bereits durch bekannte Maßnahmen (z.B. Host, Genauigkeitseinstellungen, Konvergenzverfahren, Startbedingungen, etc.) verbessert wurde.

Der Beitrag vergleicht Methoden, um die Simulationszeit zu verkürzen. Dazu werden zunächst Methoden zur Verifikation von Schaltungen gezeigt. Basierend auf den genutzten kommerziellen Schaltungssimulator Spectre [1] von Cadence und dessen Implementierung im ZMDI hausinternen Werkzeug zmdAnalyser [2] werden Eigenschaften der Simulationsjobs abgeleitet und Verbesserungsmaßnahmen vorgeschlagen. Danach werden heute nutzbare Parallelisierungstechniken vorgestellt. Der letzten Abschnitt beschäftigt sich mit den Anforderungen an Hardware und Software Ressourcen infolge der Parallelisierungstechniken.

2 Verifikation von Schaltungen

Die Verifikation der analogen Schaltung erfolgt über sogenannten PVT (Process, Voltage, Temperature) Bedingungen, d.h. das Schaltungsverhalten wird für unterschiedliche technologische Lagen des Prozesses und bei unterschiedlichen Betriebsbedingungen untersucht (s. Abb. 1). Die Betriebsbedingungen (operating conditions) sind parametrisierbar und schließen Lastbedingungen und Betriebsmodi mit ein.

Zur Verifikation haben sich zwei Analysen bewährt. Mit Hilfe der Corner Analyse werden verschiedene technologische Lagen bei den Grenzen des Prozesses betrachtet. Sie liefert PASS/FAIL Aussagen bei der Verifikation. Statistischen Verfahren, wie die Monte Carlo (MC) Analyse, nutzen globale und lokale Prozessstatistiken zur Bestimmung der technologischen Lagen. Bei dieser Analyse ist eine Ausbeuteabschätzung möglich.

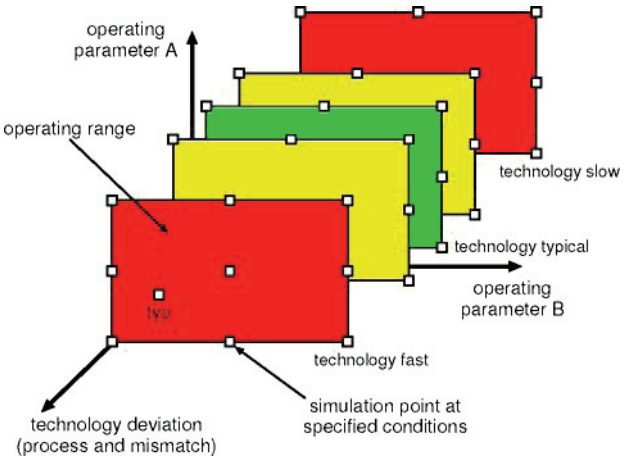

Bild 1. Veranschaulichung der Verifikation über Prozessstreuung und zwei Einsatzbedingungen (A und B).

Bei der parametrischen Corner Analyse (vgl. Abb. 2a) wird für jede Variante, bestehend als Kombination aus I technologischen Lagen und J Einsatzbedingungen, je ein einzelner Simulationsjob generiert. Die technologische Lage i wird auch als Prozess-Corner bezeichnet. Jede Einsatzbedingung j enthält einen Parametersatz. Typisch für diese Analyse sind viele einzelne Simulationsjobs $I \cdot J$, die unabhängig voneinander abgearbeitet werden (s. simulation points in Abb. 1). Die Laufzeit der Jobs entspricht der Simulationszeit $t_{job} = t_{sim}$.

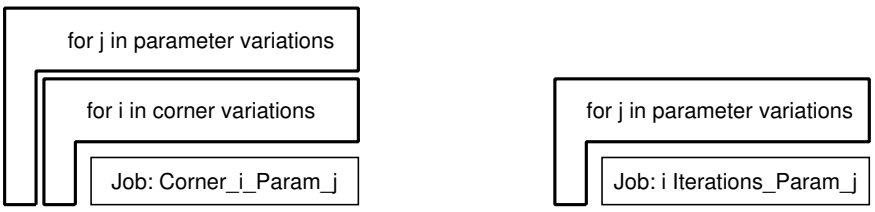

Bild 2. Generierung der Simulationsjobs für Corner Analyse und MC Analyse. a) Bei der Corner Analyse wird je Kombination, bestehend aus Prozess-Corner i und Einsatzbedingung j, ein Simulationsjob gebildet. b) Weil die MC Analyse im Simulator implementiert ist, erfolgt die Iteration des Prozesses i im Simulationsjob selbst. Es ist lediglich pro Einsatzbedingung j ein Simulationsjob notwendig.

Die parametrische MC Analyse wird vom Simulator Spectre nicht für eine beliebige Parameteranzahl unterstützt. Deshalb wurde für die Variation des Prozesses die in Spectre eingebaute Monte Carlo Analyse benutzt und durch eine externe Parametrisierung erweitert (s. Abb. 2b). Die Implementierung der MC Analyse wirkt wie eine Gruppierung einzelner Simulationsjobs über die technologischen Lagen, die auch MC Iterationen I genannt werden. Aus dieser Implementierung resultiert pro Betriebsbedingung ein Simulationsjob. In der Praxis ist die Anzahl der Betriebsbedingungen J (z.B. 10) geringer als die Anzahl der MC Iteration I (z.B. 500), d.h. die Anzahl der Simulationsjobs ist im Vergleich zur Corner Analyse gering und besitzen mit $t_{job} \sim I \cdot t_{sim}$ eine lange Laufzeit.

Durch die vorliegende Implementierung werden in der Praxis die folgenden Fälle durch das Verteilsystem nur unzureichend unterstützt:

- Kurzläufer mit $t_{sim} < 1min$ bei Corner Analyse: Der zeitliche Mehraufwand durch das Verteilsystem addiert sich spürbar zur Simulationszeit. Diese Simulationsjobs der Corner Analyse sollten gruppiert werden.
- Langläufer mit $t_{sim} > 1h$ bei MC Analyse: Der Gesamtjob läuft mehr als einen Tag. Für eine effektivere Abarbeitung im Verteilsystem sollten diese Simulationsjobs der MC Analyse geteilt werden.

3 Nutzung von Parallelisierungstechniken

3.1 Parallelisierung eines einzelnen Jobs durch Multi-Threading

Die Abarbeitungszeit eines einzelnen Simulationsjobs kann durch Parallelisierung innerhalb des Programms beschleunigt werden (single replication in parallel (SRIP)). Dies wird zunehmend für Simulationsaufgaben von Langläufern während des interaktiven Entwurfes genutzt.

Spectre bietet hierzu folgende Möglichkeiten:

- Aktivierung von Multi-Threading mit Hilfe der Option: +mt N (N: Anzahl der Threads, für Spectre 1-4)
  - Vorteil: Lizenzbedarf ohne/mit Multi-Threading ist gleich
  - Nachteil: begrenzter Zeitgewinn, nutzbar ab 256 BSIM Bauelementen
  - Beispiel: bis zu ca. 140% CPU Auslastung (CPU utilization) für ausgewählte Beispiele mit N=2
- Benutzung von Spectre als Accelerated Parallel Simulator (APS) [3] mit Option: +aps
  - Bemerkung: benutzt Multi-Threading und neuen Solver
  - Vorteil: spürbarer Zeitgewinn vor allem bei Langläufern
  - Nachteil: Zeitgewinn hängt stark von Schaltung ab, höherer Lizenzbedarf und für Anwender komplexes Lizenzmodel (s. Cadence licence model), keine Empfehlung/Automatisierung einer effizienten Thread-Anzahl), auf einen Host begrenzt
  - Beispiel: bis zu ca. 450% CPU Auslastung (CPU utilization) für ausgewählte Beispiele mit N=8

Erfahrungen zeigten, dass eine Platzierung mehrerer Threads auf nur einem CPU-Kern zu keinem Geschwindigkeitsgewinn führt. Deshalb werden die vom Schaltkreissimulator Spectre bzw. Spectre APS erzeugten Threads von vornherein auf unterschiedliche CPU-Kerne einer Multi-Core CPU verteilt. Weitere Parallelisierungstechniken, wie z.B. Multi-Hosts, werden zur Zeit von Spectre nicht unterstützt.

3.2 Parallelisierung von vielen Jobs durch Multi-Job

Zahlreiche einzelne Simulationsjobs werden zeitgleich auf verschiedene CPUs bzw. Rechner verteilt und parallel abgearbeitet (multiple replication in parallel (MRIP)). Dieses Verfahren wird auch als einfache Parallelisierung bezeichnet und wird häufig genutzt, um Zeitaufwand der Simulation für Corner oder MC Analysen zu verkleinern. Alle abzuarbeitenden Simulationen werden vorbereitet (Netzliste, Stimuli, etc.) und der Job als Auftrag in das Verteilsystem (im ZMDI: LSF [4]) gestellt. Das Verteilsystem bearbeitet diese Aufträge entsprechend der Konfiguration (Last, Fairshare usw.), s. Abb. 3.

Ein zusätzlicher Zeitbedarf durch das Verteilsystem wird beobachtet, der vor allem bei Kurzläufern spürbar ist. Aus diesem Grund bringen einige Werkzeuge, welche eine hohe Anzahl von Jobs generieren, eigene Verteilsysteme

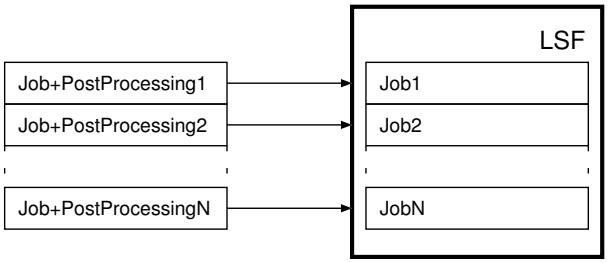

Bild 3. Jobs in das Verteilsystem (LSF-Cluster) schicken.

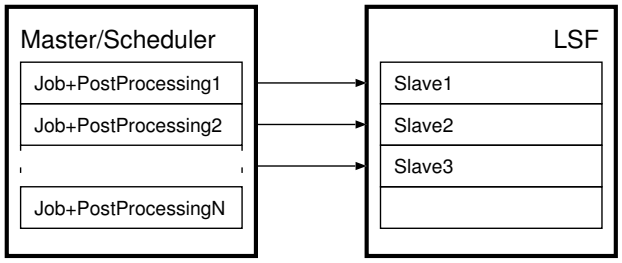

Bild 4. Werkzeuge mit eigenen Verteilsystem hebeln ein bestehendes Verteilsystem aus.

als Master/Slave Konzept mit (s. Abb. 4). Dabei werden zunächst Jobs auf freie Hosts im Verteilsystem gestartet. Diese verteilten Jobs dienen als Slave und führen die Simulationsaufgaben durch, die durch einen zentralen Job (Master) koordiniert werden. Ein bestehendes Verteilsystem wird dabei für die Simulationsjobs umgangen und dient nur zum Starten der Slaves.

Beispiele sind:

- WiCkeD/MunEDA: Distributor/Simserver
- zmdAnalyser/ZMDI: Master/Slave

3.3 Kombination aus Multi-Threading und Multi-Job

Die Kombination aus Multi-Threading und Multi-Job wird zunehmend bei der Analyse von Varianten mit langer Simulationszeit des einzelnen Jobs genutzt. Als Varianten werden Arbeitsbedingungen oder Betriebsmodi untersucht. Beispiele hierzu sind:

- die parametrische Simulation vom Cadence ADE (Analog Design Environment) mit Verteilung der Jobs im LSF und Benutzung des APS Simulators
- die zmdAnalyser Corner Analyse mit Job-Verteilung und Benutzung des APS Simulators

3.4 Anforderung an die Infrastruktur

Klassische Verteilsysteme, wie z.B. LSF, sind auf die CPU-Last fokussiert und können die Hardware optimal ausnutzen. Diese Jobs laufen im Batch - Betrieb, z.B. in der Nacht. Darüber hinaus muss in der Praxis auf eine ausgewogene Lizenzauslastung für Jobs außerhalb des Batch - Betriebes geachtet werden. Um interaktive Tätigkeiten im ZMDI nicht zu behindern, werden den Jobs im Batch - Betrieb nicht die gesamten Lizenzen zur Verfügung gestellt, sondern Lizenzen für interaktive Aufgaben vorgehalten.

Bei Nutzung der Multi-Threading Technik muss dem Verteilsystem für jeden Simulationsjob die Anzahl der Threads bzw. CPUs und Lizenzen mitgeteilt werden. Dies ist notwendig, um andere Jobs nicht zu blockieren, die Lizenzauslastung berechnen zu können und einen Vorhalt an Lizenzen für die interaktive Arbeit zu garantieren. Das Lizenzmodel der Simulatoren ist komplex und hängt bei Spectre u.a. vom Solver (Spectre, APS, UltraSim), Anzahl der Threads oder Version ab.

4 Ressourcen-Bedarf

Den Zeitgewinn bei Anwendung von Multi-Threading und Multi-Job erkauft man sich im Allgemeinen durch einen erhöhten Bedarf an Hard- und Software Ressourcen. Dabei ist unter Software Ressourcen der Lizenzbedarf zu verstehen. I.d.R. stehen bei industrieller Anwendung sowohl Hard- als auch Software Ressourcen nicht unbegrenzt zur Verfügung.

Die Software Ressourcen können als Kompromiss aus Kosten und Nutzen geplant und bereitgestellt werden. Bei Engpässen von Software Ressourcen ist eine kurzfristig Erweiterung möglich. In der Regel werden Hardware Ressourcen langfristig geplant und realisiert. Zeitnah und temporär können Engpässe von Hardware Ressourcen innerhalb einer Firma nicht ausreichend angepasst werden. Deshalb ist ein Einsatz von Hardware Ressourcen außerhalb der Firma, z.B.durch GRID, interessant.

Bei der industriellen Anwendung im ZMDI wird ausschließlich kommerzielle Software im Schaltungsentwurf eingesetzt. Aktuelle Lizenzverträge schließen eine Nutzung der Lizenzen außerhalb der Infrastruktur der Firma aus. D.h. unabhängig von Sicherheitsaspekten (ungewollte Nutzung der Lizenzen durch Dritte) ist heute die Verwendung kommerzieller Software im öffentlichen GRID durch ZMDI nicht möglich. Folglich ist die Hardware an die bestehende Infrastruktur der Firma gebunden und kann bei Bedarf nicht durch einen öffentlichen GRID erweitert werden.

5 Beispiel: zmdAnalyser

Obige Maßnahmen von Multi-Job und Multi-Thread werden für sich oder in Kombination mit dem Werkzeug zmdAnalyser unterstützt. Die in Abschnitt 2 vorgeschlagenen Maßnahmen zur besseren Abarbeitung im Verteilsystem müssen bereits bei der Vorbereitung der Simulationsjobs berücksichtigt werden. Davon betroffen sind sowohl Optionen beim Simulatoraufruf als auch die Netzliste und deren weitere Bestandteile.

Die Implementierung von Multi-Threading erfolgte, wie in Abschnitt 3.1 beschrieben, mit Hilfe der verfügbaren Argumente, die beim Jobaufruf dem Simulator mit übergeben werden. Der Zeitgewinn, bzw. die Simulationszeit t_{sim}, skaliert nicht linear mit der Anzahl der

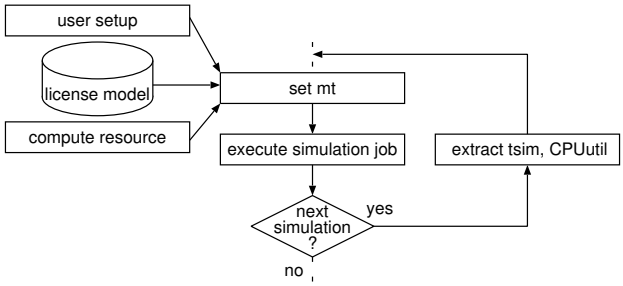

Bild 5. Vorgehen zur Adaption von Multi-Thread Optionen zur Verbesserung von CPU-Auslastung CPU_{util}, Simulationszeit t_{sim} und Anzahl der Threads mt.

Threads mt (u.a. Amdahl's Gesetz in [5]) und hängt u.a. vom Simulationsproblem ab. Da der Simulator Spectre keine Empfehlungen zur optimalen Anzahl der Threads gibt, kann dies z.B. mit Hilfe der Simulationszeit t_{sim} and CPU-Auslastung CPU_{util} bestimmt werden (Teil der Log Datei des Simulators). Abb. 5 zeigt ein mögliches Vorgehen zur Adaption der Thread-Anzahl mt in Abhängigkeit zur ermittelten Simulationszeit t_{sim} und CPU-Auslastung CPU_{util}. Dieses Verfahren bietet sich u.a. bei der Kombination von Multi-Job und Multi-Thread an

Der bereits getestete Ansatz für die Corner Simulation ermittelt zunächst die Simulationszeit von Einzelsimulationen und legt anschließend fest, ob alle weiteren Simulationsjobs zu gruppieren sind (vgl. Abb. 6). Der zusätzliche Zeitbedarf durch das Verteilsystem wird dadurch für Kurzläufer minimiert. Näherungsweise kann der Zeitbedarf für die gesamte Analyse $t_{analyse}$ mit folgendem linearen Zusammenhang bestimmt werden:

$$t_{analyse} = t_{prepare} + \frac{IJ}{n_{job}} \overline{t_{job}} \quad mit \quad IJ >> n_{jobs} \qquad (1)$$

$$t_{prepare} = t_{netlist} + t_{scripting} + t_{start} \qquad (2)$$

$$\overline{t_{job}} = t_{dist} + \overline{t_{sim}} + \overline{t_{measure}} \qquad (3)$$

Die Vorbereitungszeit $t_{prepare}$ bildet sich aus den Zeitanteilen zum Netzlisten $t_{netlist}$ (schaltungsabhängig), zur Skripterzeugung $t_{scripting}$ (ca. < 10s) und zum Start der Jobs t_{start} (z.B. LSF: ca. 1/2 Minute). Jeder Job mit der mittleren Zeit $\overline{t_{job}}$ benötigt Zeit zum Verteilen der Netzliste t_{dist} (<1s), eine mittlere Simulationszeit $\overline{t_{sim}}$ (schaltungsabhängig) sowie eine mittlere Zeit zur Extraktion von Schaltungseigenschaften $\overline{t_{measure}}$ (schaltungsabhängig). Der Zeitaufwand für die Job-Anzahl IJ mit der mittleren Zeit je Job $\overline{t_{job}}$ kann durch die Anzahl der Jobs n_{job} durch Multi-Job verbessert werden.

Eine einmalige Zeitmessung zu Beginn der Jobverteilung hat den Nachteil, dass diese nicht auf Schwankungen der Simulationszeit reagieren kann. Gründe hierfür können z.B. sein:

- Schaltung nicht stabil und schwingt über technologische Lagen und/oder Einsatzbedingungen

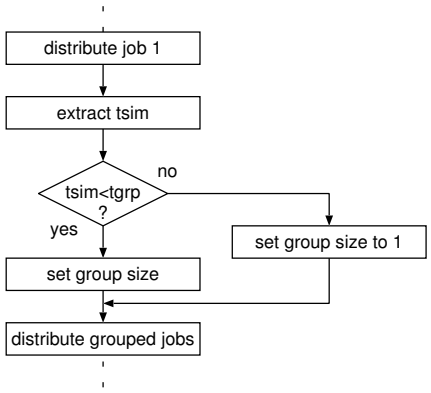

Bild 6. Simulationszeitabhängige Gruppierung der Simulationsjobs zur Jobverteilung. Ist die Simulationszeit t_{sim} kleiner als die vorgegebene Zeitvorgabe zur Gruppierung t_{grp} erfolgt eine Gruppierung der Simulationsjobs.

- Simulation unterschiedlicher Betriebsmodi, gesteuert durch Parametersatz

Eine wiederholte Zeitmessung kann dies verbessern.

6 Zusammenfassung

Die Verifikation von analogen Schaltungen basiert auf der Variantensimulation mit Hilfe der Corner Analyse oder Monte Carlo Analyse. Für die Beschleunigung der Verifikation können die Parallelisierungstechniken Multi-Thread und Multi-Job für sich getrennt oder in Kombination genutzt werden. Bei beiden steigt der Bedarf an Hard- und Software (Lizenz) Ressourcen, die durch ein Verteilsystem zu berücksichtigen sind. Durch das Gruppieren von vielen kurzen Simulationsjobs bzw. das Aufteilen von langen Simulationsjobs für den Multi-Job Ansatz kann die Verifikation mit dem zmdAnalyser beschleunigt werden.

Danksagung

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung, und Forschung unter dem Förderkennzeichen 01IG09011F gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Literatur

[1] Cadence. *Virtuoso Spectre Circuit Simulator User Guide*, 2008.

[2] Udo Sobe and Uwe Henniger. zmdAnalyser: Ein Design Tool von Analog/Mixed Signal Designern. *DASS*, pages 29–33, 2005.

[3] Cadence. *Virtuoso Accelerated Parallel Simulator User Guide*, 2008.

[4] Platform. LSF configuration reference, 2009.

[5] Gene Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *AFIPS*, pages 483–485, 1967.