Diese Beschreibung ist als Ergänzung zu „Full-Custom Layout“ gedacht. Hier wird beschrieben, wie aus einer Transistornetzliste – als Schematic – mit Hilfe der Werkzeuge zur Layoutsynthese ein Full-Custom Layout erzeugt wird.

Die Eingaben des Benutzer sind dabei mit den schon vorher verwendeten Symbolen dargestellt.

Der Entwurfsablauf ist, als Erganzung zu dem in „Full-Custom Layout“ 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 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.

1 Layoutsynthese & Full-Custom Layout

1. Start des Systems

   > `es2_cdk` [xterm]

   Achtung: Da dieser Befehl ein Skript startet, darf er nicht im Hintergrund ausgeführt werden. Beim ersten Aufruf (in einem Directory) wird nach dem Prozeß `-ecpd-` und dem Programm `-icfb-` gefragt, dabei sind bei beiden Fragen die voreingestellten Werte zu bestätigen.

2. Design der Schaltung — siehe: „Dokumentation zu CADENCE“, 2.6 Der Schematic-Editor.

   □ `Open - Design...` [ES2 0.7um ...]

   ≡ `Library Name` `=<lib name>` [Open Design]
   `Cell Name` `=<cell name>`
   `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` [Editing:...]

≡ `Library` `=<lib>` `/ sample` [ES2 ... Library Menu]
`Cell` `=<cell>` `/ 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 anschließend über Properties angegeben.

□ `Edit - Properties - Objects... / ⊙ q` [Editing:...]

↑$_l$ `Add` [Edit Object Properties]

≡ `Name` `=w` `/ l` [Add Property]
`Type` `=float`
`Value` `=<width> / 0.8`

Nach Abschluß des Befehls werden die Properties an den instanziierten Transistoren als `<width>/<length>` 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` [Editing:...]

≡ `Pin Names` `=<string>` `/ vdd, gnd` [Add Pin]
`Direction` `=<input/output>` `/ inout`
`Usage` `=schematic`

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

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

□ `Design - Create Cellview - From Cellview...` [Editing:...]

≡ `Library Name` `=<lib name>` [Cellview From Cellview]
`Cell Name` `=<cell name>`
`Display Cellview` `=<off>`
`Edit Options` `=<off/on>`
`From View Name` `=schematic`
`To View Name` `=symbol`

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

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

□ `Design - Check and Save` / ⊙ `X` [Editing:...]

4. Design der Testumgebung und Simulation der Schaltung[2]
   Das Vorgehen entspricht genau demjenigen in: „Full-Custom Design“

   9. Design der Testumgebung

   10. Simulation vorbereiten

   11. Stimuli editieren

   12. Simulation starten

   13. 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` [Editing:...]

   für die Auswahl von Signalen zur Anzeige.

   □ `Design - Hierarchy - Return` / ⊙ `^e` [Reading:... schematic]

   für die Rückkehr zur Testumgebung.

   14. Kontrolle der Ergebnisse

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

   Plazierung: Ersetzung der Transistoren des Schematic durch in Kanallänge und -weite parametrisierbare *symbolic devices*.[3]

   Verdrahtung: Verbindung der Transistoren untereinander durch *symbolic wires*. Sind dabei Übergänge zu anderen Layern notwendig, so werden die entsprechenden Kontakte erzeugt.[3]

   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 letztendlich 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.

[2] **optional:** nur wenn die Transistornetzliste der `schematic`-View simuliert werden soll. Die Simulation sollte aber *in jedem Fall* später mit den extrahierten Layoutdaten durchgeführt werden.

[3] Siehe „Dokumentation zu CADENCE“, 2.5.6 Symbolisches Layout.

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` [Editing:...]

□ `LAS - Generate Cell...` [Editing:...]

≡ `Input Library` = `<lib name>` [Generate]
`Output Library` = `<lib name>`
`Input Cell` = `<cell name>`
`Input View` = `schematic`
`Output View` = `compacted`
`Timing File` = `<off>`
`Run Placement` = `<on>`
`Run Routing` = `<on>`
`Run Compaction` = `<on>`
`Compaction Sequence` = `<xyxy/yxyx/...>`
`Auto Jog` = `<unlimited>`
`Object Grid` = `<on> 0.1`

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.

≡ `Set Constraints ...` = ↑$_l$ `Pin` [4] [Generate]

≡ `Internal Pins` = `<off/on>` [4] [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 Metallayer erzeugt.

In dem Abschnitt `Terminal Specific` können für einzelne Pins der Schaltung deren Anschlußrichtung und -layer festgelegt werden. Dabei ist jeweils ein Pin mit `Next/Previous` auszuwählen und nach Einstellung von `Side` und `Layer`-Paaren der Befehl mit `Apply` zu bestätigen.[5]

≡ `Set Constraints ...` = ↑$_l$ `Placement` [Generate]

≡ `Method` = `<Optimized/Quick>` [Placement Options]
`Aspect Ratio` = `<Rows> 1`
`Cluster` = `<Auto>`
`Flatten` = `<Auto>`

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

≡ `Set Constraints ...` = ↑$_l$ `Routing` [Generate]

≡ `Routing Style` = `<H1V2/V1H2/Hybrid>` [Routing Options]
`-- Metal:`
`Intra Row Tracks` = `Auto`
`Over Transistor Routing` = `<on>`
`Use Vertical Layer Horizontally` = `<on>`
`-- Poly:`
`Intra Row` = `<on>`
`Inter Row` = `<Auto>`
`I/O Routing` = `<None>`
`N to P Gate Routing` = `<poly>`

≡ `Set Constraints ...` = ↑$_l$ `Pwr/Gnd` [Generate]

≡ `Horizontal Bus Width` = `<nr>` [Pwr/Gnd Options]
`Merge Horizontal Bus` = `<off/on>`
`No Jog Horizontal Bus` = `<on>`

Die Breite der Leitungen für `PWR` und `GND` sollte mindestens 5.00 betragen.

≡ `Set Constraints ...` = ↑$_l$ `Process` [Generate]

≡ `Stacked Vias` = `<off>` [Process Options]
`Bent Gate` = `<off>`
`-- Substrate Contacts:`
`Location` = `<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 ([ES2 0.7um ...]) die Ausgaben der Programme und ein neues Editorfenster wird für die erzeugte `compacted`-View geöffnet.

Ist man mit den Syntheseergebnissen nicht zufrieden, so muß die Layoutgenerierung (mit anderen Optionen) erneut gestartet werden; ansonsten kann der Schematiceditor verlassen werden.

□ `Window - Close` [Editing:... schematic]

6. Aufruf des Kompaktierungswerkzeugs
Alle nachfolgenden Schritte gehen davon aus, daß die `compacted`-View bearbeited wird.

□ `Tools - Compactor` [Editing:...]

7. Nachbearbeitung des kompaktierten Layouts[6]
   Der häufigste Grund, weswegen Nachbearbeitungen wünschenswert sind, ist die unterschiedliche Lage der Transistoren in den Zeilen.

   Eine ausfürliche 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...` [Editing:...]

   ≡ `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...` [Editing:...]

   ≡ `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.

[6] **optional:** nur notwendig, wenn Transistoren ausgerichtet werden sollen oder weitere Kompaktifizierungsschritte notwendig sind.

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.

≡ $\uparrow_l$ `AS` [LSW]

Es werden wieder alle Layer selektierbar gemacht.

□ `Constraints - Wire Width - Create...` [Editing:...]

$\uparrow_l$ `<wire>` [Editing:...]

≡ `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 zufriedenstellend, so muß man gegebenenfalls die Kompaktierung mit anderen Randbedingungen wiederholen.

□ `Compact - Recompact` [Editing:...]

8. Erzeugen eines „richtigen“ Layouts

□ `Compact - Convert to Geometric...` [Editing:...]

≡ `Output Cell Name` `=<cell name>` [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: [Editing:... 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` [Editing:... compacted]

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

9. Endbearbeitung des Layouts, Extraktion und Simulation

   □ `Tools - Layout` [Editing:... 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:

   4. Design Rule Check

   5. — entfällt, da die Anschlüsse der Schaltung schon durch die Layoutsynthese gekennzeichnet wurden.

   6. Extraktion der elektrischen Netzliste für die Simulation

   7. Extrahiertes Netz ansehen

   8. — entfällt, da das Symbol schon vorher, aus der `schematic`-View heraus, erzeugt wurde.

   9. Design der Testumgebung
   Wurde noch *keine* Simulation durchgeführt – Punkt 4. dieser Beschreibung –, so muß jetzt die Testumgebung entworfen werden.

   Andernfalls muß nur das schon vorhandene Schematic geöffnet werden.

   □ `Open - Design...` [ES2 0.7um ...]

   ≡ `Library Name` = `<lib name>` [Open Design]
   `Cell Name` = `<testcell name>`
   `View Name` = `schematic`

   10. Simulation vorbereiten

   11. Stimuli editieren

   12. Simulation starten

   13. Signale für Waveforms markieren

   14. Kontrolle der Ergebnisse

   15. fertig !
   Die Layoutsynthese erzeugt eine `las`-View, diese entspricht dem symbolischen Layout vor der ersten Kompaktierung, und sollte gesichert werden.