Von einer VHDL-Beschreibung zum Layout –

Einführung in Silicon Ensemble mit Beispiel

Inhaltsverzeichnis

1 Synthese

1.1 Vorbereitungen

Hierzu legt ihr euch am besten ein neues Verzeichnis z.B. **synopsys** mit mkdir an. In das Verzeichnis sollte ein Unterverzeichnis **WORK** angelegt werden, das wird vom Design-Analyzer benötigt. Wenn das WORK-Verzeichnis nicht existiert funktioniert der Design-Analyzer zwar auch, aber er schreibt eine Unmenge von Dateien direkt ins zuvor angelegte Verzeichnis **synopsys**.

Weiterhin benötigt der Design-Analyzer zwei Startdateien, die **.synopsys_dc.setup** und die **.synopsys_vss.setup**, diese solltet ihr noch von der VHDL-Übung haben. Sie müssen für die Synthese auf die UMC18-Zielarchitektur noch etwas verändert werden. In der **.synopsys_dc.setup** werden die Link-, Target- und Symbollibrary für die UMC18-Zielarchitektur eingebunden. Die Link- und die Targetlibrary sind die gleichen, ihr findet sie unter:

/opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/db/umcl18u250t2_typ.db

/opt/des_kits/UMC/0.18/UMCL18U300D2_1.3/db/umcl18u300t2_typ.db

/opt/des_kits/UMC/0.18/UMCL18U350D2_1.2/db/umcl18u350t2_typ.db

Diese Dateien sind für die laufende Synopsys-Version erstellt. Bei Problemen kann testweise der Pfad **db** durch den Pfad **design_compiler** ersetzt werden.

Dabei enthält die umcl18u250t2-Datei die Informationen für die Standardzellen und die beiden anderen Dateien die Informationen für die Padzellen (PAD-limitiert: umcl18u300t2, CORE-limitiert: umcl18u350t2). Die Symbollibrary ist in dem gleichen Pfad zu finden und heißt z.B. **umcl18u250t2.sdb**. Außerdem muss in der **.synopsys_vss.setup** ebenfalls der Pfad für die Pad- und Standardzellen angegeben werden. Dieser Pfad ist der Gleiche wie in der **.synopsys_dc.setup**.

1.2 VHDL-Toplevel-Beschreibung

Für die Synthese wird eine VHDL-Beschreibung des Designs benötigt. Wenn das Design in VHDL beschrieben ist, müssen nun noch die Padzellen als Komponenten eingebunden werden und mit den Ein- und Ausgängen verknüpft werden. Dazu schreibt man der Übersicht halber eine neue VHDL-Beschreibung für den Top-Level. In diese Beschreibung kommt zusätzlich die Angabe der Synthesebibliotheken in folgender Form:

```
-- synopsys translate_off

library umcl18u250t2_typ;
```

```
library umcl18u300t2_typ;
library umcl18u350t2_typ;
-- synopsys translate_on
```

In der Architekturbeschreibung werden die Padzellen als Komponenten eingebunden

```
component C3I40
  port(
    PAD : in    STD_LOGIC;
    DI  : out   STD_LOGIC);
end component;
```

und anschließend wie folgt verbunden:

```
bus: for i in 0 to 7 generate
      A_pad : C3I40 port map (
            summand_a(i), a(i));
end generate bus;
```

Folgende Padzellen stehen dabei in der umcl18u300t2 Bibliothek (PAD-limitiert) zur Verfügung:

- C3I40 – Input Pad;
- C3O10 – Output Pad;
- C3B10 – Bidirektionales Pad;
- C18C32 – Clock Pad;
- VVSS – Ground Pad;
- VVDD – Power Pad;
- ANIO – Analog Pad

Bei den Pads aus der umcl18u350t2 Bibliothek (CORE-limitiert) steht immer noch ein „W“ (wide) zusätzlich vor dem Namen, z.B. WC3I40 – Input Pad.

1.3 Synthese mit dem Design-Analyzer

Damit ist die Top-Level-Beschreibung des Designs fertig gestellt und es kann mit der Synthese begonnen werden. Dazu wechselt ihr in das dazu angelegte Verzeichnis und startet den Design-Analyzer mit **`design_analyzer`**. Um alle Aktivitäten des Design-Analyzers verfolgen zu können, empfiehlt es sich, das Command Window unter

```
Setup -> Command Window
```

zu öffnen. Hier können Befehle per Hand eingeben und gleichzeitig als Kontrolle die Arbeitsschritte und die eventuelle Ausgabe von Fehlern beobachtet werden. Da der Vorgang der Synthese auf die Zielarchitektur möglichst automatisch ablaufen soll, sollte ein Script geschrieben werden. Das sieht dann z.B. folgendermaßen aus:

```
/* Designer and Company setzen - nicht unbedingt noetig */
designer = "Fiedler"
company  = "Uni Rostock"

/* Variable TOP den Namen der Top-Level-Entity zuweisen */
/* Top-Level = oberste Hierarchieebene */
TOP = adder_top
remove_design -all

/* Namen der VHDL-Datei eintragen */
/* Analysieren des VHDL-Files und in die work-Library schreiben */
analyze -format vhdl -lib work adder.vhd
analyze -format vhdl -lib work TOP + ".vhd"

/* Design TOP aus der work-Library Elaborieren */
elaborate TOP -arch TOP + "_arch" -lib WORK -update

/* Aktuelles Design auswaehlen */
current_design TOP

/* falls mehrere Instanzen einer Komponente eingebaut wurden muessen */
/* diese mit "uniquify" durchnummeriert werden */
uniquify

/* Auswaehlen welche Ports (Anschluesse) des Designs auf die */
/* Pads des Chips gefuehrt werden sollen */
/* "*" bedeutet alle */
set_port_is_pad "*"

/* Alle Timing- und Area-Constraints entfernen */
/* -> Keine besonderen Optimierungen */
remove_constraint -all

/* Synthetisieren und Optimieren*/
compile -map_effort med

/* Namenskonvertierung für die Verilog-Netzliste */
change_names -rule verilog
```

```
/* Verilog-Netzliste speichern */
write -format verilog -hierarchy -output TOP + ".v"
```

Ausgeführt werden kann das Skript im Design-Analyzer durch:

```
Setup -> Execute Script
```

Durch die Synthese ist eine Verilognetzliste entstanden, mit der die Arbeit des Layouts aufgenommen werden kann.

2 Silicon Ensemble

Hier soll eine Einführung in Silicon Ensemble der Firma Cadence gegeben werden. Dabei handelt es sich um ein Tool für die Umsetzung einer Netzliste in ein entsprechend gelayoutetes Design. Zunächst sei hier der grobe Design Flow genannt, dazu gehören die folgenden Schritte: Floorplanning, Placement, Clock-Tree-Synthese, Routing, Analyse und Verifikation. Bei dieser Einführung wird grundsätzlich in der linken Spalte immer das Fenster angegeben, in dem die Anweisungen der rechten Spalte ausgeführt werden.

Welche Dateien werden zum Design Flow benötigt? Als erstes natürlich die synthetisierte Netzliste im Verilog-Format. Diese enthält alle Instanzen und deren Verbindungen zueinander. In der Regel ist sie hierarchisch aufgebaut, d.h. innerhalb der Netzliste existieren viele Komponenten, welche dann auf unterschiedlichster Weise mit einander verknüpft sind und andere in der Hierarchie höher liegende Blöcke bilden. Weiterhin benötigt Silicon Ensemble LEF-Dateien, dabei handelt es sich um ein Library Exchange Format welches für jede Art von Makrozellen benötigt wird. Sie beinhaltet unter anderem die Größe der Zelle, die existierenden Pins, die Blockages, die Ausrichtung und noch einige andere Werte, sie ist damit technologieabhängig. Was aber nicht in der LEF-Datei steht sind die Timinginformationen der einzelnen Zellen. Diese müssen für jede Zelle durch eine TLF-Datei importiert werden. TLF steht für Timing Library File, darin sind für alle Zellen die Timinginformationen enthalten. Die Verzögerungszeit wird dabei durch eine Tabelle in Abhängigkeit von der Last am Ausgang und der Steilheit der Eingangsflanke bestimmt. Man unterscheidet bei den Timinginformationen zwischen drei Fällen Best, Typical und Worst.

Diese Unterscheidungen ergeben sich durch die Spannung an den Versorgungspads, der Umgebungstemperatur und der Prozessstreuung. Diese drei Parameter variieren in einem bestimmten Bereich, wobei das für den UMC18-Prozess wie folgt aussieht.

	Worst-Case	Typical-Case	Best-Case
Spannung [V]	1,62	1,8	1,98
Temperatur [°C]	125	25	0
Prozessstreuung	0,8	1	1,2

Diese Fall-Unterscheidung ist für die spätere Timinganalyse wichtig. Dort wird das Design auf Setup- und Holdverstöße untersucht. Ein Setupverstoß tritt auf wenn ein Signal von einem Flipflop zum nächsten länger braucht, als das Clocksignal minus der Setuptime des empfangenden FF. Hier ist also die Betrachtung der Gatter im Worst-Case notwendig, denn durch die geringere Spannung und höhere Temperatur ist die Signallaufzeit auch höher. Setupverstöße treten in der Regel bei den langen Pfaden auf oder wenn eine sehr hohe Last getrieben werden muss. Anders ist es bei einem Holdtimeverstoß. Dort sind die Daten des sendenden Flipflops noch vor dem Ablauf der Holdtime des empfangenden Flipflops angekommen. Damit gerät das empfangende Flipflop in einen unsicheren Zustand und ein Fehler tritt auf. Für die Analyse von Holdtimeverstößen ist also die Betrachtung im Best-Case nötig, da dort durch die höhere Spannung und geringere Temperatur die Signallaufzeiten geringer sind.

Um dem Design Vorgaben über die Clocksignale, das gewünschte Zeitverhalten, externe Lasten oder den Arbeitsbedingungen für den Chip zu geben wird eine GCF-Datei importiert. GCF heißt Gobal Constraint File und enthält unter anderem auch einen Link auf die TLF-Dateien.

Weiterhin benutzt Silicon Ensemble DEF Formate, was für Design Exchange Format steht. DEF beinhaltet alle Informationen aus der Netzliste, sowie zusätzliche Informationen aus der Layout-Sicht. Dazu gehören zum Beispiel die Koordinaten der Zellen sowie deren Ausrichtung, aber auch welche Spezialnets existieren bis hin zu Angaben über das Routing des Designs. Damit wurde alle wichtigen Dateiformate vorgestellt, einige andere sind: sdc, sdf, rspf, mac, gds2, …

2.1 Datenvorbereitung – automatisch

Um Silicon Ensemble zu starten sind verschiedenste Einstellungen notwendig. Diese können von Hand gemacht werden oder aber es wird ein vorbereitetes Script benutzt. Im nachfolgenden Kapitel wird dann die manuelle Vorgehensweise beschrieben.

Für die automatische Ausführung zunächst das Startscript in das Arbeitsverzeichnis kopieren:

```
cp /opt/des_kits/UMC/0.18/MD_Vorlagen/start.sh .
```

Nun kann das Startscript mit `source start.sh` oder `./start.sh` ausgeführt werden. Das Script legt automatisch die notwendige Verzeichnisstruktur an. Zusätzlich

kopiert es verschiedene Dateien, weshalb der erste Aufruf einige Zeit in Anspruch nimmt.

2.2 Datenvorbereitung – manuell

Zu erst müssen in Euer `.cshrc` noch einige Pfade für Cadence gesetzt werden. Folgende Zeilen müssen in die `.cshrc` hinzugefügt werden:

```
source /opt/cadence/ic4.46/start.sh

      setenv PATH /opt/cadence/dsmse5.3/tools/dsm/bin:

                  /opt/cadence/dsmse5.3/tools/bin:

                  /opt/cadence/dsmse5.3/tools/dfII/bin:$PATH
```

Die andere Möglichkeit besteht darin sich die `.cshrc` aus folgendem Home-Verzeichnis zu kopieren: `/home/gf94/.csrhc`

Der zweite Schritt ist das Erzeugen der Verzeichnisstruktur für Silicon Ensemble, dazu wird ein Verzeichnis `cadence_se` angelegt. In diesem Verzeichnis müssen folgende Unterverzeichnisse `work, def, lef` und `netlist` angelegt werden, damit die Daten möglichst gut überschaubar bleiben. Alle DEF-Dateien kommen ins `def` Verzeichnis und so weiter. Für die Arbeit mit Silicon Ensemble ist das `work` Verzeichnis angelegt worden, darin muss noch das Verzeichnis `dbs` angelegt werden. In das `..cadence_se/work/dbs/` Verzeichnis werden alle Designs gespeichert. Zum Starten von Silicon Ensemble wird die Initialisierungsdatei se.ini und gsd2.map benötigt, diese können aus dem Pfad

`/opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/silicon_ensemble/`

ins `../cadence_se/work/` Verzeichnis kopiert werden. Da der Link für die TLF-Datei in der umcl18u250t2_typ_sample.gcf nicht stimmt muss diese Datei auch von dem Pfad

`/opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/tlf/`

ins `../cadence_se/` Verzeichnis kopiert und anschließend bearbeitet werden. Dazu müssen die Pfade innerhalb der Datei in etwa wie folgt verändert werden:

```
(operating_conditions "typical" 1.0 1.8 25)

(extension "TLF_FILES"

            (

      /opt/des_kits/UMC/0.18/UMCL18U300D2_1.3/tlf/umcl18u300t2_typ_3.1.tlf

      /opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/tlf/umcl18u250t2_typ.tlf

            ) )
```

Die Namen müssen an die Bibliotheken, mit denen gearbeitet werden soll, angepasst werden. Je nach gewählter PAD-Bibliothek und je nach gewählten Zeitwerten (typ – typical, wc – worst case, bc – best case) unterscheiden sich diese Angaben.

2.3 Starten von Silicon Ensemble

Silicon Ensemble ist immer aus dem Verzeichnis **../cadence_se/work/** zu starten. Um Silicon Ensemble zu starten wird in die Unix - Shell **sedsm** eingegeben. Durch Eingabe von **sedsm -h** werden unterschiedliche Startoptionen angezeigt. Zum Beispiel startet Silicon Ensemble bei der Eingabe von **sedsm -m=200&** mit einem Arbeitsspeicher von 200 MB.

2.4 Importieren des Designs

Hier ist Reihenfolge des Importierens der einzelnen Designdaten sehr wichtig. Begonnen wird mit dem Einladen der LEF-Dateien der Standard- und der Padzellen.

- ***Silicon Ensemble*:** **File -> Import -> LEF**
- ***Import LEF*:** **Ins Verzeichnis**
  /opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/
  silicon_ensemble/ **wechseln**
  Button Clear Existing Design Data aktivieren
  Datei header_6m_5.3.lef **auswählen**
  Apply
  Button Clear Existing Design Data deaktivieren
  Datei umcl18u250t2.lef **auswählen**
  Apply
- ***Warning*:** **Yes**
- ***Import LEF*:** **Anschließend ins Verzeichnis**
  /opt/des_kits/UMC/0.18/UMCL18U300D2_1.3/
  silicon_ensemble/ **wechseln**
  Datei header_6m_5.3.lef **auswählen**
  Apply
- ***Warning*:** **Yes**
- ***Import LEF*:** **Datei** umcl18u300t2.lef **auswählen**
  OK
- ***Warning*:** **Yes**

Im nächsten Schritt werden die Timinginformationen in Form einer GCF-Datei eingebunden.

- ***Silicon Ensemble*:** **File -> Import -> Timing Library**
- ***Import Timing Library*:** **Aus dem cadence_se-Verzeichnis die erstellte .gcf (siehe 2.2) auswählen**
  Wurde Cadence automatisch gestartet, befinden sich die Dateien im Pfad gcf
  OK

Jetzt lohnt es sich, das Design schon einmal zu speichern. Dabei wird das Design standardmäßig ins **../cadence_se/work/dbs/** Verzeichnis geschrieben.

- ***Silicon Ensemble*:** **File -> Save as**
- ***Save as*:** **Einen beliebigen Designnamen eingeben**
  OK

Als vorletzten Schritt des Designimportierens müssen zwei initialisierende Verilog-Netzliste eingeladen werden (für die Core- und den PAD-Zellen).

- ***Silicon Ensemble*:** **File -> Import -> Verilog**
- ***Import Verilog*:** **Auf Browse klicken**
- ***MultiBrowse*:** **Ins Verzeichnis** /opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/silicon_ensemble/ **wechseln**
  Datei umcl18u250t2_floorplan.v **auswählen**
  Add
  OK
- ***Import Verilog*:** **Verilog Top Modul leer stehen lassen**
  OK
- ***Warning*:** **Yes**

Das gleiche je nachdem für die Dateien umcl18u300t2_floorplan.v und umcl18u350t2_floorplan.v aus den entsprechenden Pfaden durchführen.

Nun kann abschließend die eigentliche Verilognetzliste importiert werden. Dabei wird ähnlich wie bei dem vorherigen Schritt vorgegangen.

- ***Silicon Ensemble*:** **File -> Import -> Verilog**
- ***Import Verilog*:** **Auf Browse klicken**
- ***MultiBrowse*:** **Ins Verzeichnis**
  /cadence_se/netlist/ **wechseln**
  Datei design.v **z.B.** adder_top.v **auswählen**
  Add
  OK
- ***Import Verilog*:** **in Verilog Top Modul das Topmodul z.B.**
  adder_top eintragen
  evtl. Nets anpassen
  OK
- ***Warning*:** **Yes**

Damit sind die Designdaten erfolgreich importiert worden und es sollte an dieser Stelle wieder gespeichert werden. Dies kann entweder durch:

- ***Silicon Ensemble*:** **File -> Save as**
- ***Save as*:** **Einen beliebigen Designnamen eingeben**
  OK

geschehen oder wenn der Name nicht verändert werden soll einfach durch:

- ***Silicon Ensemble*:** **File -> Save**

2.5 Floorplanning

Nun sind alle notwendigen Designdaten importiert und es kann mit dem Floorplanning begonnen werden. Als erster Schritt wird bei der Initialisierung des Floorplan die Chipgröße festgelegt. Bei der Festlegung der Chipgröße spielen drei unterschiedliche Faktoren eine große Rolle. Das Design kann CORElimitiert sein, d.h. es sind so viele Standardzellen vorhanden, dass dadurch die Größe bestimmt wird. Ein anderer Fall ist PADlimitiert, dabei ist die Chipgröße allein durch die Anzahl

und Größe der Padzellen bestimmt. Weiterhin ist es möglich, dass ein Design BLOCKlimitiert ist, wenn die Größe der internen Makroblöcke (Memory, PLL, DAC) die Chipgröße bestimmt. Weiterhin wird beim Floorplanning das Coregebiet festgelegt, dazu werden Rows platziert in denen später die Standardzellen platziert werden. Die Rows sind quasi Zeilen mit einer Höhe die der Standardzellhöhe entspricht und einer beliebigen Breite.

- ***Silicon Ensemble*:** **Floorplan -> Initialize Floorplan**
- ***Initialize Floorplan*:** **Im Unterfenster Die Size Constraint werden die Abmaße des Chips festgelegt.**

  Weiterhin kann im IO to Core Distanz - Fenster der Abstand der Corefläche von den IOPads angegeben werden.

  Im Fenster Core Area Parameters können vorgaben für das Coregebiet getroffen werden.

  Calculate

  OK

Damit ist der Floorplan initialisiert und die Pads können platziert werden. Dazu wird eine Datei im DEF-Format geschrieben, die die Koordinaten und Ausrichtungen der Padzellen enthält. Damit Euch das schreiben der Datei nicht so schwer fällt, kann man sich durch:

- ***Silicon Ensemble*:** **File -> Export -> DEF**
- ***Export DEF*:** **Aktivieren von Cells**

  Angeben eines Dateinamens

  OK

eine DEF-Datei erzeugen lassen in der schon alle Signalpads existieren. Diese Datei muss anschließend überarbeitet werden, das sieht in etwa so aus:

```
COMPONENTS 35 ;
...
- U100 CORNER + PLACED ( -444800 150100 ) E ;
- U101 CORNER + PLACED ( -444800 -424400 ) N ;
- U8 C3O10B + PLACED ( -444800 -60000 ) E ;
- U9 C3O10B + PLACED ( -444800 -102000 ) E ;
- U10 C3O10B + PLACED ( -444800 -144000 ) E ;
```

```
- U11 C3O10B + PLACED ( -144000 -424400 ) N ;
- U12 C3O10B + PLACED ( -102000 -424400 ) N ;
...
END COMPONENTS
```

Diese Datei ist im **../cadence_se/def/** - Pfad zu speichern und wird wie folgt in Silicon Ensemble importiert.

- ***Silicon Ensemble*:** **File -> Import -> DEF**
- ***Import DEF:*** **In den** /cadence_se/def/ **Pfad wechseln**
  Geschriebene DEF-Datei auswählen
  OK

Damit sind die Pads platziert und es kann mit dem nächsten Schritt der Verlegung der Powerleitungen begonnen werden. Als erstes wird ein Power- und Groundring um das Coregebiet gelegt.

- ***Silicon Ensemble*:** **Route -> Plan Power**
- ***Plan Power:*** **Add Rings**
- ***PP Add Rings:*** **angeben der Netznamen** „vdd!“ „gnd!“
  Auswahl der Metalllayer
  Festlegen der Breite der Versorgungsleitungen
  Core Ring Width 10.000
  Festlegen des Abstandes vom Coregebiet
  Core Ring Spacing 2.000
  OK
- ***Plan Power:*** **Close**

Der folgende Schritt ist das Verbinden des Ringes mit den Power- und Groungpads sowie das Followpinrouting. Darunter versteht man das horizontale Verlegen von Power- und Groundleitungen durch das Coregebiet, in der Weise das alle platzierten Standardzellen versorgt werden.

- ***Silicon Ensemble*:** **Route -> Connect Ring**
- ***Connect Ring:*** **IO Pad und All Ports aktivieren**

Followpin aktivieren

Alle anderen deaktivieren

OK

Damit sind alle notwendigen Powerversorgungen verlegt worden, womit das Floorplanning abgeschlossen ist.

2.6 Placement

Im folgenden Schritt werden die Standardzellen in den Rows platziert.

- ***Silicon Ensemble*:** **Place -> Place Cells**
- ***Place Cells:*** **OK**

2.7 Routing

Anschließend folgt das Routing, also das Verbinden der Signalleitungen im Layout. Dazu wird der WarpRouter benutzt, der in zwei Phasen arbeitet: dem Gobal und dem Final Route.

- ***Silicon Ensemble*:** **Route -> WRoute**
- ***WRoute:*** **OK**

Damit ist das Design gelayoutet, es fehlen jetzt unter anderem noch die abschließenden Schritte Timing Analyse, Physical Verification und das Exportieren einer GDS2-Datei.

3 Silicon Ensemble - Workshop

Hierbei soll ein Layout von einem Zähler erstellt werden. Ausgangspunkt dazu ist eine vorhandene Verilognetzliste. Als ersten Schritt muss ein Arbeitsverzeichnis z.B. **workshop** in eurem Unix-Home-Verzeichnis angelegt werden:

mkdir workshop

Anschließend wechselt ihr in das Verzeichnis und kopiert euch folgendes Skript:

cp /opt/des_kits/UMC/0.18/MD_Vorlagen/start.sh .

Dieses Skript legt die notwendige Verzeichnisstruktur im **workshop**-Verzeichnis an und kopiert alle erforderlichen Startdateien für Silicon Ensemble. Dazu gehört unter anderem auch die Verilognetzliste des Zählers. Sind alle Dateien kopiert, wird Silicon Ensemble automatisch gestartet. Aufgerufen wir das Skript einfach mit:

source start.sh

Durch das Skript wurden schon 2 abgespeicherte Designvorlagen mitkopiert, eine für Core- und eine für Pad-limitierte Designs. Bei unserem Zähler handelt es sich um ein Pad-limitiertes Design, worauf beim einladen geachtet werden muss.

- ***Silicon Ensemble*:** **File -> Open**
- ***File Open*:** LEER_CORE_1.3_PAD_TIME_FLOORPLAN **auswählen**
  OK

Nun folgt das Importieren der Verilognetzliste:

- ***Silicon Ensemble*:** **File -> Import -> Verilog**
- ***Import Verilog*:** **Auf Browse klicken**
- ***MultiBrowse*:** **Ins Verzeichnis**
  /workshop/netlist/ **wechseln**
  Datei counter_top.v **auswählen**
  Add -> OK
- ***Import Verilog*:** **Verilog Top Modul:** counter_top **eintragen**
  OK

Nun kann das Design schon einmal abgespeichert werden. Dabei ist zu kontrollieren, dass das Design im richtigen Pfad **/workshop/work/dbs/** gespeichert wird:

- ***Silicon Ensemble*:** **File -> Save as**
- ***Save as*:** **Design Name: z.B.** counter
  OK

Danach kommt das Initialisieren des Floorplans, wobei die Chipgröße, der Abstand von den Pads zum Coregebiet und die Ausrichtung der Rows vorgegeben werden:

- ***Silicon Ensemble*:** **Floorplan -> Initialize Floorplan**
- ***Initialize Floorplan*:** **Im Unterfenster „Die Size Constraint":**
  Fixed Size: Height: 720.000
  Width: 720.000
  Im „IO to Core Distanz" - Fenster:
  Left/Right: 165.000
  Top/Bottom: 165.000
  Im Fenster „Core Area Parameters"
  Flip Every Other Row **und** Abut Rows **aktivieren**
  Calculate
  OK

Mit dem Schalter **„Flip Every Other Row"** wird jede zweite Row gespiegelt, wodurch später vdd! und gnd! Leitungen eingespart werden können. Die Taste **„Abut Rows"** eliminiert dabei den Platz zwischen den Rows. Damit ist der Floorplan initialisiert und im nächsten Schritt können die Pads platziert werden. Dazu wird eine DEF-Datei vom jetzigen Design in das def-Verzeichnis exportiert:

- ***Silicon Ensemble*:** **File -> Export -> DEF**
- ***Export DEF*:** **Aktivieren von** Cells **und** Specialnets
  DEF File Name: ../def/counter_prepads.def
  OK

Diese Datei müsste jetzt per Hand angepasst werden, da dieser Schritt aber etwas zeitaufwendig ist, wurde durch das Skript auch schon eine fertige DEF-Datei mit kopiert. Darin wurden die Koordinaten und die Ausrichtungen der Padzellen manuell festgelegt und auch die notwendigen Powerpads hinzugefügt. Weiterhin

wurden auch die Fillerzellen im IO-Bereich platziert. Diese Datei wird jetzt wieder in Silicon Ensemble importiert:

- ***Silicon Ensemble*:** **File -> Import -> DEF**
- ***Import DEF:*** **In den** /workshop/def/ **Pfad wechseln**
  DEF-Datei: counter_postpads.def **auswählen**
  OK

Nach dem Einladen der **„counter_postpads.def"** muss ein Redraw durch **CTRL+R** durchgeführt werden, damit die platzierten Pads sichtbar werden. In den weiteren Schritten wird die Powerverdrahtung für den Chip vorgenommen. Dazu wird zuerst ein Ring aus vdd! und gnd! um das Coregebiet verlegt:

- ***Silicon Ensemble*:** **Route -> Plan Power**
- ***Plan Power:*** **Add Rings**
- ***PP Add Rings:*** **Festlegen der Breite und des Abstandes der Versorgungsleitungen**

	Core Ring Width		Core Ring Spacing
Horizontal:	10.000	Center->From Core	4.000
Vertical:	10.000	Center->From Core	4.000

  OK
- ***Plan Power:*** **Close**

Nun folgen das Verbinden des Ringes mit den Power- und Groundpads, sowie das Followpinrouting. Darunter versteht man das Erstellen von Power- und Groundleitung innerhalb des Coregebietes für die Versorgungspins der Standardzellen.

- ***Silicon Ensemble*:** **Route -> Connect Ring**
- ***Connect Ring:*** IO Pad, All Ports **und** Followpin **aktivieren**
  Stripe, Block **und** IO Ring **deaktivieren**
  OK

Nun sind die Versorgungsleitungen für den Chip verlegt und es können jetzt die Standardzellen in das Coregebiet platziert werden:

- ***Silicon Ensemble*:** **Place -> Cells**
- ***Place Cells:*** **OK**

Da es sich bei einem Zähler um ein Design handelt, welches nicht nur aus kombinatorischer Logik besteht, sondern auch Flipflops enthält ist die Verteilung des Clocksignals von großer Bedeutung. Dabei ist es wichtig, dass die Flipflops das Clocksignal etwa alle zur gleichen Zeit bekommen. Um das zu erreichen wird das Clocknetz in eine baumförmige Inverterstruktur überführt, dieser Schritt wird ClockTreeSynthese genannt. Um eine CTS durchzuführen wird ein extra Tool benötigt. Welches als Eingabe 3 Dateien benötigt, einmal das platzierte Design im DEF-Format:

- ***Silicon Ensemble*:** **File -> Export -> DEF**
- ***Export DEF*:** **Deaktivieren von** Cells **und** Spezialnets

  DEF File Name: ../def/counter_preclk.def

  OK

dann **ctgen.cmd,** das ist die Kommandodatei für CTGEN (ClockTreeGENerator) und **ctgen.constraints** worin einige Bedingung festgelegt sind. Die beiden letzten Dateien wurden auch schon in das **workshop/ctgen/**-Verzeichnis kopiert. Bevor mit der CTS begonnen werden kann muss aber noch überprüft werden ob in der **ctgen.cmd** die richtigen Pfade stehen. Dazu geht ihr in das Terminal in welchem die **start.sh** ausgeführt wurde. Mit **CTRL+Z** wird der aktuelle Prozess gestoppt und dann mit **bg** in den Hintergrund geschoben, damit kann das gleiche Terminal für die ClockTreeSynthese benutzt werden. Um die Pfade in der **ctgen.cmd** zu überprüfen wechselt man in den **workshop/ctgen/**-Pfad und kann dann mit **more ctgen.cmd** in die Datei hineinschauen. Die Pfade sollten wie folgt aussehen:

/home/username/workshop/...

Wenn dort nicht euer Username steht oder ihr am Anfang einen anderen Pfad als **workshop/** erstellt habt, müsst ihr die **ctgen.cmd** entsprechend anpassen. Dazu öffnet ihr die Datei in einem Editor z.B. XEmacs und ändert darin alle Pfade: **/home/username/workshop/...** auf euren Usernamen und evtl. den richtigen Pfad in dem ihr Anfangs die **start.sh** ausgeführt habt. Nun kann aus einem Terminal CTGEN gestartet werden:

ctgentool ctgen.cmd

Dadurch wird eine counter_postclk.def Datei erzeugt, die einen ClockTree enthält. Diese wird nun wieder in Silicon Ensemble importiert. Da die Netzliste durch das Einfügen neuer Zellen und Netze verändert wurde muss die Def-Datei durch Import DEF ECO importiert werden. ECO steht für Engineering Change Order und muss

immer benutzt werden, wenn eine Änderung der Netzlistenstruktur außerhalb von Silicon Ensemble entstanden ist:

- ***Silicon Ensemble*:** **File -> Import -> DEF ECO**
- ***Import DEF ECO:*** **In den** /workshop/def/ **Pfad wechseln**
  DEF-Datei: counter_postclk.def **auswählen**
  OK

Auch hier muss wieder ein Redraw durch **CTRL+R** durchgeführt werden, damit die Veränderungen sichtbar werden. Damit sind die neu hinzugekommenen Zellen für den Clocktree importiert und platziert. Nun können in die noch vorhandenen Zwischenräume im Coregebiet Fillerzellen eingefügt werden. Diese Aufgabe soll durch eine Mac-Datei durchgeführt werden, welche auch schon mit kopiert wurde und sich im Verzeichnis **workshop/mac/** befindet. Zum Ausführen der Mac-Datei:

- ***Silicon Ensemble*:** **File -> Execute**
- ***Execute:*** **In das** /workshop/mac/ **Verzeichnis wechseln**
  Datei: fillcore.mac **auswählen**
  OK

Als folgender Schritt wird das Clocknetz geroutet:

- ***Silicon Ensemble*:** **Route -> Clock Route**
- ***Clock Route:*** **OK**

Jetzt ist der Clocktree erstellt und es kann das Routing der Signalleitungen durchgeführt werden:

- ***Silicon Ensemble*:** **Route -> WRoute**
- ***WRoute:*** **OK**

Damit ist ein Layout von einem Zähler erstellt worden, abschließend kann nun eine GDS2-Datei erstellt werden, welche dann zur Fertigung des Chips benötigt wird:

- ***Silicon Ensemble*:** **File -> Export -> GDSII**
- ***Export GDSII:*** GDS-II File **aktivieren**

counter.gds2 **eingeben**

OK

4 Layout des LEON-Prozessors mit Audio-Core

In diesem Abschnitt wird auf den kompletten Backend-Design-Flow des LEON mit Audio-Core eingegangen. Dabei steht die Erstellung der MAC-Dateien für einen automatisierten Design-Flow im Vordergrund. Die MAC-Datei enthalten dabei Kommandos für Silicon Ensemble, welche dann sequentiell abgearbeitet werden. Um derartige MAC-Dateien zu erstellen, können die entsprechenden Befehle entweder aus dem Manual für Silicon Ensemble entnommen werden oder aber aus der Kontrollausgabe von Silicon Ensemble kopiert werden. In dem Kontrollausgabefenster werden alle per „Mausklick“ ausgeführten Befehle ausgegeben. Damit die Übersichtlichkeit der Dateien erhalten bleibt, sollte man auch für die MAC-Dateien ein Verzeichnis **mac** in seinem **cadence_se** Pfad anlegen. Um dann eine MAC-Datei auszuführen geht man wie folgt vor:

- ***Silicon Ensemble*:** **File -> Execute**
- ***Execute*:** **Ins Verzeichnis**

  /cadence_se/mac/ **wechseln**

  gewünschte Datei *.mac **auswählen**

  OK

Weiterhin wird in diesem Abschnitt auf die Vorhergehensweise für die Platzierung von Memoryblöcken eingegangen, da bei diesem Design 8 solcher Blöcke platziert werden müssen. Wie Silicon Ensemble eingerichtet und gestartet wird, wurde in dem vorherigem Abschnitt bereits dargestellt, wodurch gleich mit dem Importieren der Designdaten fortgefahren werden kann.

4.1 Importieren der Designdaten

Jetzt werden zusätzlich zu den Herstellerdaten der Standardzellen noch die Daten für die verschiedenen Memoryblöcke benötigt. Es ist also notwendig die LEF- die TLF- und die Verilog-Dateien für die Memoryblöcke zu laden. Zuerst werden die entsprechenden LEF-Dateien geladen, anschließend die TLF-Dateien und dann die Verilog-Dateien, genau die gleiche Reihenfolge wie bei den Standardzellen. In Form einer MAC-Datei sieht es dann wie folgt aus:

```
##-- Import Library Data
##-- LEF
FINPUT LEF FILENAME /opt/des_kits/UMC/0.18/UMCL18U300D2_1.3/silicon_ensemble/header_6lm_5.3.lef ;
INPUT LEF FILENAME /opt/des_kits/UMC/0.18/UMCL18U300D2_1.3/silicon_ensemble/umcl18u300t2_6lm.lef ;
INPUT LEF FILENAME /opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/silicon_ensemble/umcl18u250t2.lef ;
INPUT LEF FILENAME /home/fm22/projekt/leon/SPR/R1024X32M4.lef ;
INPUT LEF FILENAME /home/fm22/projekt/leon/SPR/R256X24M4.lef ;
INPUT LEF FILENAME /home/fm22/projekt/leon/TPR/RF136X32M1.lef ;
INPUT LEF FILENAME /home/fm22/projekt/leon/TPR/RF256X32M1.lef ;
```

```
##-- Import Timing Data
##-- GCF File
INPUT CTLF INITFILE "/home/fm22/projekt/leon/cadence_se/gcf/umc_pad_core_ram_clocktiming.gcf" ;
##-- GCF Constraints für Clocksignal
INPUT GCF FILENAME "../gcf/umc_pad_core_ram_clocktiming.gcf" REPORTFILE "importgcf.rpt" ;

##-- Import Design Data
##-- Verilog
INPUT VERILOG FILE "/opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/silicon_ensemble/umcl18u250t2_floorplan.v" LIB
"cds_vbin" REFLIB "cds_vbin " ;
INPUT VERILOG FILE "/home/fm22/projekt/leon/SPR/R1024X32M4.v" LIB "cds_vbin" REFLIB "cds_vbin " ;
INPUT VERILOG FILE "/home/fm22/projekt/leon/SPR/R256X24M4.v" LIB "cds_vbin" REFLIB "cds_vbin " ;
INPUT VERILOG FILE "/home/fm22/projekt/leon/TPR/RF136X32M1.v" LIB "cds_vbin" REFLIB "cds_vbin " ;
INPUT VERILOG FILE "/home/fm22/projekt/leon/TPR/RF256X32M1.v" LIB "cds_vbin" REFLIB "cds_vbin " ;
INPUT VERILOG FILE "../netlist/leon_audio.v" LIB "cds_vbin" REFLIB "cds_vbin " DESIGN
"cds_vbin.leon_audio:hdl" ;

##-- Save design
SAVE "leon_audio_loaded_clockgcf" ;
```

Beim Einlesen der LEF-Dateien ist ersichtlich das der erste Importbefehl FINPUT heißt. Dabei wird die angegebene Datei importiert und alle vorherigen Daten gelöscht. Das entspricht also dem „**Clear Existing Design Data**“ aus Abschnitt 2.4. Die weiteren INPUT-Befehle importieren dann alle notwendigen Daten. Weiterhin ist es für eine an das Layout anschließende Timinganalyse notwendig, ein Clocksignal zu definieren. Das geschieht in der GCF-Datei, welche durch die oben dargestellte MAC-Datei schon eingeladen wurde. Dort ist ersichtlich, dass die GCF-Datei 2-mal eingeladen wurde. Einmal um die notwendigen Timinginformationen in Form von TLF-Dateien für die Zellen zu importieren und zum zweiten um die Bedingungen, also das definierte Clocksignal, einzuladen. Um zu verdeutlichen wie ein Clocksignal definiert wird, sei die GCF-Datei hier einmal dargestellt:

```
(gcf
   (header
      (version "1.2")
      (TIME_SCALE 1.0E-9)
      (CAP_SCALE 1.0E-12)
   )
   (globals
      (globals_subset timing
       (waveform "master" 11 (negedge 0) (posedge 5.5))
      )
      (globals_subset environment
         (process 0.80 1.20)
         (voltage 1.62 1.98)
         (temperature 0 125)
         (operating_conditions "typical" 1.0 1.80 25)
         (extension "TLF_FILES"
            (
      Angabe der Pfade der TLF-Dateien für die Standard-, Pad- und
      Memoryzellen
            )
         )
      )
   )
  (CELL ()
    (SUBSET TIMING
      (ENVIRONMENT
      (Clock "master" clk)
      )
    )
```

```
  )
)
```

Die rot markierten Bereiche stellen die Definition für das Clocksignal da. Als erstes wird ein „master“ waveform mit der Periodendauer (11), dem Startzeitpunkt für die negative Flanke (negedge 0) und dem Startzeitpunkt für die positive Flanke (posedge 5.5) definiert. Damit kann ein Clocksignal mit einem beliebigen Duty-Cycle erzeugt werden, wobei alle Zeitangaben in Nanosekunden interpretiert werden. Im zweiten Bereich wird das erzeugte master-Signal mit dem Clockpin „clk“ verbunden.

4.2 Floorplanning

Auch hier sind die grundsätzlichen Aufgaben gleich, hinzu kommt an dieser Stelle das Platzieren und Verdrahten der Memory-Blöcke. Vorher muss aber wie gewohnt die Größe des Chips festgelegt, die Rows für die Standardzellen und die Padzellen platziert werden. Die Befehle für diese Schritte sehen wie folgt aus:

```
##-- Chipgröße und Row Utilization
FINIT FLOORPLAN  rowu 0.85 rowsp 0 blockhalo 2000 f x 2152000 y 2152000 abut xio 65000 yio
65000 ;

##-- Pads mit FillerCells
FINPUT DEF FILENAME "../def/leon_audio_postpads_clockgcf.def" ;
```

Damit wird der Floorplan mit einer Chipgröße von 2152 µm Kantenlänge, einem Abstand von 65 µm von den Padzellen zu dem Coregebiet und geflipten aneinander gereihten Rows initialisiert. Der zweite Befehl lädt die DEF-Datei mit den platzierten Pads. Nun müssen die Memoryblöcke platziert werden, was zum einen automatisch durch Silicon Ensemble, durch manuelles Verschieben mit der Maus oder durch Angeben der Koordinaten und Ausrichtungen für die einzelnen Blöcke geschehen kann. Um ein automatisches Platzieren der Memoryblöcke durchzuführen geht man wie folgt vor:

- ***Silicon Ensemble*:** **Place -> Place Blocks**
- ***Place Blocks:*** **OK**

Die Erfahrungen mit diesem Design zeigten, dass ein automatisches Platzieren von Blöcken kein zufriedenstellendes Resultat lieferte. Denn durch ihre Platzierung wurde der Standardzellenbereich sehr zerrissen, wodurch sich in späteren Designschritten, z.B. beim Routen, schneller Probleme ergeben können. Dadurch können dann auch die Leitungslängen größer werden, womit die Signallaufzeit zwischen den Gattern ebenfalls vergrößert wird. Ein weiterer Punkt ist das Verlegen der Power- und Groundleitungen, welches hierdurch ebenso erschwert wird. Aus diesen Gründen ist dem automatischen Platzieren ein manuelles Platzieren der

Memoryblöcke vorzuziehen. Wie schon gesagt, kann dies durch Verschieben mit der Maus geschehen. Das bringt den Vorteil leicht und recht schnell eine gute Lösung zu finden.

- ***Silicon Ensemble:*** **linke Maustaste -> Cell Sl aktivieren**
- ***Silicon Ensemble:*** **mittlere Maustaste -> auf Move klicken**
- ***Move:*** **hier kann die Zelle beliebig gedreht werden**
- ***Silicon Ensemble:*** **linke Maustaste -> auf die zu verschiebende Zelle klicken und verschieben**

In dieser Weise können die Zellen verschoben werden. Dabei ist darauf zu achten, dass die Pins der Zellen leicht zugänglich bleiben, d.h. die nächste Zelle nicht direkt davor platziert wird. Weiterhin sollten die Zellen möglichst so platziert werden, dass sich kurze Leitungslängen ergeben, also nicht dass zwei Memoryblöcke, die direkt miteinander verbunden sind, in der oberen rechten und unteren linken Ecke liegen. Weiterhin sollten die Signalpins in Richtung des Coregebietes zeigen und die Power - und Groundpins der Blöcke gut erreichbar sein, da gerade hier breite Leitungen gelegt werden müssen. Günstig ist es dabei, wenn die Versorgungspins so liegen, dass sie direkt mit dem äußeren Power- und Groundring verbunden werden können. Außerdem sollte der Standardzellenbereich möglichst kompakt und zusammenliegend sein und durch die Platzierung möglichst wenig Ecken ins Coregebiet gebracht werden. Um all diese Dinge zu berücksichtigen, kann es sein, die Chipgröße auch wieder verändern zu müssen. Zum Beispiel ist dann ein quadratischer Chip nicht so gut geeignet sondern ein etwas länglicher Chip. Das hängt natürlich von der Größe und Anzahl der Memoryblöcke ab, in wieweit eventuell die Chipgröße angepasst werden muss. Wenn man eine gute Lösung für die Platzierung gefunden hat, können die Koordinaten und Ausrichtungen für die Zellen entnommen werden und in eine MAC-Datei eingegeben werden.

```
##-- MemoryBlöcke platzieren
MOVE CELL mcore0/proc0/rf0/u0/u1  FE  at  (-734430 532325) ;
MOVE CELL mcore0/proc0/rf0/u0/u0  E  at  (-241631 532560) ;
MOVE CELL mcore0/proc0/cmem0/idata0_0/u0/id0  FE  at  (-734622 -732591) ;
MOVE CELL mcore0/proc0/cmem0/ddata0_0/u0/id0  S  at  (337917 -704418) ;
MOVE CELL mcore0/proc0/cmem0/itags0_0/u0/id0  E  at  (55670 -732480) ;
MOVE CELL mcore0/proc0/cmem0/dtags0_0/u0/id0  E  at  (55670 -418320) ;
MOVE CELL mcore0/audio0/rec_fifo/RF256X32M1_1  N  at  (-529008 -311376) ;
MOVE CELL mcore0/audio0/play_fifo/RF256X32M1_1  N  at  (-734580 -311376) ;

##-- Rows unter den Memoryblöcken und 22 µm drumherum entfernen
CUT ROW BLOCKHALO 22000 ;
```

Mit dem letzten Kommando werden die Rows unter allen Memoryblöcken und 22 µm darum herum entfernt, da unter den Blöcken keine Zellen platziert werden

können und um die Blöcke herum noch Platz für die Verdrahtung gelassen werden muss.

Damit sind wir schon beim nächsten Punkt dem Verlegen der Powerverdrahtung. Dabei wird zuerst der äußere Ring gelegt und dann ist es durch die Memoryblöcke notwendig, um das Standardzellengebiet auch einen Ring zu legen, damit später das Followpinrouting durchgeführt werden kann. Dazu werden von Silicon Ensemble zuerst Channels erzeugt, diese laufen um die Corefläche und alle Memoryblöcke. Das sind Vorschläge von Silicon Ensemble für die Verlegung der Power- und Groundleitungen. Bei dem Design war es notwendig, einige Channels wieder zu entfernen. Anschließend werden dann die Ringe verlegt. Dazu wird angegeben, in welchem Layer mit welcher Breite und evtl. in welchem Abstand zum Channel die Ringe verlegt werden sollen. Wenn das getan ist, wird der Channel wieder geschlossen.

```
##-- Channels erzeugen
SET VAR DRAW.CHANNEL.AT ON ;
BUILD CHANNEL ;

##-- Power Channels löschen
DELETE CHANNEL ( -629251 526369 );
DELETE CHANNEL ( -473979 529898 );
DELETE CHANNEL ( -530441 445204 );
DELETE CHANNEL ( 344730 -228819 );
DELETE CHANNEL ( 291796 -419380 );
DELETE CHANNEL ( 341201 -493488 );
DELETE CHANNEL ( 35949 -546421 );
DELETE CHANNEL ( 35949 -378798 );

##-- äußeren Powerring erzeugen
CONSTRUCT RING NET "gnd!" NET "vdd!" LAYER MET3 CORERINGWIDTH 30000 SPACING
CENTER BLOCKRINGWIDTH 0 LAYER MET2 CORERINGWIDTH 30000 SPACING CENTER
BLOCKRINGWIDTH 0  ;
##-- Powerring zwischen den RAM-Blöcken erzeugen
CONSTRUCT RING NET "gnd!" NET "vdd!" LAYER MET3 CORERINGWIDTH 0 SPACING
CENTER BLOCKRINGWIDTH 8000 LAYER MET2 CORERINGWIDTH 0 SPACING CENTER
BLOCKRINGWIDTH 8000  ;

##-- Channels wieder schließen
DISPOSE CHANNEL ;
SET VAR DRAW.CHANNEL.AT OFF ;
```

Eventuell ist es auch nötig, zusätzlich an anderen Stellen noch Powerleitungen zu ziehen, was sich wie folgt realisieren lässt.

```
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET3
WIDTH 8000 ADDLAYER  MET2 WIDTH  MET2 8000 (-311926,517453) (-
248686,535806);
```

Damit lassen sich zum Beispiel auch die Initialen aller Mitwirkenden in den Chip integrieren. Wobei dann zusätzlich über das Gebiet mit den Initialen noch eine

Blockage gelegt werden muss, damit keine anderen Signalleitungen über das Gebiet gelegt werden und damit die Schrift nicht mehr lesbar ist.

```
ADD BLOCKAGE  Layer MET1  Layer MET2  Layer MET3  Layer MET4  Layer MET5
Layer MET6  (338774 -732435) (736054 -705695);
```

Abschließend werden jetzt die Verbindungen der Pads und der Memoryblöcke zu den Versorgungsleitungen erzeugt und das Followpinrouting durchgeführt.

```
CONNECT RING NET "gnd!" NET "vdd!" IOPAD ALLPORT ;
CONNECT RING NET "gnd!" NET "vdd!" BLOCK ;
CONNECT RING NET "gnd!" NET "vdd!" FOLLOWPIN ;
```

4.3 Placement

Hierbei werden die einzelnen Standardzellen in das Rowgebiet platziert. Hier wurde nur ein Standardplacement durchgeführt. Eine andere Möglichkeit besteht im TimingdrivenPlacement, wobei auf die von der Signallaufzeit kritischen Pfade besondere Rücksicht genommen wird. Weiterhin ist es möglich, während der Platzierung die Zellen in Hinsicht auf ihre Treiberstärke zu optimieren. Dabei werden zuerst die Treiberstärken aller Zellen auf das Minimum reduziert und dann in den kritischen Pfaden die Treiberstärken der Zellen erhöht, so dass für das Timing später ein besserer Wert als vorher entsteht. Diese Optimierungen konnten leider nicht ausprobiert werden, da dafür keine Lizenzen vorhanden waren. Weiterhin kann beim Platzieren vorab eine Timing Analyse durchgeführt werden.

```
##-- Timingdriven Placement
SET VAR QPLACE.TIMING.MODE "TRUE";
SET VAR TIMING.REPORTTIMING.LOGFILENAME "";
SET VAR QPLACE.OPT.REPORT.NAME "leon_audio_loaded_clockgcf_opt.qpopt.rpt";
SET VAR QPWR.RSPF "";
SET VAR QPLACE.PLACE.GROUTE.ANALYSIS "";
SET VAR QPLACE.OPT.TIMING.TYPE "";
SET VAR QPLACE.PLACE.PIN "";
QPLACE NOCONFIG;
```

4.4 Clock Tree Synthese

Hier besteht das Ziel, das Clocknetz des Chips so zu routen und zu buffern, dass das Clocksignal mit einem möglichst kleinen Skew an den FlipFlops ankommt. Dazu werden zwei Dateien benötigt, ctgen.cmd zum Steuern von CTGEN und ctgen.constraints für das Angeben der Bedingungen für den Clocktree. Diese sehen wie folgt aus:

```
##-- ctgen.constraints
specify_tree
        root_pin 'U5' 'DI'
set_constraints  # t in[ns]
```

```
##--    waveform        rise_time   high_time   fall_time   low_time
       waveform        0.1         20          0.1         20
       min_delay       0.2
       max_delay       0.5
       max_skew        0.1
```

```
##-- ctgen.cmd
set_rundir  '/home/fm22/projekt/leon/cadence_se/ctgen'
set_format se
read_technology
      lef_files
            '/opt/des_kits/UMC/0.18/UMCL18U300D2_1.3/silicon_ensemble/header_6lm_5.3.lef'
            '/opt/des_kits/UMC/0.18/UMCL18U300D2_1.3/silicon_ensemble/umcl18u300t2_6lm.lef'
            '/opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/silicon_ensemble/umcl18u250t2.lef'
            '/home/fm22/projekt/leon/SPR/R1024X32M4.lef'
            '/home/fm22/projekt/leon/SPR/R256X24M4.lef'
            '/home/fm22/projekt/leon/TPR/RF136X32M1.lef'
            '/home/fm22/projekt/leon/TPR/RF256X32M1.lef'

      gcf_env_file
            '/home/fm22/projekt/leon/cadence_se/gcf/umc_pad_core_ram_clocktiming.gcf'

read_design
      def_file
            '/home/fm22/projekt/leon/cadence_se/def/leon_audio_preclk.def'

read_constraints
      constraints_file
            '/home/fm22/projekt/leon/cadence_se/ctgen/ctgen.constraints'

build_clocktrees
remove_overlaps
write_design
      def_file
            '/home/fm22/projekt/leon/cadence_se/def/leon_audio_postclk.def'
```

Um die ClockTreeSynthese durchzuführen, muss vorerst noch eine DEF-Datei exportiert werden, welche von CTGEN benötigt wird.

```
##-- DEF-File exportieren
OUTPUT DEF FILENAME  "../def/leon_audio_preclk.def" ;
```

```
##-- CTS
CTGEN FILENAME ../ctgen/ctgen.cmd ;

##-- DEF-File importieren
SET VAR ALLOW.EC TRUE ;
INPUT DEF ECO FILENAME "../def/leon_audio_postclk.def" REPORTFILE
"leon_loaded.defeco.rpt" KEEPDISTCELLS KEEPDISTNETS NOKEEPSYNTPINS
NOKEEPEEQMODELS NOKEEPLEQMODELS NOKEEPORIGPLACE KEEPTIMING
NOKEEPCONSTRAINTS ;
```

Da jetzt alle Zellen platziert sind und daran auch nichts mehr geändert wird, können die Zwischenräume der Standardzellen mit Fillerzellen aufgefüllt werden. Dabei

werden zuerst die größtmöglichen Fillerzellen und dann immer kleiner werdende Fillerzellen in das Coregebiet eingefügt.

```
SROUTE ADDCELL MODEL FILL16 PREFIX CFIL16 NO FN SO FS AREA ( -1075800 -
1075760 ) ( 1077120 1076880 ) ;
SROUTE ADDCELL MODEL FILL8 PREFIX CFIL8 NO FN SO FS AREA ( -1075800 -
1075760 ) ( 1077120 1076880 ) ;
SROUTE ADDCELL MODEL FILL4 PREFIX CFIL4 NO FN SO FS AREA ( -1075800 -
1075760 ) ( 1077120 1076880 ) ;
SROUTE ADDCELL MODEL FILL2 PREFIX CFIL2 NO FN SO FS AREA ( -1075800 -
1075760 ) ( 1077120 1076880 ) ;
SROUTE ADDCELL MODEL FILL1 PREFIX CFIL1 NO FN SO FS AREA ( -1075800 -
1075760 ) ( 1077120 1076880 ) ;
```

4.5 Routing

Jetzt sind alle Zellen platziert und das Routing des Leon kann durchgeführt werden. Es wird dabei in zwei getrennten Durchgängen vollführt. Als erstes wird das Clocknetz und abschließend werden die Signalleitungen geroutet.

```
##-- Clock Routing
CLOCKROUTE ALL ;

##-- Signal Routing
SET VAR WROUTE.FINAL TRUE ;
SET VAR WROUTE.GLOBAL TRUE ;
SET VAR WROUTE.INCREMENTAL.FINAL FALSE ;
WROUTE NOCONFIG ;
```

Beim Routen der Signalleitungen sind nach dem erstem Durchlauf etwa 1100 Fehler aufgetreten, die aber durch nachfolgende Routingdurchläufe beseitigt werden konnten. Das zeigt, dass ab einer bestimmten Auslastung der Rows, das Routing nicht mehr fehlerfrei zu durchlaufen ist. Abhilfe könnte ein neuer Floorplan bringen, insbesondere die Anordnung der Memoryblöcke. Das wird nur eine von vielen Möglichkeiten sein, wenn gar nichts klappt, bleibt immer noch die Möglichkeit den Chip zu vergrößert, was aber gleich mit höheren Produktionskosten zu Buche schlägt und aus diesem Grunde erst zuletzt in Betracht gezogen werden sollte.

4.6 Timing Analyse

Nach dem Routen des Designs ist ein vollständiges Layout entstanden. Abschließend muss das Design verifiziert werden, wozu auch die Timing Analyse gehört. Dabei handelt es sich aber um keine Backannotation. Vielmehr werden hier nur die zeitkritischen Pfade im Design aufgezeigt, um eventuell gleich noch Optimierungen durchzuführen, damit ein gewünschtes Timing erreicht wird. Um eine Timing Analyse durchzuführen, ist es notwendig, ein künstliches Clocksignal zu erzeugen. Das wurde in Abschnitt 4.1 durch die GCF-Datei getan. Weiterhin ist eine SDF-Datei

notwendig. SDF steht dabei für „Standard Delay Format“ und enthält die genauen Verzögerungen für jede einzelne Signalleitung. Diese Datei wird dann auch für eine Backannotation benötigt. Um sie zu erzeugen muss erst eine RSPF-Datei erzeugt werden.

```
##-- RSPF Generierung
REPORT RC FILENAME ../sdf/leon_audio_opt.rspf ;

##-- SDF Generierung
SET VAR TIMING.RSPF.FILE "../sdf/leon_audio_opt.rspf";
REPORT DELAY SDFOUTPUT FILENAME ../sdf/leon_audio_opt.sdf USERSPF ;
```

Zusätzlich wird noch eine DEF-Datei des komplett gerouteten Designs benötigt.

```
##-- DEF-File exportieren
OUTPUT DEF FILENAME  "../def/leon_audio_routed.def" ;
```

Damit kann jetzt die eigentliche Timing Analyse durchgeführt werden. Leider funktionierte die Timinganalyse innerhalb von Silicon Ensemble nicht. Es muss für eine Timinganalyse Pearl gestartet werden. Dies geschieht mit dem Aufruf

```
pearl
```

in dem `work`-Verzeichnis. Anschließend werden folgende Befehle an Pearl gegeben, um eine Timinganalyse durchzuführen.

```
Logfile pearl.tmplog
ReadGCFTimingLibraries
/home/fm22/projekt/leon/cadence_se/gcf/umc_pad_core_ram_clocktiming.gcf
ReadDefNetlist
/home/fm22/projekt/leon/cadence_se/work/leon_audio_routed.def
ReadSPF /home/fm22/projekt/leon/cadence_se/work/leon_audio_routed.rspf
SetDelayPathFormat out_delay delta_delay out_node device pin_to_pin
IdealClocks yes
ReadGCFConstraints
/home/fm22/projekt/leon/cadence_se/gcf/umc_pad_core_ram_clocktiming.gcf
SetMaxPossibilities 10
CheckTiming
CheckLimits -check max_load,max_slew > checklimits.log
TimingVerify  -check setup,hold -max_slack 0  > leon_audio_routed.path
ShowPossibility 1 500 >> leon_audio_routed.path
Quit
```

Damit werden im **work**-Verzeichnis die Dateien **checklimits.log** und **leon_audio_routed.path** erstellt. Sie erhalten die gewünschten Timinginformationen des Designs. Die **checklimits.log** enthält unter anderem die Slew- und Loadviolations der Standardzellen im Design. An dieser Stelle zeigte sich, dass es wichtig ist, schon bei der Synthese Optimierungen durchzuführen. Denn bei unserem Design ergaben sich an dieser Stelle Slewviolations bei Gattern mit einem Fanout von über 30, weil ihre Treiberstärke vom Synthesetool nicht optimal angepasst wurde. Durch manuelles Erhöhen der Treiberstärken der Gatter konnten diese Violations beseitigt werden. Wenn eine Lizenz für das Optimieren der Gatter in Silicon Ensemble verfügbar ist, kann die fehlende Optimierung im Synthesetool an dieser Stelle wieder ausgebessert werden. Dieser Schritt wäre auch sinnvoll, wenn schon eine Optimierung bei der Synthese durchgeführt wurde, denn bei der Synthese sind keinerlei Layoutinformationen des Designs verfügbar. Im Layout sind Informationen über die Leitungslängen vorhanden und damit können die Treiberstärken der Gatter angepasst werden. In der zweiten Datei **leon_audio_routed.path** sind die Pfade des Designs enthalten, die eine Setup- oder Holdviolation hervorrufen. Bei einer Setupviolation besitzt der Datenpfad von Flipflop durch die Logik zum nächsten Flipflop eine größere Signallaufzeit als die Clockperiode minus der Setupzeit, wodurch an dem ankommenden Flipflop die Daten nicht übernommen werden können. Hierbei ist eine Betrachtung mit den Worstcasemodellen der TLF-Dateien notwendig, weil die Signallaufzeiten im Worstcase größer sind. Bei einer Holdviolation hingegen sind die Daten des sendenden Flipflops noch innerhalb der vorgeschriebenen Holdtime des empfangenden Flipflops am empfangenden Flipflop angekommen, so dass dieses in einen unsicheren Zustand gerät und ein Fehler auftritt. Da die Signallaufzeiten im Bestcase geringer sind, ist bei dieser Fehleranalyse die Betrachtung im Bestcase nötig. Aus diesem Grund ist es wichtig für die Timinganalyse einmal die TLF-Dateien für den Best- und einmal für den Worstcase einzulesen. Wenn das getan ist und beide Dateien keinen Fehler mehr enthalten, ist das Timing der Schaltung sichergestellt und die Timinganalyse abgeschlossen.

4.7 Automatisierter Designflow

In den vorherigen Abschnitten wurde die Steuerung von Silicon Ensemble durch die MAC-Dateien erläutert. Um diesen Designflow übersichtlich zu halten, kann für jeden Abschnitt im Designflow auch eine MAC-Datei angefertigt werden. Die komplette Ausführung aller MAC-Dateien kann dann wiederum durch eine MAC-Datei übernommen werden. Das bringt gerade bei der Erstellung der MAC-Dateien den Vorteil, dass nicht immer ein sehr großer Teil auskommentiert werden muss, wenn man nur ein oder zwei Befehle ausprobieren möchte. Hier sei abschließend einmal die MAC-Datei für den Aufruf der einzelnen MAC-Dateien dargestellt.

```
##-- Library laden
EXECUTE "../mac/library_leon_audio_clockgcf.mac" ;

##-- Floorplanning
```

```
EXECUTE "../mac/floor_leon_audio_clockgcf_names.mac" ;

##-- Powerplanning
EXECUTE "../mac/power_leon_audio_names.mac" ;

##-- Placement
EXECUTE "../mac/place_leon_audio_clockgcf.mac" ;

##-- Clock Tree Synthese
EXECUTE "../mac/cts_leon_audio.mac" ;

##-- Fillerzellen
EXECUTE "../mac/fillcore_leon_audio.mac" ;

##-- Routing
EXECUTE "../mac/route_leon_audio_clockgcf.mac" ;

##-- SDF Generierung
EXECUTE "../mac/sdf_leon_audio.mac" ;
```

5 Anhang

Hier sind alle MAC-Dateien für den Designflow des Leonprozessors mit dem Audiocore dargestellt.

```
##-------------------------------------------------------------
##--
##--  Silicon Ensemble Macro File make_leon_audio_clockgcf.mac
##--
##-------------------------------------------------------------

##-- Library laden
EXECUTE "../mac/library_leon_audio_clockgcf.mac" ;

##-- Floorplanning
EXECUTE "../mac/floor_leon_audio_clockgcf_names.mac" ;

##-- Powerplanning
EXECUTE "../mac/power_leon_audio_names.mac" ;

##-- Placement
EXECUTE "../mac/place_leon_audio_clockgcf.mac" ;

##-- Clock Tree Synthese
EXECUTE "../mac/cts_leon_audio.mac" ;

##-- Fillerzellen im Coregebiet
EXECUTE "../mac/fillcore_leon_audio.mac" ;

##-- Routing
EXECUTE "../mac/route_leon_audio_clockgcf.mac" ;

##-- SDF Generierung
EXECUTE "../mac/sdf_leon_audio.mac" ;

##-- Timing Analyse
EXECUTE "../mac/timing_leon_audio.mac" ;
```

```
##------------------------------------------------------------
##--
##--  Silicon Ensemble Macro File library_leon_audio_clockgcf.mac
##--
##------------------------------------------------------------

##-- Import Library Data

##-- LEF
##-- FINPUT (Clear Existing Design Data) löscht alle vorherigen Informationen
FINPUT LEF FILENAME
/opt/des_kits/UMC/0.18/UMCL18U300D2_1.3/silicon_ensemble/header_6lm_5.3.lef ;
INPUT LEF FILENAME
/opt/des_kits/UMC/0.18/UMCL18U300D2_1.3/silicon_ensemble/umcl18u300t2_6lm.lef ;
INPUT LEF FILENAME
/opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/silicon_ensemble/umcl18u250t2.lef ;
INPUT LEF FILENAME /home/fm22/projekt/leon/SPR/R1024X32M4.lef ;
INPUT LEF FILENAME /home/fm22/projekt/leon/SPR/R256X24M4.lef ;
INPUT LEF FILENAME /home/fm22/projekt/leon/TPR/RF136X32M1.lef ;
INPUT LEF FILENAME /home/fm22/projekt/leon/TPR/RF256X32M1.lef ;

##-- CTLF Timing
##-- GCF File
INPUT CTLF INITFILE
"/home/fm22/projekt/leon/cadence_se/gcf/umc_pad_core_ram_clocktiming.gcf" ;
##-- GCF Constraints für Clocksignal
INPUT GCF FILENAME "../gcf/umc_pad_core_ram_clocktiming.gcf" REPORTFILE
"importgcf.rpt" ;

##-- Import Design Data
##-- Verilog
## INPUT VERILOG FILE
"/opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/silicon_ensemble/umcl18u250t2_floorplan.v"
LIB "cds_vbin" REFLIB "cds_vbin " ;
INPUT VERILOG FILE "/home/fm22/projekt/leon/SPR/R1024X32M4.v" LIB "cds_vbin" REFLIB
"cds_vbin " ;
INPUT VERILOG FILE "/home/fm22/projekt/leon/SPR/R256X24M4.v" LIB "cds_vbin" REFLIB
"cds_vbin " ;
INPUT VERILOG FILE "/home/fm22/projekt/leon/TPR/RF136X32M1.v" LIB "cds_vbin" REFLIB
"cds_vbin " ;
INPUT VERILOG FILE "/home/fm22/projekt/leon/TPR/RF256X32M1.v" LIB "cds_vbin" REFLIB
"cds_vbin " ;
INPUT VERILOG FILE "../netlist/leon_audio.v" LIB "cds_vbin" REFLIB "cds_vbin "
DESIGN  "cds_vbin.leon_audio:hdl" ;

##-- Save design
SAVE "leon_audio_loaded_clockgcf" ;

##-- Design laden
#FLOAD DESIGN "/home/fm22/projekt/leon/cadence_se/work/dbs/leon_audio_loaded" ;
```

```
##-------------------------------------------------------
##--
##--  Silicon Ensemble Macro File floor_leon_audio.mac
##--
##-------------------------------------------------------

##-- Chipgröße und Row Utilization können ebenfalls durch das Def-File
vorgegeben werden
#FINIT FLOORPLAN  rowu 0.85 rowsp 0 blockhalo 2000 f x 2152000 y 2152000
abut xio 65000 yio 65000 ;

##-- DEF-File exportieren um die Padpositionen anzupassen
##-- es müssen nur die Cells- und Specialnetsinformationen veraendert
werden
#OUTPUT DEF  CELLS  SPECIALNETS  FILENAME  "../def/leon_audio_prepads.def"
;

##-- Pads mit FillerCells
FINPUT DEF FILENAME "../def/leon_audio_postpads_clockgcf.def" ;

##-- Place RAM-Blöcke (automatisch)
#QPLACE NOCONFIG BLOCK ;
##-- wieder entfernen
#CHANGE CELL * UNPLACECORECELL ;

##-- Place RAM-Blöcke (manuell)
MOVE CELL mcore0/proc0/rf0/u0/u1  FE  at  (-734430 532325) ;
MOVE CELL mcore0/proc0/rf0/u0/u0  E  at  (-241631 532560) ;
MOVE CELL mcore0/proc0/cmem0/idata0_0/u0/id0  FE  at  (-734622 -732591) ;
MOVE CELL mcore0/proc0/cmem0/ddata0_0/u0/id0  S  at  (337917 -704418) ;
MOVE CELL mcore0/proc0/cmem0/itags0_0/u0/id0  E  at  (55670 -732480) ;
MOVE CELL mcore0/proc0/cmem0/dtags0_0/u0/id0  E  at  (55670 -418320) ;
MOVE CELL mcore0/audio0/rec_fifo/RF256X32M1_1  N  at  (-529008 -311376) ;
MOVE CELL mcore0/audio0/play_fifo/RF256X32M1_1  N  at  (-734580 -311376) ;

##-- Rows unter den RAMs und ?? nm drumrum wegschneiden
CUT ROW BLOCKHALO 22000 ;
```

```
##-------------------------------------------------------
##--
##--  Silicon Ensemble Macro File power_leon_audio_names.mac
##--
##-------------------------------------------------------

##-- Alle gerouteten Specialnets löschen
DELETE WIRE ALLNETS NOKEEPGLOBAL NODRC SP;

##-- Channel oeffnen das entspricht Route->Plan Power
SET VAR DRAW.CHANNEL.AT ON ;
BUILD CHANNEL ;

##-- Power Channels löschen
##-- um nicht zwischen allen RAM-Blöcken Powerleitungen zu legen
DELETE CHANNEL ( -629251 526369 );
DELETE CHANNEL ( -473979 529898 );
DELETE CHANNEL ( -530441 445204 );
DELETE CHANNEL ( 344730 -228819 );
DELETE CHANNEL ( 291796 -419380 );
DELETE CHANNEL ( 341201 -493488 );
DELETE CHANNEL ( 35949 -546421 );
DELETE CHANNEL ( 35949 -378798 );

##-- aeußeren Powerring erzeugen
CONSTRUCT RING NET "gnd!" NET "vdd!" LAYER MET3 CORERINGWIDTH 30000 SPACING
CENTER BLOCKRINGWIDTH 0 LAYER MET2 CORERINGWIDTH 30000 SPACING CENTER
BLOCKRINGWIDTH 0  ;

##-- Powerring zwischen den RAM-Blöcken erzeugen
CONSTRUCT RING NET "gnd!" NET "vdd!" LAYER MET3 CORERINGWIDTH 0 SPACING
CENTER BLOCKRINGWIDTH 8000 LAYER MET2 CORERINGWIDTH 0 SPACING CENTER
BLOCKRINGWIDTH 8000  ;

##-- Channel wieder schließen das entspricht dem Schließen des Plan Power
DISPOSE CHANNEL ;
SET VAR DRAW.CHANNEL.AT OFF ;

##-- Powerleitung manuell ziehen, die nicht automatisch gezogen wurden
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET3
WIDTH 8000 ADDLAYER  MET2 WIDTH  MET2 8000 (-311926,517453) (-
248686,535806);

ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET3
WIDTH 8000 ADDLAYER  MET2 WIDTH  MET2 8000 (-248569,517687)
(243828,525921);

ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET gnd! LAYER
MET3  WIDTH 8000 ADDLAYER  MET2 WIDTH  MET2 8000 (40017,-310154) (40551,-
762155);

ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET3  WIDTH 8000 ADDLAYER  MET2 WIDTH  MET2 8000 (48656,-311041) (49162,-
793037);


##-- Initialien einfügen
EXECUTE "../mac/names_leon_audio.mac" ;
ADD BLOCKAGE  Layer MET1  Layer MET2  Layer MET3  Layer MET4  Layer MET5
Layer MET6  (338774 -732435) (736054 -705695);
```

```
##-- Power- und Groundpads mit dem aeußeren Ring verbinden.
CONNECT RING NET "gnd!" NET "vdd!" IOPAD ALLPORT ;

##-- Power- und Groundpins aller RAM-Blöcke mit dem Powerring verbinden.
#CONNECT RING NET "gnd!" NET "vdd!" BLOCK ALLPORT ;
CONNECT RING NET "gnd!" NET "vdd!" BLOCK ;
##-- Power- und Groundpins eines RAM-Blöcke mit dem Powerring verbinden.
##--CONNECT RING NET "gnd!" NET "vdd!" BLOCK NAME "mcore0/proc0/rf0/u0/u0"
;

##-- Followpinrouting
CONNECT RING NET "gnd!" NET "vdd!" FOLLOWPIN ;









##------------------------------------------------------------
##--
##--  Silicon Ensemble Macro File names_leon_audio.mac
##--
##------------------------------------------------------------

##-- Alle gerouteten Specialnets löschen
#DELETE WIRE ALLNETS NOKEEPGLOBAL NODRC SP;

##-- Initialien

SET VAR DRAW.INFO.GEOMETRY.AT "OFF";

##-- C
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (353015,-709066) (341018,-709066) (341018,-727062) (353015,-
727062);
##-- J
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (354229,-709018) (365641,-709018) (365641,-727013) (357520,-
727013) (357870,-723442);
##-- C
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (383015,-709066) (371018,-709066) (371018,-727062) (383015,-
727062);

##-- M
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (393653,-728062) (393653,-709066) (408649,-709066)
(408649,-728062);
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (401309,-708987) (401309,-728062) ;
##-- S
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (413200,-727062) (424196,-727062) (424196,-717985) (414620,-
717985) (414620,-708987) (424881,-708987) ;
##-- F
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (427369,-709066) (439444,-709066);
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (432341,-708987) (432341,-728062);
```

```
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (429184,-717827) (435735,-717827);

##-- F
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (448369,-709066) (460444,-709066);
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (453341,-708987) (453341,-728062);
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (450184,-717827) (456735,-717827);
##-- G
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (476034,-709096) (464085,-709096) (464085,-727088) (475003,-
727088) (474003,-719045) (470120,-721045) ;

##-- A
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (487403,-728026) (487403,-709083) (497436,-709083)
(497436,-728026) ;
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (487510,-716711) (497114,-716711);
##-- H
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (503119,-728026) (503119,-707975) ;
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (502012,-716819) (513186,-716819) ;
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (512648,-728026) (512648,-707975) ;

##-- R
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (524047,-728031) (524047,-709059) (534852,-708952)
(533996,-716226) (524261,-715905) (524047,-719221) (526829,-719649)
(526936,-721896) (529610,-721789) (529824,-724891) (532391,-724784)
(532177,-727854);
##-- H
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (540119,-728026) (540119,-707975) ;
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (540012,-716819) (550186,-716819) ;
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (550648,-728026) (550648,-707975) ;

##-- F
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (561369,-709066) (573444,-709066);
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (566341,-708987) (566341,-728062);
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (563184,-717827) (569735,-717827);
##-- P
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (576614,-728006) (576614,-709047) (587254,-709322)
(587254,-717290) (576614,-717290) ;

##-- F
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (596369,-709066) (608444,-709066);
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (601341,-708987) (601341,-728062);
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (598184,-717827) (604735,-717827);
##-- S
```

```
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (611200,-727062) (623196,-727062) (623196,-717985) (612620,-
717985) (612620,-708987) (623881,-708987) ;

##-- S
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (632200,-727062) (644196,-727062) (644196,-717985) (633620,-
717985) (633620,-708987) (644881,-708987) ;
##-- S
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (648200,-727062) (660196,-727062) (660196,-717985) (649620,-
717985) (649620,-708987) (660881,-708987) ;

##-- D
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (672428,-728046) (672242,-709124) (680271,-709124)
(680271,-711260) (682221,-711260) (682035,-725332) (680085,-725332)
(680364,-727467) (672149,-727282);
##-- T
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (686821,-709032) (696942,-709032) ;
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (692299,-708846) (692114,-728046);

##-- G
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK NET vdd! LAYER MET1
WIDTH 3000 (716034,-709096) (704085,-709096) (704085,-727088) (715003,-
727088) (715003,-719045) (710120,-721045) ;
##-- W
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (720630,-707990) (720630,-727075) (732863,-727075)
(732863,-707990) ;
ADD WIRE SPECIAL DRC NOVIASATCROSSOVER SHORTSCHECK YFIRSTL NET vdd! LAYER
MET1 WIDTH 3000 (726746,-726615) (727010,-712344) ;
```

```
##-------------------------------------------------------
##--
##--  Silicon Ensemble Macro File place_leon_audio.mac
##--
##-------------------------------------------------------

##-- Placement der Zellen
SET VAR TIMING.REPORTTIMING.LOGFILENAME "" ;
SET VAR QPLACE.OPT.REPORT.NAME "leon_audio.qpopt.rpt" ;
SET VAR QPWR.RSPF "" ;
SET VAR QPLACE.PLACE.GROUTE.ANALYSIS "" ;
SET VAR QPLACE.OPT.TIMING.TYPE "" ;
SET VAR QPLACE.PLACE.PIN "" ;
QPLACE NOCONFIG ;

##-- Timingdriven Placement
#SET VAR QPLACE.TIMING.MODE "TRUE";
#SET VAR TIMING.REPORTTIMING.LOGFILENAME "";
#SET VAR QPLACE.OPT.REPORT.NAME "leon_audio_loaded_clockgcf_opt.qpopt.rpt";
#SET VAR QPWR.RSPF "";
#SET VAR QPLACE.PLACE.GROUTE.ANALYSIS "";
#SET VAR QPLACE.OPT.TIMING.TYPE "";
#SET VAR QPLACE.PLACE.PIN "";
#QPLACE NOCONFIG;

##-- DEF-File für CTS exportieren
OUTPUT DEF FILENAME  "../def/leon_audio_preclk.def" ;

##-- SAVE Design leon_placed
SAVE DESIGN "leon_audio_placed_clockgcf" ;
```

```
##------------------------------------------------------------
##--
##--  Silicon Ensemble Macro File cts_leon_audio.mac
##--
##------------------------------------------------------------

##-- CTS
CTGEN FILENAME ../ctgen/ctgen.cmd ;

##-- DEF-File von CTS wieder importieren
SET VAR ALLOW.EC TRUE ;
INPUT DEF ECO FILENAME "../def/leon_audio_postclk.def" REPORTFILE
"leon_loaded.defeco.rpt" KEEPDISTCELLS KEEPDISTNETS NOKEEPSYNTPINS
NOKEEPEEQMODELS NOKEEPLEQMODELS NOKEEPORIGPLACE KEEPTIMING
NOKEEPCONSTRAINTS ;



##------------------------------------------------------------
##-- ctgen.constraints für CTGEN
##------------------------------------------------------------

##-- geht vom ClkPad auf alle Clockeingänge
specify_tree
        root_pin 'U5' 'DI'
set_constraints  ##-- t in[ns]
##--  waveform rise_time high_time  fall_time  low_time
        waveform  0.1      20           0.1     20
        min_delay 0.2
        max_delay 0.5
        max_skew  0.1


##------------------------------------------------------------
##-- ctgen.cmd für CTGEN
##------------------------------------------------------------

set_rundir  '/home/fm22/projekt/leon/cadence_se/ctgen'
set_format se
read_technology
      lef_files
            '/opt/des_kits/UMC/0.18/UMCL18U300D2_1.3/silicon_ensemble/header_6lm_5.3.lef'
            '/opt/des_kits/UMC/0.18/UMCL18U300D2_1.3/silicon_ensemble/umcl18u300t2_6lm.lef'
            '/opt/des_kits/UMC/0.18/UMCL18U250D2_2.4/silicon_ensemble/umcl18u250t2.lef'
            '/home/fm22/projekt/leon/SPR/R1024X32M4.lef'
            '/home/fm22/projekt/leon/SPR/R256X24M4.lef'
            '/home/fm22/projekt/leon/TPR/RF136X32M1.lef'
            '/home/fm22/projekt/leon/TPR/RF256X32M1.lef'
      gcf_env_file
            '/home/fm22/projekt/leon/cadence_se/gcf/umc_pad_core_ram_clocktiming.gcf'

read_design
      def_file
            '/home/fm22/projekt/leon/cadence_se/def/leon_audio_preclk.def'

read_constraints
      constraints_file
            '/home/fm22/projekt/leon/cadence_se/ctgen/ctgen.constraints'

build_clocktrees
remove_overlaps
write_design
      def_file
            '/home/fm22/projekt/leon/cadence_se/def/leon_audio_postclk.def
```

```
##-------------------------------------------------------
##
## Silicon Ensemble Macro File fillcore_leon_audio.mac
##
## platzieren der Fillerzellen im Coregebiet
##
##-------------------------------------------------------

SROUTE ADDCELL MODEL FILL16 PREFIX CFIL16 NO FN SO FS AREA ( -1075800 -
1075760 ) ( 1077120 1076880 ) ;
SROUTE ADDCELL MODEL FILL8 PREFIX CFIL8 NO FN SO FS AREA ( -1075800 -
1075760 ) ( 1077120 1076880 ) ;
SROUTE ADDCELL MODEL FILL4 PREFIX CFIL4 NO FN SO FS AREA ( -1075800 -
1075760 ) ( 1077120 1076880 ) ;
SROUTE ADDCELL MODEL FILL2 PREFIX CFIL2 NO FN SO FS AREA ( -1075800 -
1075760 ) ( 1077120 1076880 ) ;
SROUTE ADDCELL MODEL FILL1 PREFIX CFIL1 NO FN SO FS AREA ( -1075800 -
1075760 ) ( 1077120 1076880 ) ;
```

```
##-------------------------------------------------------
##--
##--  Silicon Ensemble Macro File route_leon_audio.mac
##--
##-------------------------------------------------------

##-- Clock Routing
CLOCKROUTE ALL ;

##-- Signal Routing
SET VAR WROUTE.FINAL TRUE ;
SET VAR WROUTE.GLOBAL TRUE ;
SET VAR WROUTE.INCREMENTAL.FINAL FALSE ;
WROUTE NOCONFIG ;

##-- Blockage der Initialien löschen
DELETE BLOCKAGE  Layer MET1  Layer MET2  Layer MET3  Layer MET4  Layer MET5
Layer MET6  (338774 -732435) (736054 -705695);

##-- Save Design
SAVE DESIGN "leon_audio_routed_clockgcf" ;

##-- Connectivity Check
#VERIFY CONNECTIVITY ;

##-- Search and Repair
#FROUTE SEARCHANDREPAIR RUNTIMELIMIT 1000 TOTALTIMELIMIT 86400  XSBOXSIZE 8
YSBOXSIZE 7;
```

```
##-------------------------------------------------------
##--
##--  Silicon Ensemble Macro File sdf_leon_audio.mac
##--
##-------------------------------------------------------

##-- RSPF Generierung
REPORT RC FILENAME ../sdf/leon_audio_opt.rspf ;

##-- SDF Generierung
SET VAR TIMING.RSPF.FILE "../sdf/leon_audio_opt.rspf";
REPORT DELAY SDFOUTPUT FILENAME ../sdf/leon_audio_opt.sdf USERSPF ;




##-------------------------------------------------------
##--
##--  Silicon Ensemble Macro File timing_leon_audio.mac
##--
##--  klappte aus Silicon Ensemble heraus nicht
##--  mit pearl funktionierte die Timinganalyse
##--
##-------------------------------------------------------

SET V TIMING.REPORTTIMING.DELAYSOURCE "SDF" ;
SET V TIMING.REPORTTIMING.DELAYSRCFILE "../sdf/leon_audio_opt.sdf" ;
SET V TIMING.REPORTTIMING.CHECKTYPE 6147 ;
REPORT TIMING ;




##-------------------------------------------------------
##--
##--  Pearl Kommando Datei
##--
##-------------------------------------------------------

Logfile pearl.tmplog

ReadGCFTimingLibraries
/home/fm22/projekt/leon/cadence_se/gcf/umc_pad_core_ram_clocktiming.gcf

ReadDefNetlist
/home/fm22/projekt/leon/cadence_se/work/leon_audio_routed.def

ReadSPF /home/fm22/projekt/leon/cadence_se/work/leon_audio_routed.rspf

SetDelayPathFormat out_delay delta_delay out_node device pin_to_pin

IdealClocks yes

ReadGCFConstraints
/home/fm22/projekt/leon/cadence_se/gcf/umc_pad_core_ram_clocktiming.gcf

SetMaxPossibilities 10

CheckTiming

CheckLimits -check max_load,max_slew > checklimits.log

TimingVerify  -check setup,hold -max_slack 0  > leon_audio_routed.path

ShowPossibility 1 500 >> leon_audio_routed.path

Quit
```