Layoutsynthese

Werkzeuge : CADENCE IC, HSPICE
Design-Kits : ES 2 Full-Custom
designSetup : `ic2 hsp es2`

A. Mäder

Diese Beschreibung ist als Ergänzung zu „Full-Custom Design“ gedacht. Hier wird beschrieben, wie aus einer Transistornetzliste – als Schematic – mit Hilfe der Werkzeuge zur Layoutsynthese ein Full-Custom Layout erzeugt und simuliert wird wird. Die Eingaben des Benutzer sind dabei mit den schon vorher verwendeten Symbolen dargestellt.

Design-Flow

Der Entwurfsablauf ist, als Ergänzung zu dem in „Full-Custom Design“ beschriebenen Vorgehen, durch folgende Schritte charakterisiert:

1. Eingabe der Transistornetzliste mit dem Schematic-Editor
2. Generierung eines Symbols für den Schematic-Editor
• Aufbau einer Testumgebung
• Simulation der Schaltung — die in der Testumgebung simulierte Schaltung besteht dabei *nur* aus Transistoren!
3. Synthese (Erzeugung und Kompaktifizierung) eines symbolischen Layouts
4. Nachbearbeitung des symbolischen Layouts und Umwandlung in ein fertiges Layout
5. Kontrolle durch einen Design Rule Check
6. Extraktion der elektrischen Netzliste
7. Aufbau einer Testumgebung (sofern dies nicht schon vorher geschehen ist)
8. Simulation der Schaltung — in der Testumgebung wird die extrahierte Netzliste simuliert, bestehend aus Transistoren, Kapazitäten, Dioden...

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

Arbeitsschritte

Schematiceingabe

1. Start des Systems

$>$ `ES2FCStart` [xterm]

2. Bibliothek erzeugen
optional: Nur beim ersten Start ist eine eigene Bibliothek einzurichten.

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

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

≡ `Attach To...` = `techLib` [Attach Design...]

3. Design der Schaltung — siehe: „CADENCE Grundlagen", 4 Der Schematic-Editor
Werden neue Entwürfe erstellt:

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

≡ `Library Name`	= ⟨*libId*⟩		[Create New File]
`Cell Name`	= ⟨*cellId*⟩		
`Tool`	= `Composer-Schematic`		

Um bereits existierende Entwürfe zu bearbeiten:

☐ `File - Open...` [icfb - Log:...]

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

Die Schaltung wird aus P- und N-Kanal Transistoren aufgebaut. Deren Instanzen sind in der Bibliothek `sample` in der Zellkategorie `TRANS` zu finden.

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

≡ `Library`	= ⟨*libId*⟩	`sample`	[Add Component]
`Cell`	= ⟨*cellId*⟩	`pmos` | `nmos`	
`View`	= `symbol`		

Anmerkung: da die *bulk*-Anschlüsse der Transistoren ohnehin mit `vdd` (P-Transistor) und `gnd` (N-Transistor) verbunden wird, sollten `pmos` und `nmos` eingesetzt werden, **nicht:** `pfet`, `nfet`.

Die Werte für die Kanalweite und -länge werden über Properties angegeben.

☐ `Edit - Properties - Objects...`/⊙ `q` [Composer-Schematic...]

↑$_l$ `Add` [Edit Object Properties]

≡ `Name`	= `w`	`l`	[Add Property]
`Type`	= `float`		
`Value`	= ⟨*trWidth*⟩	`0.8`	

Nach Abschluss des Befehls werden die Properties an den instantiierten Transistoren als ⟨*trWidth*⟩/⟨*trLength*⟩ angezeigt.

Achtung: Um dem späteren Layout zu entsprechen, dürfen keine Instanzen von `vdd` und `gnd` für die Anschlüsse der Versorgungsspannung verwendet werden. Diese werden erst später (in der Testumgebung) hinzugefügt.

Anschließend können die Ein- und Ausgänge des Schematic mit Pins versehen werden, über die die Schaltung in einer Hierarchie angeschlossen wird.

☐ `Add - Pin...`/⊙ `p` [Composer-Schematic...]

≡ `Pin Names`	= ⟨*pinIdLis*⟩	`vdd` | `gnd`	[Add Pin]
`Direction`	= `input` | `output`	`inout`	
`Usage`	= `schematic`		

Anmerkung: prinzipiell kann auch ein hierarchisches Design aus den Transistoren aufgebaut werden.[1] Es ist immer darauf zu achten, dass die Spannungsversorgung *erst* in der Testumgebung angeschlossen wird.

[1] Bei „kleinen" Schaltungen (Übungen, Praktika) ist von einer Hierarchie allerdings abzuraten.

4. Symbol generieren
Wurde das Schematic (die Hierarchie) fertig entworfen, so kann das Symbol als Schnittstelle zur Testumgebung erzeugt werden.

□ `Design - Create Cellview - From Cellview...` [Composer-Schematic...]

≡ `Library Name` = ⟨*libId*⟩ [Cellview From Cellview]
`Cell Name` = ⟨*cellId*⟩
`Display Cellview` = `off`
`Edit Options` = `off` | `on`
`From View Name` = `schematic`
`To View Name` = `symbol`

Anschließend kann gesichert werden — dabei sollte auch gleich ein Schematic Rule Check durchgeführt werden.

□ `Design - Check and Save`/⊙ `X` [Composer-Schematic...]

Simulation der Schematic-Netzliste

5. Design der Testumgebung und Simulation der Schematic-Netzliste
optional: Nur wenn die Transistornetzliste der `schematic`-View simuliert werden soll. Die Simulation wird aber *in jedem Fall* später mit den extrahierten Layoutdaten durchgeführt.

Das Vorgehen entspricht genau demjenigen in „Full-Custom Design“:

- 11. Design der Testumgebung
- 12. Simulation vorbereiten
- 13. Stimuli editieren
- 14. Simulation starten
- 15. Signale für Waveforms markieren
  Sollen interne Signale der Schaltung für die Simulation markiert werden, so findet der dort beschriebene Abstieg in der Hierarchie natürlich zu einer `schematic`-View statt. Dementsprechend gelten andere Befehle:
  □ `Design - Probe - Add Net`/⊙ `9` [Composer-Schematic...]
  für die Auswahl von Signalen zur Anzeige.
  □ `Design - Hierarchy - Return`/⊙ `^e` [Composer ... Reading:...]
  für die Rückkehr zur Testumgebung.
- 16. Kontrolle der Ergebnisse

Layoutsynthese

6. Synthese eines kompaktierten symbolischen Layouts
Der Vorgang der (automatischen) Layoutsynthese besteht eigentlich aus einer Folge von drei nacheinander ablaufender Einzelaktionen.

Platzierung: Ersetzung der Transistoren des Schematic durch, in Kanallänge und -weite parametrisierbare, *symbolic devices*.[2]

[2] Siehe „CADENCE Grundlagen“, 3.9 Symbolisches Layout.

Verdrahtung: Verbindung der Transistoren untereinander durch *symbolic wires*.[2] Sind dabei Übergänge zu anderen Layern notwendig, so werden die entsprechenden Kontakte erzeugt.

Kompaktierung: Es wird versucht, das Layout flächenoptimal anzuordnen. Dabei wird der Entwurf, entsprechend den Design Rules, abwechselnd in x- und y-Richtung gestaucht.

Die Zellgenerierung der Layoutsynthese erzeugt als Ausgabe eine `compacted`-View des Schematic. Es gibt vielfältige Möglichkeiten diesen Vorgang zu beeinflussen, Zwischenergebnisse zu erzeugen und zu bewerten und so letztendlich ein „besseres“ Layout zu erhalten.

Die nachfolgenden Schritte beschreiben den einfachsten Fall, wo man versucht mit möglichst wenig manuellen Eingriffen zu einem Layout zu gelangen.

☐ `Tools - Design Synthesis - Layout Synthesis` [Composer-Schematic...]

☐ `LAS - Generate...` [Composer-Schematic...]

≡ `Run Mode` = `Foreground` [Generate]
`Run Placement` = `on`
`Run Routing` = `on`
`Run Compaction` = `on`
`Input  Library` = ⟨*libId*⟩
`Output Library` = ⟨*libId*⟩
`Input  Cell` = ⟨*cellId*⟩
`Input  View` = `schematic`
`Output View` = `compacted`
`Compaction Sequence` = `xyxy` | `yxyx` | `...`
`Auto Jog` = `Unlimited`

Achtung: Bevor die Fill-Form mit `OK` bestätigt wird, müssen einige der Optionen eingestellt werden.

Bei den nachfolgend beschriebenen Fill-Forms sind jeweils nur diejenigen Optionen angegeben, bei denen man eine Wahlfreiheit hat oder wo bestimmte Werte eingestellt sein müssen.

≡[3] `Set Constraints ...` = $\uparrow_l$ `Pin` [Generate]

≡[3] `Internal Pins` = `off` | `on` [Pin Constraints]
`Access Type` = `Vertical`

Normalerweise sollte `Internal Pins` deaktiviert sein, dann werden die Pins an den Grenzen des Layouts erzeugt; andernfalls werden nur im Layout Kontakte auf die Metalllayer erzeugt.

In dem Abschnitt `Pin Constraints` können für einzelne Pins der Schaltung deren Anschlussrichtung und -layer festgelegt werden. Dazu wählt man zuerst einen Pin über `Next` | `Previous` aus und bestätigt nach der Einstellung von `Side` und `Layer`-Paaren den Befehl mit `Merge` | `Replace`.[4]

[3]Gegenüber den Voreinstellungen muss hier nichts zwingend verändert werden.
[4]Die genaue Benutzung ist der CADENCE Online-Hilfe zu entnehmen.

≡ `Set Constraints ...` = $\uparrow_l$ `Placement` [Generate]

≡ `Method` = `Optimized` | `Quick` [Placement Options]
`Aspect Ratio` = `Rows 1`

≡ `Set Constraints ...` = $\uparrow_l$ `Routing` [Generate]

≡ `Routing Style` = `H1V2` | `V1H2` | `Hybrid` [Routing Options]
`-- Metal:`
`Intra Row Tracks` = `Auto`
`Over Transistor Routing` = `None` | `Moderate` | `...`
`-- Poly:`
`Intra Row` = `Minimum` | `Moderate`
`Inter Row` = `Auto`
`I/O Routing` = `None`
`N to P Gate Routing` = `poly`

≡ `Set Constraints ...` = $\uparrow_l$ `Pwr/Gnd` [Generate]

≡ `Unmerged Bus Width` = ⟨*nr*⟩ [Pwr/Gnd Options]
`Merge Bus` = `off` | `on`
`No Jog Horizontal Bus` = `on`
Die Breite der Leitungen für `PWR` und `GND` sollte mindestens `5.00` betragen.

≡ `Set Constraints ...` = $\uparrow_l$ `Process` [Generate]

≡ `Diffusion Alignment` = `Inter-Row Side` | `Center` [Process Options]
`Stacked Vias` = `off`
`prBoundry` = `off`
`leakage Diode` = `off`
`Bent Gate` = `None`
`prCell Type` = `Macro`
`Location` = `Under Power`
`-- Substrate Contacts:`
`Substrate Contacts` = `Under Power`
`Spacing` = `Sub to Sub` ⟨*nr*⟩
In der Fill-Form kann der Abstand zwischen Substratkontakten festgelegt werden, höchstens: `50.00`.

Wird nach dem Ausfüllen aller Fill-Forms der Befehl zur Generierung des Layouts gestartet, so erscheinen in dem CADENCE Eingabefenster ([icfb - Log:...]) die Ausgaben der Programme und ein neues Editorfenster wird für die erzeugte `compacted`-View geöffnet.

7. Kontrolle der generierten `compacted`-View
Ist man mit den Syntheseergebnissen nicht zufrieden, so muss die Layoutgenerierung (mit anderen Optionen) erneut gestartet werden; ansonsten kann der Schematic-Editor verlassen werden.

□ `Window - Close` [Composer-Schematic...]

Layoutnachbearbeitung

8. Aufruf des Kompaktierungswerkzeugs
Alle nachfolgenden Schritte gehen davon aus, dass die `compacted`-View mit dem Layout-Editor bearbeitet wird.

□ `Tools - Compactor` [Virtuoso... compacted]

9. Nachbearbeitung des kompaktierten Layouts
optional: Dieser Schritt kann im Allgemeinen ausgelassen werden. Er ist nur notwendig, wenn Transistoren ausgerichtet werden sollen (unterschiedliche Lage in den Zeilen) oder weitere Kompaktifizierungsschritte notwendig sind.

Eine ausführliche Beschreibung der Kompaktierungswerkzeuge ist der Online Dokumentation zu entnehmen, hier sind nur die Schritte für solch eine Ausrichtung beschrieben.

≡ $\uparrow_l$ `NS` [LSW]
Um nicht versehentlich Teile des Layouts, sondern nur die Transistoren (Instanzen in der Layouthierarchie), zu selektieren, wird die Selektierbarkeit der Layer abgeschaltet.

□ `Constraints - Alignment - Create...` [Virtuoso... compacted]

≡ `Direction of Constraint` = `y` [Create Alignment Constraint]
`Alignment Edge` = `top|bottom`
`Object Layer` = `BBox`
Jetzt müssen der Reihe nach alle P-Kanal/N-Kanal Transistoren im symbolischen Layout angewählt werden: $\uparrow_l$ ⟨*PTR/NTR*⟩.

Anmerkung: dies geschieht am einfachsten, indem man sich an den „überstehenden“ Diffusionslayern orientiert: `CPPI` für P-Transistoren und `CNPI` für N-Transistoren.

Nachdem der erste Transistor ausgewählt wurde, erscheint in der Fill-Form der Eintrag `Object Layer`. Dies ist der Layer, an dem später alle Objekte einer „alignment group“ ausgerichtet werden (`BBox`).

Sind schließlich alle Transistoren einer Art ausgewählt, so kann die Alignment-Bedingung mit `Apply` erzeugt werden.

Eine Kontrolle der eingegebenen Randbedingungen kann mit folgenden Befehlen durchgeführt werden.

□ `Constraints - Alignment - Edit...` [Virtuoso... compacted]

≡ `Select by Status` = `all` [Edit Alignment Constraint]
`Select by Direction` = `y`
`Cycle through` = `constraint groups|individual objects`

`constraint groups`	
`Previous` | `Next`	wechselt zwischen den Gruppen
`Alignment Constraint = ⟨...⟩`	aktiviert, deaktiviert oder löscht Constraints
`individual objects`	
`Previous` | `Next`	wechselt zwischen den Objekten einer Gruppe
`Alignment Edge = ⟨...⟩`	zeigt/verändert die Kante für die Ausrichtung
`Object Layer = ⟨...⟩`	zeigt/verändert den Ausrichtungslayer
`Delete Object from Constraint`	löscht Objekt aus der Gruppe

Alle Veränderungen sind mit `Apply` zu bestätigen.

Für die Sannungsversorgungen `vdd` und `gnd` ist die Breite der Metallbahn entsprechend den Angaben bei der Layoutsynthese als Randbedingung vorzugeben (zum Beispiel: `5.0`). Ansonsten wird bei der nächsten Kompaktierung die Mindestbreite von 1.2 benutzt.

≡ ↑$_l$ `AS` [LSW]

Es werden wieder alle Layer selektierbar gemacht.

□ `Constraints - Wire Width - Create...` [Virtuoso... compacted]

↑$_l$ ⟨*`wire`*⟩ [Virtuoso... compacted]

≡ `Width Setting` = ⟨*nr*⟩ z.B.: `5.0` [Create Wire Width Constraint]

Nach Auswahl der `vdd`- und `gnd`-Leitungen wird die entsprechende Breite festgelegt.

Nach Festlegung aller Randbedingungen kann die Nachkompaktierung gestartet werden. Sind die Ergebnisse nicht zufrieden stellend, so muss man gegebenenfalls die Kompaktierung mit anderen Randbedingungen wiederholen.

□ `Compact - Recompact` [Virtuoso... compacted]

10. Erzeugen eines „richtigen" Layouts

□ `Compact - Convert to Geometric...` [Virtuoso... compacted]

≡ `Output Cell Name` = ⟨*`cellId`*⟩ [Convert To Geometric]
`Output View Name` = `layout`
`Smash To View Level` = `on`
`Conversion Option` = `fill gap`

Eine `layout`-View der Schaltung wird erzeugt und in einem neuen Fenster dargestellt: [Virtuoso... layout]

Durch die Option `Smash To View Level` wird dabei die Hierarchie des symbolischen Layouts aufgelöst und alle Strukturen werden als Polygone dargestellt.

Achtung: lässt man die Option ausgeschaltet (die Hierarchie bleibt erhalten), so kann es vorkommen, dass Teile des Layouts nicht auf den Raster liegen! Eine Nachbearbeitung ist in diesem Fall unbedingt notwendig.

Anschließend kann die `compacted`-View gesichert und das Editor-Fenster geschlossen werden.

□ `Design - Save`/⊙ `f2` [Virtuoso... compacted]

□ `Window - Close`/⊙ `^w` [Virtuoso... compacted]

Simulation der extrahierten Netzliste

11. Endbearbeitung des Layouts, Extraktion und Simulation der extrahierten Netzliste

☐ `Tools - Layout` [Virtuoso... layout]
Lädt den Layout-Editor als Werkzeug für alle weiteren Schritte.

Das weitere Vorgehen ist im Detail in „Full-Custom Design“ beschrieben, dabei sind folgende Schritte auszuführen:

- 5. Design Rule Check
- – 6. entfällt, da die Anschlüsse der Schaltung schon durch die Layoutsynthese gekennzeichnet wurden.
- 7. Extraktion der elektrischen Netzliste für die Simulation
- 8. Extrahiertes Netz ansehen
- 9. Schematic-Editor für Testumgebung aufrufen
- – 10. entfällt, da das Symbol schon vorher, aus der `schematic`-View heraus, erzeugt wurde (Schritt 4.).
- 11. Design der Testumgebung
  Wurde noch *keine* Simulation durchgeführt – Punkt 5. dieser Beschreibung –, so muss jetzt die Testumgebung entworfen werden.

  Andernfalls wurde das schon vorhandene Schematic vorher geöffnet.
- 12. Simulation vorbereiten
- 13. Stimuli editieren
- 14. Simulation starten
- 15. Signale für Waveforms markieren
- 16. Kontrolle der Ergebnisse
- 17. CADENCE beenden