Platzierung & Verdrahtung von Standardzellen

Werkzeuge : CADENCE IC, CADENCE Cell Ensemble
Design-Kits : AMS Hit-Kit

A. Mäder

Achtung: diese Anleitung gilt nur für *2-Lagen Verdrahtungen!* Als Layoutwerkzeug wird CADENCE Cell Ensemble eingesetzt, hier beispielhaft mit dem AMS-Prozess `cub`. Für das Layout drei- und mehrlagiger Prozesse muss „Silicon Ensemble“ benutzt werden!

In dieser Beschreibung werden die grundlegenden Schritte beschrieben, wie eine Verilog-Datei als Ausgabe des SYNOPSYS Design Analyzer in eine Datenbasis für CADENCE umgewandelt und anschließend das physikalische Layout durchgeführt wird.

Dabei sind jeweils nur die einfachsten Schritte gezeigt — bei Problemfällen, die bei größeren Entwürfen (wahrscheinlich) auftreten werden, sei auf die CADENCE Manuals verwiesen. . .
In folgenden Kurzanleitungen finden sich zusätzliche Informationen:

„CADENCE Grundlagen“: allgemeine Konzepte, Benutzung des Layout-Editors

„VHDL-Synthese“: Verilog-Datei generieren

Bei der Beschreibung der Benutzereingaben gelten die schon vorher verwendeten Symbole.

Vorüberlegungen

Während der Entwurf bisher von der Funktion der Schaltung geprägt war (VHDL Codierung), treten jetzt neue Aspekte in den Vordergrund, die einige Vorüberlegungen vor den eigentlichen Layout notwendig machen.

PR-Strategie
Im Ablauf des Platzierungs- und Verdrahtungsprozesses gibt es mehrere Möglichkeiten wie die (normalerweise immer vorhandene) Hierarchie der Schaltung behandelt werden soll.

hierarchisches Layout: Dabei werden Teile der Hierarchie (untergeordnete `Cells`) komplett platziert und verdrahtet. Sie haben natürlich keine Padzellen, sondern nur Anschlusspunkte. In der nächsthöheren Hierarchieebene können sie dann wie Makrozellen behandelt werden. Sowohl top-down als auch bottom-up Vorgehensweisen sind möglich:

- Flächenabschätzung und Anordnung von Blöcken, deren Layout und anschließende Verdrahtung der obersten Ebene.
- Platzierung und Verdrahtung von Teilen der Hierarchie und deren spätere Benutzung als Makrozelle.

Ein hierarchisches Layout ist in folgenden Fällen zu empfehlen:

- bei hochregulären Entwürfen, mit vielen „gleichen“ Instanzen — diese müssen allerdings schon bei der Synthese entsprechend behandelt worden sein, siehe „VHDL-Synthese“ Abschnitt: „Behandlung der Hierarchie“.
- bei sehr großen Entwürfen, da die Laufzeiten und Speicherbedarfe der einzelnen Programme (teilweise) exponentiell ansteigen.

flaches Layout: Die Hierarchie wird komplett aufgelöst und als eine große Netzliste bearbeitet. Gegebenenfalls kann hier später noch eine Hierarchie (künstlich) erzeugt werden. Ein Vorteil dieser Methode ist das „bessere“ Layoutergebnis, da die flache Netzliste mehr Optimierungsmöglichkeiten bietet, was allerdings mit längeren Programmlaufzeiten verbunden ist.

Art des Designs

Bei Standardzellentwürfen wird oft zwischen Pad- und Core-limitierten Designs unterschieden, je nachdem ob die Fläche des späteren ICs durch die Anzahl der Pads (viele I/O Pads, wenig Logik) oder durch die Standardzellen (sehr viele Standardzellen, Makrozellen und wenig Pads) bestimmt wird. Da sich die Kosten für die Fertigung nach der Fläche berechnen, trifft im Idealfall keines dieser beiden Extreme zu, sondern die Standardzellfläche im Core-Bereich ist so groß, dass alle Padzellen gerade um diese Fläche angeordnet werden können.

Die Chiphersteller – wie hier AMS – bieten deshalb oft zwei Arten von Pad-Bibliotheken an, die für Pad-bestimmte Entwürfe eher hoch und schmal, für Core-bestimmte eher flach und lang sind.

Pinout

Mit dem Stichwort Pinout ist hier sowohl die Wahl des Gehäuses, als auch die eigentliche Pinbelegung gemeint. Dieser Punkt ist bei unseren Entwürfen zwar kaum von Bedeutung, betrachtet man aber die industrielle Praxis, dann wird das Pinout eines ICs schon während des VHDL Entwurfs festgelegt. Man versucht, in einer möglichst frühen Phase des Designs, zu entscheiden welches Gehäuse einzusetzen ist und wie die Pinbelegung aussieht, um parallel zu dem eigentlichen IC Entwurf ein Platinenlayout, beziehungsweise die Umgebung der Schaltung zu erstellen.

Die Entscheidungen werden dabei von wirtschaftlichen Aspekten (Kosten), von vorgegebenen Randbedingungen (z.B.: Pinkompatibilität zu ...) und natürlich von der Größe des Entwurfs beeinflusst. Da die Gehäuse nur bestimmte Minimal- und Maximalchipgrößen zulassen, ist, neben der Anzahl der Pins, eine frühzeitige und genaue Flächenabschätzung wichtig (z.B.: durch den SYNOPSYS Design Analyzer).

Spannungsversorgung des ICs

Während der Synthese konnten zwar schon für die Ein- und Ausgänge der Schaltung (Ports der `entity`) Padzellen eingefügt werden, es fehlen aber noch die Pads für die Spannungsversorgung.

Dabei können der Padkranz und der Core-Bereich (Standardzellen) über gemeinsame Power-Pads oder aber getrennt versorgt werden. Dies ist insbesondere bei größeren Entwürfen vorteilhaft, da sich Störungen durch die bessere elektrische Trennung nicht so schnell auf die Schaltung auswirken. Normalerweise werden *mehrere* Vdd-/Gnd-Pad Paare benötigt, damit eine stabile Spannungsversorgung der Peripherie-Pads (wichtig für Ausgänge) gewährleistet ist.

Um die Anzahl der benötigten Spannungsversorgungspads zu ermitteln, gibt es entsprechende Richtlinien der Chiphersteller, in denen (hauptsächlich) zwei Einflussgrößen entscheidend sind:

Taktfrequenz beeinflusst vor allem im Core den Stromverbrauch des ICs — CMOS-Schaltungstechnik: Ströme fliessen nur beim Umschalten (Ausgangswert wechselt).

Padzellen in Art und Anzahl. Gerade wenn viele Ausgänge in der Schaltung vorhanden sind, müssen ausreichend Versorgungspads bereitgestellt und gleichmäßig auf dem Padkranz (ringförmige Versorgung s.u.) verteilt werden, um die benötigten Ströme zu liefern.

Power-Routing

Bei der Realisierung der Spannungsversorgung werden im Core-Bereich und in den Padzellen unterschiedliche Konzepte verfolgt. Während der Core durch Netze für (`vdd!` und `gnd!`) versorgt wird, übernehmen Ringe die Spannungsversorgung der Ein- und Ausgangspads.

Padkranz Die Verbindung wird über die ringförmig umlaufenden Leitungen hergestellt — meist in Form mehrerer (unterschiedlicher) Vdd und Gnd Ringe. Da jedes Versorgungspad nur einen bestimmten Strom liefern kann, lässt sich entlang des Padkranzes direkt, durch Aufsummieren der benötigten Ströme, bestimmen, wo Vdd- und Gnd-Pads einzufügen sind. Teilweise können, bzw. müssen (bei analogen Padzellen), diese Ringe durch Einfügen zusätzlicher „Trennzellen“ in unterschiedliche (elektrische) Segmente, mit jeweils eigenen Versorgung, unterteilt werden.

Bei den meisten Technologien werden in den Ecken des Chips spezielle Zellen (*Corner-Cells*) eingefügt, die die Spannungsversorgungsleitungen zu einem Ring schliessen — wird durch die Verdrahtungsalgorithmen bedingt.

Core Hier sind vor allem die zeilenweise angeordneten Standardzellen interessant, da bei Makrozellen (RAM, ROM etc.) die Anschlüsse vorgegeben sind und in den entsprechenden Kanälen direkt angeschlossen werden. Für die Versorgung der Standardzellzeilen werden noch zwei Arten zusätzlicher Zellen eingefügt:

Cap-Zellen sind Anschlusszellen die links/rechts an die Standardzellzeilen angefügt werden und dort den Anschluss von Vdd und Gnd ermöglichen.

Power-Feeds sind zusätzliche Anschlusspunkte in den Standardzellzeilen. Dabei werden die Zeilen so ausgerichtet, dass diese Zellen in allen Zeilen vertikal übereinander liegen und später durch vertikale Leitungen direkt miteinander verbunden werden.

Mit Hilfe dieser Zellen sind unterschiedliche Anschlusskonzepte möglich, wobei die Auswahl von der Länge der Standardzellzeilen und der Taktfrequenz abhängt.

1. sehr kurze Standardzellzeilen werden an der einen Seite an Vdd an der Anderen an Gnd angeschlossen (Cap-Zellen).
2. Zeilen bis 2 mm Länge sollten an beiden Seiten an beide Versorgungsspannungen angeschlossen werden (Cap-Zellen).
3. noch längere Standardzellzeilen enthalten dann einen oder mehrere Power-Feeds — herstellerabhängig auch noch Cap-Zellen, z.B. bei AMS. Als Faustregel ist nach jeweils 1 mm eine Spannungsversorgung vorzusehen.

Durch die unterschiedlichen Möglichkeiten der Einspeisung wird gewährleistet, dass der Spannungsabfall entlang einer Standardzellzeile nicht zu groß wird.

Verdrahtung spezieller Netze
Neben der oben besprochenen Spannungsversorgung gibt es noch einige andere Netze, die unter Umständen bei der Verdrahtung besonders zu behandeln sind. Im allgemeinen Fall werden die Netze bei der Verdrahtung höher priorisiert oder es wird eine Metallisierungsebene als Vorzugslayer festgelegt.

Darüberhinaus werden (technologie- und herstellerabhängig) speziell für Clocknetze besondere Routingalgorithmen verwendet. Während sonst die Minimierung der Gesamtleitungslänge im Vordergrund steht, ist hier eine Reduzierung des Clock-Skew (Zeitversatz durch Leitungslaufzeiten) wichtig.

Design-Flow

Für das Layout eines fertig synthetisierten Standardzellentwurfs (Platzierung und Verdrahtung) sind folgende Einzelschritte durchzuführen:

1. Einlesen der Verilog-Netzliste
2. Nachbearbeitung der Schaltung, Einfügen von Spannungsversorgungspads
3. Datentransfer und Initialisierung von P & R
4. Platzierung der Pad-Zellen
• Makrozellen anordnen und Platzierungsregionen festlegen
5. Platzierung der Standardzellen
6. Verdrahtungskanäle generieren
• Versorgungsspannung und spezielle Netze (global) verdrahten
7. Globale Verdrahtung
8. Kanalverdrahtung
• Extraktion zusätzlicher Kapazitäten für eine Simulation mit Backannotation

Die mit • gekennzeichneten Schritte sind optional und können unter Umständen weggelassen werden.

Arbeitsschritte

Hier sind bei der Beschreibung von CADENCE DF II nur die Schritte zur Durchführung einer (automatischen) Platzierung und Verdrahtung bei Benutzung der AMS-Zellbibliotheken beschrieben. Es werden auch nicht alle, der unter „Vorüberlegungen“ angedeuteten, Vorgehensweisen erläutert. Für kleinere und nicht extrem schnell getaktete Entwürfe sind folgende Einschränkungen möglich:

PR-Strategie: es wird nur das Layout einer flachen Netzliste vorgestellt — für ein hierarchisches Layout sei auf die CADENCE Online-Dokumentation verwiesen.

Verdrahtung spezieller Netze: ist nicht weiter beschrieben, dies betrifft insbesondere Clock-Routing/-Tree.

Verilog-Datei einlesen

1. Start des Systems

> `ams_cds -tech cub -tool dfii -mode fb` [xterm]

Über die Parameter können unterschiedliche Betriebsarten und die Technologie eingestellt werden. Die hier beschriebene Einstellung gilt für einen Standardzell Entwurf mit den 0.6 μm AMS-Bibliotheken `cub`.

Die Angabe der Technologie und des Werkzeugs wird nur für den ersten Aufruf benötigt, später kann der Entwurf direkt mit `ams_cds -mode fb` begonnen werden — die Eingabe des Betriebsmodus ist zwingend erforderlich, sonst können Platzierung und Verdrahtung nicht aufgerufen werden.

2. Bibliothek für den Entwurf anlegen

☐ `File - New - Library...` [icfb - Log:...]

≡ `Name` = ⟨*libId*⟩ [New Library]
`Technology File` = `Attach to an existing techfile`
`Design Manager` = `No DM`

≡ `Attach To...` = `TECH_`⟨*AMStech*⟩ [Attach Design...]

Bei dem hier beschriebenen Prozess (CMOS 0.6 μm) ist die Eingabe: `TECH_CUQ`.

3. Einlesen der Verilog-Datei

Die Erzeugung der Verilog-Netzliste wurde in „VHDL-Synthese“ beschrieben.

☐ `File - Import - Verilog...` [icfb - Log:...]

≡ `Target Library Name` = ⟨*libId*⟩ [Verilog In]
`Reference Libraries` = `HRDLIB basic`
`Verilog Files To Import` = ⟨*vlogFile*⟩`.v`
`-f Options` =
`/local/tech1.2/ams/ams_v3.12/artist/AMS_TH06_1/veritools/VERILOGIN.inc`

Alle anderen Werte sind entsprechend der Voreinstellung beizubehalten!
Anschließend muss man etwas warten während der Datentransfer stattfindet — siehe Ausgabe in [icfb - Log:...]. Er ist erst beendet, wenn das [VerilogIn] Fenster erscheint.

≡ – bestätigen [VerilogIn]

In dem anschließend erscheinenden Fenster wird die bei der Übersetzung entstandene log-Datei angezeigt. Sie ist unbedingt auf Fehlermeldungen und Warnungen hin zu kontrollieren:

`module` ⟨*libCellId*⟩ `already in target/reference library HRDLIB+`
Diese Meldungen sind normal und können ignoriert werden.

`VerilogIn: *W,46: Connectivity extraction failed for schematic view of` ⟨*cellId*⟩.
Es traten Fehler in der Netzliste auf, das Schematic ⟨*cellId*⟩ muss noch nachbearbeitet werden; der eigentliche Datentransfer ist aber beendet.[1]

`VerilogIn: *W,101: Could not find symbol master for instance` ⟨*instId*⟩.
Bei sehr großen Verilog-Dateien haben wir diesen Fehler beim Datentransfer beobachtet (wahrscheinlich ein Programmfehler). In diesen Fall muss die Hierarchie in „kleineren Teilen“ importiert werden. *Es kann nicht weiter gearbeitet werden!*[1]

☐ `File - Close Window` [Log File]

[1]Sprecht mich bei derartigen Fehlern an, es gibt einige Lösungswege zu diesen Problemen -AJM-

Schematic nachbearbeiten

4. Top-level Schematic öffnen

☐ `File - Open...` [icfb - Log:...]

≡ `Library Name` = ⟨*libId*⟩ [Open File]
`Cell Name` = ⟨*topCellId*⟩
`View Name` = `schematic`

5. Test der Daten

optional: Es empfiehlt sich den Entwurf einem Schematic-Rule-Check zu unterziehen. Dieser kann, ausgehend von dem top-level Schematic, hierarchisch über das gesamte Design laufen.

☐ `Check - Rules Setup ...` [Composer-Schematic...]

≡ `Floating Output Pins` = `ignored` [Setup Schematic Rules Check]
`Unconnected Wires` = `warning`
`Solder On CrossOver` = `ignored`

Anpassung an das synthetische Schematic, alle anderen Werte sind „richtig“ voreingestellt.

☐ `Check - Hierarchy ...` [Composer-Schematic...]

≡ `Process` = `every schematic` [Check Hierarchy]

Während des SRC werden die Programmeldungen in dem CADENCE Eingabefenster [icfb - Log:...] ausgegeben.

Traten Fehler auf, so sind diese vor der weiteren Bearbeitung des Entwurfs zu beheben. Um die Hierarchie zu traversieren, werden folgende Befehle benutzt:

$\uparrow_l$ ⟨*instance*⟩ [Composer-Schematic...]
Referenziertes Element selektieren, das selektierte Element wird umrahmt dargestellt.

☐ `Design - Hierarchy - Descend Edit.../`⊙ `E` [Composer-Schematic...]

≡ `View Name` = `schematic` [Descend]

Die `schematic`-View der selektierten Zelle wird zum Editieren geöffnet. Anschließend können die „Fehler“ in dem Entwurf korrigiert werden.

☐ `Design - Check and Save/`⊙ `X` [Composer-Schematic...]
Das Schematic wird erneut geprüft und anschließend gesichert.

☐ `Design - Hierarchy - Return/`⊙ `^e` [Composer-Schematic...]
Rücksprung innerhalb der Hierarchie.

6. Einfügen noch fehlender Padzellen

Sofern die Padzellen nicht schon automatisch durch den SYNOPSYS Design Analyzer eingefügt worden sind, müssen sie jetzt referenziert und angeschlossen werden. Dabei müssen die Ports der Schaltung, die jetzt schon vorhanden sind, über die Padzellen auf die internen Anschlüsse geführt werden.

Auf alle Fälle sind die noch fehlenden Pads für die Versorgungsspannung in dem Schematic zu ergänzen. Entsprechend den auf Seite 2 vorgestellten Kriterien, ist die Anzahl und Art benötigter Power-Pads zu bestimmen.

☐ `Add - Component.../`⊙ `i` [Composer-Schematic...]

≡ `Library Name` = `HRDLIB` [Add Component]
`Cell Name` = ⟨*cellId*⟩
`View Name` = `symbol`

Erzeugen von Instanzen und Platzierung der Zellen im Schematic. Über das Instantiierungsmenü können die benötigten Instanzen erzeugt werden, anschließend wird die Eingabe mit `Cancel` beendet.

Die Zellen für die Spannungsversorgung finden sich im `Library Browser` unter:

				Spannung für		Art des Designs	
`Library`	`Category`	`Cell`	Funktion	Pads	Core	padlim.	corelim.
`HRDLIB`	`POWER_PADS`	`PP01`	VSS	*	*	*	*
		`PP02`	VDD	*	*	*	*
		`PP03`	VSS		*	*	
		`PP04`	VDD		*	*	
		`PP05`	VSS	*		*	
		`PP06`	VDD	*		*	

☐ `Design - Check and Save`/⊙ `X` [Composer-Schematic...]
Das Schematic kann nun geprüft und anschließend gesichert werden.

7. Erzeugen einer Netzliste für Platzierung & Verdrahtung
Entsprechend der Vorgehensweise beim Layout (siehe Seite 1), wird in diesem Schritt eine Netzliste der Schaltung, bzw. von Teilen der Schaltung, erzeugt. Hier wird ein *flaches Layout* erstellt, das die Hierarchie der Schaltung nicht (direkt) in Layout berücksichtigt.

☐ `File - Export - PRFlatten...` [icfb - Log:...]

≡ `Library Name` = ⟨*libId*⟩ [Preview Flatten]
`Cell Name` = ⟨*topCellId*⟩
`View Name` = `schematic`
`Switch List Views` = `... netlist pr_sch|prc_sch`
`Stop List Views` = `autoAbstract abstract`
`Run` = `Generate Physical Hierarchy`

Hier ist die Art der Pad-Bibliothek einzustellen — siehe „Art des Designs“, Seite 2. In dem Feld `Switch List Views` muss dazu ein zusätzlischer Eintrag ergänzt werden:

`pr_sch` für Pad-limitierte Entwürfe
`prc_sch` für Core-limitierte Entwürfe

Layout initialisieren

8. Start von Platzierung und Verdrahtung

☐ `Design - Open...` [Composer-Schematic...]

≡ `Library Name` = ⟨*libId*⟩ [Open File]
`Cell Name` = ⟨*topCellId*⟩
`View Name` = `autoLayout`

☐ `Tools - Floorplan/P&R - Cell Ensemble` [Virtuoso...]
Laden der flachen Netzliste und Initialisierung der P & R-Werkzeuge. Dabei erscheint ein neues Fenster [OSW] in dem die Selektierbarkeit von Layoutobjekten umgeschaltet werden kann (selektierbar: dick umrandet).

9. Initialisierung und Flächenabschätzung

☐ `Floorplan - Reinitialize...` [Virtuoso...]

≡ – bestätigen [Initialize Floorplan]

Die Platzierung wird vorbereitet, dazu werden die benötigten Flächen für Core- und Pad-Bereiche abgeschätzt. Die Ergebnisse werden in [icfb - Log:...] ausgegeben. Für die Platzierung der Standardzellen wird eine Platzierungsregion, entsprechend der Flächenabschätzung, erzeugt.

Padzellen platzieren

10. Platzierung der Padzellen

Die Pad-Platzierung kann entweder automatisch oder über eine Steuerdatei, zur Festlegung einer vorbestimmten Reihenfolge, erfolgen — siehe „Pinout" und „Power-Routing" (Seite 2 und 3). Hier werden beide Möglichkeiten kurz vorgestellt.

Anordnung der Padzellen über eine Steuerdatei

\> `vi` ⟨*ioPlcFile*⟩ [xterm]

Die Datei enthält zeilenweise Einträge der folgenden Form:

⟨*instId*⟩ ⟨*side*⟩ ⟨*offset*⟩

Beispieldatei

```
|I3|PeriCell       left     0
|U192|PeriCell     left     1
|U193|PeriCell     left     2
|I2|PeriCell       bottom   0
|U189|PeriCell     bottom   1
|U190|PeriCell     bottom   2
|I1|PeriCell       right    0
|U191|PeriCell     right    1
|U186|PeriCell     right    2
|I4|PeriCell       top      0
|U187|PeriCell     top      1
|U188|PeriCell     top      2
```

⟨*instId*⟩ (Instanz der Layout-Netzliste) entspricht dabei dem Instanzennamen im Schematic, erweitert um `|PeriCell`. Er kann mit dem Schematic-Editor angesehen werden.

⟨*side*⟩ / ⟨*offset*⟩ Bei der Beschreibung der Anordnung ist die Zählrichtung der Seiten gegen den Uhrzeigersinn festgelegt.

☐ `Place - IO Commands - Read Initial File...` [Virtuoso...]

≡ `IO Frame File` = ⟨*ioPlcFile*⟩ [Build IO Frame]

`Quit if Incomplete` = on

Während die Padzellen ausgerichtet werden, erhält man im CADENCE Eigabefenster einige Meldungen „`WARNING Contact...`", die ignoriert werden können.

Automatische Platzierung der Padzellen

☐ `Floorplan - I/O Place...` [Virtuoso...]

≡ – bestätigen [IO Placer]

Im Allgemeinen werden die Padzellen bei der vollautomatischen Platzierung unsinnig angeordnet. Dementsprechend ist eine Nachbearbeitung notwendig, um beispielsweise Vdd- und Gnd-Pads gleichmäßig zu verteilen und Busse zu ordnen — siehe unten.

Kontrolle der Platzierung / Umordnung „von Hand"

Hierbei können Padzellen, bzw. Gruppen von Pads (nach vorheriger Selektion), gespiegelt, verdreht und verschoben werden, um die gewünschte Platzierung zu erreichen. Da die „genaue" Ausrichtung erst später vorgenommen wird, genügt hier eine ungefähre Positionierung.

Die Namen der Padzellen lassen sich *nur* über deren Properties feststellen (Selektion und Aufruf der Property Fill-Form).

≡ `Instance` = `on` [OSW]
⟨*`others`*⟩ = `off`

$\uparrow_l$ ⟨*`padCell`*⟩ [Virtuoso...]

Selektion einer (oder mehrerer) Padzelle(n).

☐ `Edit - Properties...`/⊙`q` [Virtuoso...]

Kontrolle des Instanzennamens. Durch anschließende Selektion anderer Padzellen können deren Eigenschaften angesehen werden.

☐ `Edit - Move`/⊙`m` [Virtuoso...]

≡ – entsprechend ausfüllen [Move]

Auswahl der Spiegelungen und Rotationen; es sind mehrere Angaben möglich, Rotationen sind 90° pro Auswahl...

11. Einfügen von Ecken (Corner-Cell)

☐ `Place - IO Commands - Add Corners...` [Virtuoso...]

≡ `Glue Cell Library Name` = `HRDLIB` [Insert IO Corner Cells]
`Glue Cell Master Name` = `CORNERP` | `CORNERC`
`Glue Cell Master View` = `abstract`
`Net Association` = `matchTermName`

Die Corner-Zellen werden automatisch eingefügt. Auch hier wird zwischen den beiden Arten der Padzellen unterschieden:

`CORNERP` für Pad-limitierte Entwürfe
`CORNERC` für Core-limitierte Entwürfe

12. Sichern? — es empfiehlt sich, den Entwurf während des Platzierungs- und Verdrahtungsprozesses des öfteren zu sichern, um beim Auftreten von Fehlern, die sich erst später bemerkbar machen, dort wieder aufsetzen zu können.

☐ `Design - Save As` [Virtuoso...]

≡ `View Name` = ⟨*`padPlaced`*⟩ [Save As]

Hier ist ein (beliebiger) „sprechender" Name der Sichtweise einzutragen. Um die Arbeit später an dieser Stelle fortzusetzen, kann diese View dann direkt geladen werden:

□ `File - Open...` [icfb - Log:...]

Bei Beenden der Programme ist darauf zu achten, dass `autoLayout` *nicht* gesichert wird, da die Daten inzwischen unter einem anderen Namen gespeichert wurden, man aber immer noch `autoLayout` bearbeitet — siehe Fenstertitel in [Virtuoso...].

Regionen festlegen

Bei der Platzierung werden die Standardzellen in Regionen angeordnet. Der Entwerfer hat dabei vielfältige Eingriffsmöglichkeiten.

Anzahl: Bei der Initialisierung wird schon eine Region erzeugt (`default`), in die alle Standardzellen gelegt werden. Es sind aber auch mehrere Platzierungsregionen möglich, bzw. notwendig: wenn Megazellen (RAM, DPRAM, FIFO, ROM...) vorhanden sind oder das Layout weitergehend beeinflusst werden soll.

Verteilung der Standardzellen: Gibt es mehrere Regionen, so können die Zellen automatisch verteilt werden; es kann aber auch genau vorgegeben werden, welche Zellen in welcher Region platziert werden.

Eigenschaften der Region(en): Für jede der Platzierungsregionen kann genau festgelegt werden, ob Standardzellzeilen horizontal oder vertikal laufen, wie „dicht" die Regionen zu füllen sind, etc.

Weitere Information zu diesen Optionen ist der CADENCE Online-Dokumentation zu entnehmen... Im Nachfolgenden wird der „einfache Fall" beschrieben, der bei den meisten Entwürfen auch die besten Ergebnisse liefert. Sind keine Makrozellen im Entwurf vorhanden, dann sollte man jetzt direkt mit Schritt 15 fortfahren!

13. Platzierung von Megazellen

optional: Nur bei der Benutzung generierter Blöcke / Megazellen erforderlich. Diese müssen von Hand platziert werden.

≡ `Instance` = `on` [OSW]
⟨*others*⟩ = `off`

□ `Edit - Move/`⊙`m` [Virtuoso...]

≡ – entsprechend ausfüllen [Move]
Mit dem `move`-Befehl werden die Megazellen (innerhalb der ursprünglichen) Standardzellregion platziert.

$\uparrow_l$ ⟨*megaCell*⟩ [Virtuoso...]

□ `Edit - Properties.../`⊙`q` [Virtuoso...]

≡ `Status` = `placed` [Edit Instance Properties]
Nach Selektion der fertig platzierten Megazelle wird in der Property-Liste ihr Platzierungsstatus umgesetzt.

14. Neueinteilung der Standardzellregionen
optional: Bei Megazellen notwendig, andernfalls möglich ... wird meist nicht benötigt.

≡ `Region` = `on` [OSW]
⟨*others*⟩ = `off`

↑$_l$ ⟨*region*⟩/⊙ `^a` [Virtuoso...]

☐ `Edit - Delete`/⊙ `del` [Virtuoso...]
Waren Megazellen zu platzieren oder werden aus anderen Gründen mehrere Regionen benötigt, wird die ursprüngliche Region `default` gelöscht.

☐ `Create - Region...`/⊙ `r` [Virtuoso...]

≡ `Choose Rows From` = `routingRatio` [Create Region]
`Routing Ratio` = `1` | ⟨*ratioVal*⟩
`Fit Instance ...` = `off`
`Analyze After Create` = `off`
Neue Regionen für die Platzierung der Standardzellen werden erzeugt, dabei ist für jede Region deren Größe mit der Maus einzugeben.

↑$_l$ ⟨*region*⟩/⊙ `^a` [Virtuoso...]
Selektion einer Region.

☐ `Analyze - Floorplan Objects...`/⊙ `i` [Virtuoso...]

≡ – siehe unten [Preview analysis]
Zeigt Größe und Ausnutzung der Platzierungsregion(en) an. Mit `Next` und `Previous` wird zwischen mehreren Regionen umgeschaltet. Um die Regionen zu bearbeiten hat man folgende Befehle zur Verfügung:

manuelle Größenanpassung der Platzierungsregion

↑$_l$ ⟨*region*⟩ [Virtuoso...]
Wird der Cursor an eine Kante der Platzierungsregion gebracht, so verändert sich seine Form und man kann durch Festhalten der linken Maustaste die entsprechende Kante verschieben.

☐ `Update` [Preview analysis]
Aktualisierung der Informationen.

Änderung der Zeilenanzahl/-orientierung

↑$_l$ ⟨*region*⟩ [Virtuoso...]

☐ `Edit - Properties...`/⊙ `q` [Virtuoso...]

≡ `#rows` = ⟨*nr*⟩ [Edit Region
`Row Orientation` = `horizontal` | `vertical`
Properties]
Über Properties der Regionen können die Anzahl der Standardzellzeilen und deren Ausrichtung verändert werden.

☐ `Update` [Preview analysis]
Aktualisierung der Informationen.

Platzierung

15. Platzierung der Standardzellen
Hier stehen zwei unterschiedliche Algorithmen zur Auswahl…

empfohlenes Verfahren

Die „normale" Platzierung arbeitet mit den üblichen Clusterverfahren und liefert für die meisten Entwürfe (klein bis mittelgroß) gute Ergebnisse.

☐ `Place - Automatic...` [Virtuoso...]

≡ [Automatic Placement]

```
Insert Feedthru          = on
Feedthru Library Name    = HRDLIB
Feedthru Master Name     = FEED
Feedthru Master View     = abstract
Placement Snap Grid      = 0.1
Mirror Cells             = on
```

Die automatische Platzierung wird gestartet. Beim Auftreten von Fehlern (üblicherweise: Überlappung der Standardzellen mit Pad- oder Megazellen) müssen die Platzierungsregionen neu eingeteilt werden.

Bei der Platzierung werden automatisch „Feedthrus" eingesetzt, diese Zellen stellen (zusätzliche) vertikale Verbindungen für die spätere Verdrahtung bereit, wodurch die vertikalen Kanäle entlang der Standardzellzeilen entlastet werden.

optimierter Algorithmus

Ein zweiter Algorithmus (QPlace: quadratic placer) wird für besonders große Entwürfe empfohlen oder wenn Zeitbedingungen zu berücksichtigen sind — Stichwort „timing-driven placement". Darüberhinaus kann er benutzt werden, um bereits teil-/platzierte Entwürfe nachzuoptimieren.

☐ `Place - Qplace...` [Virtuoso...]

≡ [CE Qplace]

```
Feed... Library Name     = HRDLIB
Feed... Cell Name List   = FEED
Feed... View Name        = abstract
Routing Layer Names      = MET1 MET2
Mirror Cells             = on
Timing Driven Placement  = off
Qplace Config File       =
Placement Snap Grid      = 0.1
```

16. Zellen für die Spannungsversorgung einfügen
Die Spannungsversorgung der Standardzellzeilen folgt den unter „Power-Routing" (Seite 3) vorgestellten Prinzipien.

Sind mehrere Platzierungsregionen vorhanden, so können diese getrennt behandelt werden, wobei vorher eine Auswahl erfolgen muss. Ist nur eine Region vorhanden, ist die Selektion nicht notwendig.

≡ `Region` = `on` [OSW]
`⟨others⟩` = `off`

↑*l* `⟨region⟩` [Virtuoso...]
Selektion der Region(en) für den nächsten Schritt.

ohne Power-Feeds Standardzellzeilen kürzer als 2 mm

☐ `Place - Power Cell - Add Auto...` [Virtuoso...]

≡ `Use Pre-Defined PwrCells` = `on` [Create and Add Power...]
`Insert Power Bars:`
`Left End of Rows` = `on`
`Right End of Rows` = `on`
`Interior of Region` = `off`
`Power Feeds Row Ends:`
`Left` = `off`
`Right` = `off`
`Insert Interior Bars using:`
`Number of Bars` = `0`
`*** Feedthru and Spacer Cells Information ***`
`Library Name` = `HRDLIB`
`Feedthru and Spacer Cells` = `FEED25 FEED10 FEED5 FEED2 FEED`
↑*l* `Pre-Define Power Cells`

≡ `*** Power Cell Information ***` [Power Cell Definition]
`Library Name` =
`Power Cell Name` =
`*** Cap Cell Information ***`
`Library Name` = `HRDLIB`
`Left Cap Cell Name` = `LCAP`
`Right Cap Cell Name` = `RCAP`
`Terminal Net Mapping File` =
Nach dem Hinzufügen der linken und rechten Cap-Zellen, ist das Ergebnis im Layout zu überprüfen!

mit Power-Feeds Standardzellzeilen länger als 2 mm

☐ `Place - Power Cell - Add Auto...` [Virtuoso...]

≡ `Use Pre-Defined PwrCells` = `on` [Create and Add Power...]

```
Insert Power Bars:
Left End of Rows         = on
Right End of Rows        = on
Interior of Region       = on
Power Feeds Row Ends:
Left                     = off
Right                    = off
Insert Interior Bars using:
Number of Bars           = ⟨nr⟩
*** Feedthru and Spacer Cells Information ***
Library Name             = HRDLIB
Feedthru and Spacer      = FEED25 FEED10 FEED5 FEED2 FEED
Cells
↑l Pre-Define Power Cells
```

Die Anzahl der Power-Bars ⟨*nr*⟩ richtet sich nach der Länge der Standardzellzeilen. Um zu überprüfen wie eine bestimmte Anzahl im Layout aussehen würde gibt es den Befehl ↑$_l$ `Show Power Bars`, beziehungsweise ↑$_l$ `Clear Power Bars` um die Anzeige zu löschen.

≡ `*** Power Cell Information ***` [Power Cell Definition]

```
Library Name             = HRDLIB
Power Cell Name          = ⟨feedCellId⟩    meist: PFEED30
*** Cap Cell Information ***
Library Name             = HRDLIB
Left Cap Cell Name       = LCAP|LCAP2
Right Cap Cell Name      = RCAP|RCAP2
Terminal Net Mapping     =
File
```

Power-Feed Zellen	PFEED30	PFEED40	PFEED50	PFEED100
Metallbreite [μm]	30	40	50	100

Achtung: die Metallbreite dieser Zellen muss später als Breite für `vdd!` und `gnd!` eingetragen werden — Punkt 22.

Nach dem Hinzufügen der Cap-Zellen und Power-Feeds, ist das Ergebnis im Layout zu überprüfen!

17. Ausrichten der I/O-Pads

☐ `Place - IO Commands - Justify...` [Virtuoso...]

≡ `IO Align Style` = `free` [Align IO Frame]

```
Align Feature            = origin
Placement Snap Grid      = 0.1
Shift IO Frame to Origin = on
```

Endgültiges Ausrichten der Zellen des Padkranzes. Anschließend müssen alle Zellen „richtig“ orientiert sein.

18. Ausrichten der Zellen

≡ `Instance` = `on` [OSW]
⟨*others*⟩ = `off`

☐ `Place - Snap to Grid` [Virtuoso...]

≡ `Selected Cells` = `off` [Snap To Grid]
`Placement Snap Grid` = `0.1`

19. Überprüfen der (fertigen) Platzierung

☐ `Place - Check...` [Virtuoso...]

≡ `Check Cells Overlap` = `on` [Placement Checker]
`Check IO or Pins Overlap` = `on`
`Check Gap Between ...` = `on`

Achtung: es dürfen keine Fehler auftreten und *alle* Zellen müssen als P&R-Status `placed` besitzen, andernfalls kann nicht verdrahtet werden. Die Ergebnisse werden im CADENCE Eingabefenster [icfb - Log:...] angezeigt.

20. Sichern?

☐ `Design - Save As` [Virtuoso...]

≡ `View Name` = ⟨*placed*⟩ [Save As]

Zusätzlich kann auch die Platzierung als ASCII Datei gespeichert werden. Damit kann der jetzige Zustand, ausgehend von `autoLayout`, wiederhergestellt werden.

☐ `Floorplan - Floorplan File - Write...` [Virtuoso...]

≡ `File Name` = ⟨*fpFile*⟩ [Save Placement Information]

Sichern des Floorplans — ⟨*fpFile*⟩ kann dann analog dazu wieder eingelesen werden: ☐ `Floorplan - Floorplan File - Read...` [Virtuoso...]

Verdrahtung vorbereiten

21. Verdrahtungskanäle generieren

☐ `Route - Channels - Create...` [Virtuoso...]

≡ – bestätigen [Create Channels]

Die Verdrahtungskanäle werden erzeugt. Sollten später nach der Kanalverdrahtung Probleme mit dem *Alignment* der Padzellen auftreten (s.u. Probleme), so kann der Wert `Initial Cut` explizit festgelegt werden.

Während die Kanäle generiert werden, erhält man im CADENCE Eigabefenster einige Meldungen „`WARNING Contact...`", sie können ignoriert werden.

22. Zuordnung der Netzprioritäten

Dieser Schritt ist notwendig, um die korrekte Verdrahtung der Versorgungsleitungen (Layer und Breite der Vdd- und Gnd-Verbindungen) zu gewährleisten.

☐ `Route - Modify Net - Modify Net Properties...` [Virtuoso...]

In der jetzt erscheinenden Fill-Form sind, abhängig von der „Art des Designs" (pad- oder core-bestimmt, Seite 2), unterschiedliche Einträge vorzunehmen.

Tipp: Anstatt die Fill-Form mehrmals hintereinander aufzurufen, kann man die Eingaben mit `Apply` bestätigen und nimmt nur beim letzten Ausfüllen `Ok`.

Achtung: Bei den folgenden Einstellungen gilt, dass die jeweils aufgeführten Optionen aktiviert sind, alle anderen **müssen** deaktiviert sein!

alle Entwürfe

≡ `Net Names` = `vdd! gnd!` [Modify Net Properties]
`Edit Net Width` = ⟨*`widthVal`*⟩ z.B.: `30`
`Edit Net Layer` = `MET1`
`Fix Net Global Route ...` = `on`

Achtung: sind Power-Feeds vorhanden, so muss der Wert ⟨*`widthVal`*⟩ der Metallbreite der Feed Zellen entsprechen.

Pad-bestimmte Entwürfe

≡ `Net Names` = `PVDDR1! PVDDR2! PVSSR1! PVSSR2! PVSSR3!` []
`Edit Net Priority` = `120`
`Edit Net Layer` = `MET2`
`Fix Net Global Route ...` = `on`

≡ `Net Names` = `PVSSR4!` [Modify Net Properties]
`Edit Net Priority` = `120`
`Edit Net Layer` = `MET1`
`Fix Net Global Route ...` = `on`

Core-bestimmte Entwürfe

≡ `Net Names` = `PVDDR1! PVDDR2! PVSSR2!` []
`Edit Net Priority` = `120`
`Edit Net Layer` = `MET2`
`Fix Net Global Route ...` = `on`

≡ `Net Names` = `PVSSR1! PVSSR3!` [Modify Net Properties]
`Edit Net Priority` = `120`
`Edit Net Layer` = `MET1`
`Fix Net Global Route ...` = `on`

Globale Verdrahtung

23. Verdrahtung der Feed-Zellen
optional: Wird benötigt, wenn Power-Feeds in den Standardzellzeilen vorhanden sind. Diese werden jetzt (vertikal) verbunden.

□ `Route - Special Net Route - Route Power Ladders...` [Virtuoso...]

≡ `Routing Mode` = `Create` [Ladder Route]
`Net Name(s)` = `vdd! gnd!`

24. Verdrahtung der Core-Versorgungsspannung
optional: Dieser Schritt *sollte aber durchgeführt werden*, um eine „gute" Verdrahtung der Versorgungsspannung zu erhalten.

Dabei werden, vor dem automatischen Verdrahten der Netze, die Versorgungsspannungen `vdd!` und `gnd!` schon „von Hand" (global) verdrahtet. So wird gewährleistet, dass die Standardzellzeilen entsprechend ihrer Länge „richtig", ein- oder zweiseitig angeschlossen werden — vergleiche „Power-Routing", Seite 3.

empfohlenes Verfahren
Bei dieser Methode werden nur die Kanäle ausgewählt, durch die die Verdrahtung geführt werden soll. Alle Anschlussunkte in diese Kanäle werden dann verbunden.

□ `Route - Special Net Route - Route Rail...` [Virtuoso...]

≡ `Net Name` = `vdd!`|`gnd!` [Rail Routing]
`Routing Mode` = `Create`
`Routing Info`
`Width` = ⟨*widthVal*⟩ z.B.: `30`
`Preferred Layer` = `MET1`
`Spacing` = ⟨*spaceVal*⟩ z.B.: `3.2`

Die Netze `vdd!` und `gnd!` werden hier getrennt behandelt, die Schritte (bis „Connectivity Check") sind für beide Netze getrennt durchzuführen.

⟨*widthVal*⟩ Hier ist der Wert aus Schritt 22 einzutragen, beziehungsweise, wenn Power-Feeds vorhanden sind, deren Metallbreite.

⟨*spaceVal*⟩ gibt den Mindestabstand zu anderen Leitungen an und sollte größer sein als der (technologische) Minimalwert.

Nach der Einstellung der Parameter werden die Kanäle mit der Maus ausgewählt.

$\uparrow_l$ ⟨*channel*⟩ [Virtuoso...]
In dem Layout Fenster wird die schon vorhandene globale Verdrahtung dargestellt und nach Auswahl eines Kanals entsprechend aktualisiert.

Wurden alle Kanäle, entsprechend dem gewünschten Power-Routing, ausgewählt, wird die Eingabe durch den „Connectivity Check" beendet.

$\uparrow_l$ `Connectivity Checker` [Rail Routing]
In dem anschließend erscheinenden Fenster [Global Routing Connectivity Status] sollte stehen: `Net vdd!`|`gnd! is completely routed.`

alternative Methode
In Ausnahmefällen kann auch die interaktive globale Verdrahtung benutzt werden.

□ `Route - Global Route - Interactive Global Route...` [Virtuoso...]
Währen des gesamten Vorgangs erscheinen in dem CADENCE Fenster Ausgaben, die man beachten sollte.

≡ $\uparrow_l$ `Initialize Net...` [Interactive Global Route]

≡ `Net Name` = `vdd!`|`gnd!` [Initialize IGLR]
Achtung: Nach der Initialiserung sollten im Layout die Anschlusspunkte des Netzes gelb markiert sein. Ist nichts zu sehen, so muss in dem Fenster [LSW] die Sichtbarkeit der Pins und aller Layer eingestellt werden. Nach einem Redraw-Befehl müssten dann die Anschlüsse (gelb) zu sehen sein.

≡ $\uparrow_l$ `Point-to-Point` [Interactive Global Route]

$\uparrow_l$ ⟨*exitPoint*⟩ [Virtuoso...]
Anschließend können die einzelnen Punkte mit der Maus verbunden werden. Dabei wird die (mögliche) Verdrahtung angezeigt. Der Befehl ist durch `<Esc>` zu beenden.

≡ $\uparrow_l$ `Exit Net...` [Interactive Global Route]

≡ `Fix Net Global Route ...` = `on` [Exit IGLR]

≡ – bestätigen [Single Net Connectivity Report]

≡ $\uparrow_l$ `DONE` [Interactive Global Route]

Achtung: Die Schritte von `Initialize Net...` bis `Exit Net...` sind für beide Netze `vdd!` und `gnd!` durchzuführen.

25. Globale Verdrahtung

□ `Route - Global Route - Automatic...` [Virtuoso...]

≡ `Method` = `both` [Global Route Method]
$\uparrow_l$ `Automatic`

≡ `Use Stub Routing` = `on` [Automatic Global Route]

≡ `Method` = `both` [Global Route Method]
$\uparrow_l$ `Optimizer`

≡ `Remove Unused Feedthru` = `on` | `off` [Global Route Optimizer]
`Feedthru Library Name` = `HRDLIB` |
`Feedthru Master Name` = `FEED` |
`Feedthru Master View` = `abstract`
`Align Power Cells Name` = ⟨*feedCellId*⟩

Sind im Entwurf Power-Feeds vorhanden, dann muss `Remove Unused Feedthru` ausgeschaltet werden.

Während der globalen Verdrahtung des ICs, erscheinen im CADENCE Eingabefenster die Ausgaben des Programms. Treten hierbei Probleme auf, so können die beiden Teilschritte `auto` und `optimize` getrennt ausgeführt werden.

26. Ansicht der globalen Verdrahtung

optional: Hier wird kontrolliert, wie Netze (global) verdrahtet sind. Dies ist vor allem für die Versorgungsspannungen `vdd!` und `gnd!` interessant, wenn diese nicht manuell bearbeitet wurden (Schritt 24). Es lassen sich daneben aber auch andere Netze anzeigen...

Sind die Ergebnisse der globalen Verdrahtung nicht zufriedenstellend, so müssen einzelne Netze interaktiv gelegt werden — siehe Schritt 24: „alternative Methode".

□ `Route - Global Route - Topology Display...` [Virtuoso...]

≡ `Net Name` = ⟨*netId*⟩ [Net Topology Display]

Ein Netz ⟨*netId*⟩ wird angezeigt. Durch die Eingabe von $\uparrow_l$ `remove` kann die Anzeige gelöscht werden.

Kanalverdrahtung

27. Kanalverdrahtung

□ `Route - Detail Route - Automatic...` [Virtuoso...]

≡ `Compaction Mode` = `automatic` | `rigidIO` [Route And Compact ...]
`Contact Style` = `offCentered`
`Add Conditional Via` = `off`
`Routing Layers` = `2`
$\uparrow_l$ `Compact`

≡ `Placement Snap Grid` = `0.05` [Compact Channels]

Verdrahtung der einzelnen Kanäle, die Ausgaben des Programms erscheinen im CADENCE Eingabefenster.

28. Verdrahtung sichtbar nachen

Wenn man nicht schon vorher die „Darstellungstiefe“ umgeschaltet hat, sieht man die Verdrahtung im Layout noch nicht... Die Kanäle mit den Metallbahnen sind eine zusätzliche Hierarchieebene im Entwurf.

☐ `Design - Options - Display.../⊙e` [Virtuoso...]

≡ `Display Levels` [Display Options]

`From` = `0`

`To` = `3`

Anschließend sind die Leitungen im Layout sichtbar Zoom : ⊙`z`|`Z`|`f` ...

Ruler : ⊙`k`|`K`

- Probleme im Layout

Wenn im Layout „Fehler“ aufgetreten sind (Padzellen im Core, unsinnige `vdd!/gnd!` Verdrahtung...), dann kann man einzelne Schritte wiederholen und dabei die Parameter variieren oder Operationen „von Hand“ durchführen.

Als Ausgangspunkt können vorher gesicherte Versionen ⟨*viewId*⟩ geladen werden oder Schritte gezielt rückgangig gemacht werden. Dabei gilt die entsprechend umgekehrte Reihenfolge:

☐ `Route - Detail Route - Delete...` [Virtuoso...]

☐ `Route - Global Route - Delete...` [Virtuoso...]

☐ `Route - Channels - Delete` [Virtuoso...]

☐ `Edit - Delete By View.../⊙^v` [Library Maneger...]

Bei der Kanalgenerierung werden entsprechende Views für die Kanäle erzeugt, diese können über den `Library Manager` gelöscht werden.

Achtung: da durch die Routing-Operationen auch die Kanäle verändert werden (Form), sollten, um zum platzierten Layout zurückzugelangen, alle Schritte durchgeführt werden — am einfachsten ist es meist, vorherige Versionen, bzw. eine Floorplan-Datei zu laden.

29. Sichern

☐ `Design - Save As` [Virtuoso...]

≡ `View Name` = ⟨*routed*⟩ [Save Design]

Backannotation

optional: Die folgenden Schritte beschreiben den Vorgang der „Backannotation“, also die Extraktion von Leitungsverzögerungen aus dem Layout und deren Benutzung in der (statischen) Timinganalyse und der Simulation.

Leider ist die AMS Dokumentation hier sehr dürftig; insbesondere, wie die Anbindung an weitere Werkzeuge geschieht. Die hier beschriebene Vorgehensweise wurde anhand von Beispielen entwickelt ... was heißt, dass bei anderen Entwürfen noch Probleme auftreten können.

Anmerkung: Die Dateinamen sind in Prinzip frei wählbar, aus Konsistenzgründen sollte jeweils ⟨*topCellId*⟩.⟨*ext*⟩ gewählt werden.

30. Extraktion der parasitären Elemente
Zuerst werden die Leitungskapazitäten und -Widerstände aus dem fertigen Layout extrahiert und als SPF-Datei (reduced Standard Parasitics File) gespeichert.

□ `Analyze - Parasitics - Extract...` [Virtuoso...]

≡ `For Net(s)` = `all in current level` [Extract Parasitics]
`Wire Topology Mode` = `best estimate`
`CLTF Library Search Path` =
`/local/tech1.2/ams/ams_v3.12/artist/AMS_TH06_1`
`CLTF Library` = `HRDLIB`

≡ – bestätigen [Extract All Parasitics]
Meldungen in dem CADENCE Eingabefenster [icfb - Log:...] die sich auf die Spannungsversorgung der Padzellen beziehen, können ignoriert werden — Netze: `PVDDR`... und `PVSSR`...

□ `Analyze - Parasitics - Write Reduced SPF...` [Virtuoso...]

≡ `Write Reduced SPF For Nets` = `all in current level` [Write Reduced SPF File]
`Name Mapping` = `Logical`
`Reduced SPF File Name` = ⟨*spfFile*⟩`.spf`

31. CADENCE IC beenden
Die nachfolgenden Schritte werden direkt über die Kommandozeile gestartet.

□ `File - Exit...` [icfb - Log:...]

≡ – bestätigen [Exit icfb?]

≡ – entsprechend ausfüllen! [Save Cellviews]
Achtung: für alle noch nicht gesicherten Designs wird gefragt, was mit ihnen geschehen soll. Dabei ist darauf zu achten, dass ⟨*topCellId*⟩ `autoLayout` *nicht* gesichert wird, da diese View (vorher) einen anderen Namen bekommen hat — letzter Schritt: `Design - Save As...`

32. SDF-Datei erzeugen
SDF (Standard Delay File) ist das Datenaustauschformat für Gatter- und Leitungsverzögerungen. Dabei werden die zuvor extrahierten RC-Glieder des Layouts in Verzögerungswerte umgerechnet.

> `vi dlc.init` [xterm]
Zuerst muss eine Steuerdatei für die Berechnung der Verzögerungszeiten erstellt werden, die eventuell schon vorhandene Datei ist unbedingt zu überarbeiten und sieht folgendermaßen aus:

```
CTLF_File "/local/tech1.2/ams/ams_v3.12/artist/AMS_TH06_1/HRDLIB/timing.ctlf"
Library_Name "⟨libId⟩"
Cell_Name "⟨topCellId⟩"
View_Name "schematic"
Switch_List "schematic symbol"
Parasitic_File "⟨spfFile⟩.spf"
Report_Verbose
SDF_Version 2.1
SDF_File "⟨sdfFile⟩.sdf"
Warn_Max_RF_Time (3.0 3.0 3.0)
Error_Max_RF_Time (5.0 5.0 5.0)
```

```
Warn_Max_Out_Load (10.0 10.0 10.0)
Error_Max_Out_Load (20.0 20.0 20.0)
Default_Risetime (0.0 0.0 0.0)
Default_Falltime (0.0 0.0 0.0)
Default_Outcap (0.1 0.1 0.1)
Mode "ALL"
Net_Category "⟨loadModel⟩"                          AMS-Modell: 10k|30k|100k
#
#   Name           Process     Temp    Volt
#   -------------------------------------------
#   WORST-MIL         1.27   125.00    4.50
#   WORST-IND         1.27    85.00    4.50
#   WORST             1.27    75.00    4.50
#   TYPICAL           1.00    25.00    5.00
#   BEST              0.78     0.00    5.50
#   BEST-IND          0.78   -40.00    5.50
#   BEST-MIL          0.78   -50.00    5.50
#
Temperature (0 25 75)                                    ist ggf. anzupassen
Voltage (5.5 5.0 4.5)                                                –"–
Proc_Var (0.78 1.0 1.27)                                             –"–
```

\> `dlc > dlc.log` [xterm]

Die Berechnung der SDF-Datei dauert *sehr* lange — Zeiten über 10 Min. sind normal. Anschließend sollte man sich die Ausgaben in `dlc.log` ansehen.

Diese SDF-Datei kann jetzt von den SYNOPSYS-Werkzeugen benutzt werden, um etweder in einer (statischen) Timing-Analyse Taktraten und Pfadverzögerungen zu bestimmen oder in einer Simulation der Gatternetzliste mit „genauen“ Verzögerungszeiten zu arbeiten. Beide Vorgehensweisen sind im weiteren skizziert.

Timing-Analyse SYNOPSYS Design Analyzer

- Daten einlesen

\> `ams_synopsys` [xterm]

□ `File - Read...` [Synopsys Design Analyzer]

≡ `File Names(s)` = ⟨*vlogFile*⟩`.v` [Read File]
`File Format` = `Verilog`

Hier ist es am einfachsten die Verilog-Datei einzulesen (wegen der Namensänderungen).

$\uparrow_l$ ⟨*topCell*⟩ [Synopsys Design Analyzer]

□ `File - Import - Design Timing...` [Synopsys Design Analyzer]

≡ `Instance` = [Read Design Timing Information]
`Timing File Name` = ⟨*sdfFile*⟩`.sdf`
`Convention` = `Verilog`
`Load delay location` = `Cell`

- Nach der Verarbeitung der SDF-Datei, kann die Timinganalyse durchgeführt werden, siehe dazu „VHDL-Synthese“ – Schritt:
  9. Operationsbedingungen auswählen
  10. ggf. Clock-Attribute setzen
  12. Timing-Report, Path-Timing...
- Danach kann SYNOPSYS Design Analyzer beendet werden...
  □ `File - Quit` [Synopsys Design Analyzer]
  ≡ – bestätigen [Quit Design Analyzer]

Simulation der Netzliste mit den extrahierten Leitungslaufzeiten
Ergänzend zu „VHDL-Synthese — Simulation der Netzliste“ gilt:

14. Namenskonvertierungen durchführen
    Dieser Punkt *muss* vor Beginn des Layouts durchgeführt worden sein, andernfalls „passen“ die Netznamen nicht und der gesamte Layoutvorgang ist zu wiederholen!
15. VHDL-Netzliste schreiben
    Die Gatternetzliste sollte schon bei der Synthese erzeugt werden. Ist sie aber noch nicht vorhanden, kann sie mit den folgenden Befehlen geschrieben werden:
    \> `ams_synopsys -shell` [xterm]
    ▷ `read -format verilog ⟨vlogFile⟩.v` [xterm]
    ▷ `current_design = ⟨topCellId⟩` [xterm]
    ▷ `write -format vhdl -hierarchy -output ⟨vhdlFile⟩.vhd` [xterm]
    ▷ `quit` [xterm]
16. Simulation vorbereiten
    Wenn noch nicht geschehen, sind die VHDL-Netzliste und die „passende“ Konfiguration anzupassen.

Die weiteren Schritte sind hier im einzelnen beschrieben...

- Bibliotheken austauschen
  Vor der eigentlichen Simulation sind andere Bibliotheksmodelle der Standardzellen einzustellen. Dies geschieht über die Zuordnung des Bibliotheksnamens zu einem Verzeichnis in der SYNOPSYS Steuerdatei.
  \> `vi .synopsys_vss.setup` [xterm]
  Mit einem Editor muss die Setup-Datei so bearbeitet werden, dass die Pfade umgesetzt werden:
  ```
  -- cub  : /local/tech1.2/ams/vhdlLib/cub/FTGS      auskommentieren
  cub     : /local/tech1.2/ams/vhdlLib/cub/VITAL     neuer Pfad
  ```
  Die VITAL-Bibliotheken (*V*HDL *I*nitiative *T*owards *A*SIC *L*ibraries) ermöglichen den Austausch der Timing-Information durch eine SDF-Datei; allerdings sind sie in der Simulationsabarbeitung „langsamer“ als die vorher benutzten SYNOPSYS FTGS-Modelle.
- VHDL-Codeanalyse
  Die VHDL-Dateien *müssen*, wegen der neuen Suchpfade, neu analysiert werden.
  \> `vhdlan ⟨vhdlFile⟩ ⟨vhdlCfgFile⟩` [xterm]

- Simulation der Schaltung / der Testumgebung
  Beim Start der Simulation werden in der Kommandozeile die Parameter für die SDF-Backannotation angegeben.

  > `vhdldbx` [xterm]

  ≡ `Library` = `DEFAULT` [Vhdldbx - Select...]
  `Design` = ⟨*configId*⟩
  `Time Units` = `ps`|`ns`
  `Arguments` = `-sdf_top` ⟨*uutInstId*⟩ [⟨*sdfDelay*⟩]
  `-sdf` ⟨*sdfFile*⟩`.sdf`

  ⟨*uutInstId*⟩ ist der Pfad zu der instantiierten Entity innerhalb der Testumgebung. Die Bezeichner in der SDF-Datei beziehen sich auf den entworfenen Chip ⟨*topCellId*⟩; in der Simulation hingegen wird eine Testumgebung, genauer deren Konfiguration, mit einer Instanz des Chips simuliert. ⟨*uutInstId*⟩ entspricht dem „Offset“ der Namen, dazu ein Beispiel:

```
entity testEnv is                                  Entity - Testumgebung
end testEnv;

architecture test of testEnv is                    Architecture - Testumgebung
  ...
begin
  uutI: chip port map (...);                       Instanz des Chips
  ...
end test;

configuration cfgTestEnv of testEnv is             Configuration - Testumgebung
  for test
    for uutI: chip use entity ...
    end for;
  end for;
end cfgTestEnv;
```

  Parameter: `-sdf_top /testEnv/uutI` Pfad zu `chip`

  ⟨*sdfDelay*⟩ wählt (optional) die Verzögerungsmodelle der SDF-Datei aus. Voreingestellt ist der der typische-Wert, es können aber auch Min- oder Max-Modelle explizit angegeben werden.

⟨*sdfDelay*⟩ ::=	`-sdf_min`	minimale Verzögerungszeit
	`-sdf_typ`	typische –”–
	`-sdf_max`	maximale –”–

  Achtung: Sind für ⟨*uutInstId*⟩ falsche Pfade angegeben oder konnte die SDF-Datei nicht geöffnet werden, dann gibt der Simulator entsprechende Fehlermeldungen aus.

  War die Backannotation erfolgreich, erscheint zu Beginn der Simulation eine Meldung:

  `Overlaying SDF file: "`⟨*sdfFile*⟩`.sdf" with timing index: TYPICAL|MIN|MAX`