Vom RTL zum GDS

Einführung in das VLSI-Design von ICs mit Fokus auf Cadence Innovus

Fatih Mollamehmetoglu

17. Januar 2026

Inhaltsverzeichnis

1 Motivation und Zielsetzung

Das Ziel dieser Ausarbeitung ist es, Studierenden der Elektro- und Informationstechnik, Informatik sowie verwandter Fachrichtungen einen verständlichen Einblick in die Welt des digitalen Designs integrierter Schaltungen zu geben. Dabei soll ein grundlegendes Verständnis für die Entwicklung moderner Technologien vermittelt werden, sodass die Leserinnen und Leser in der Lage sind, sich anschließend eigenständig weiter in die Materie einzuarbeiten.

Die Relevanz dieses Themas ergibt sich aus der wachsenden Komplexität heutiger integrierter Schaltungen und der zentralen Rolle des VLSI-Designs in nahezu allen Bereichen moderner Elektronik. Ein praxisnaher Bezug wird durch die Darstellung des spezifischen Design-Flows innerhalb der Firma REE (Renesas Electronics Europe) geschaffen. Dies soll den Einstieg in reale industrielle Prozesse erleichtern und eine Grundlage für weiterführende Projekte oder Tätigkeiten bieten.

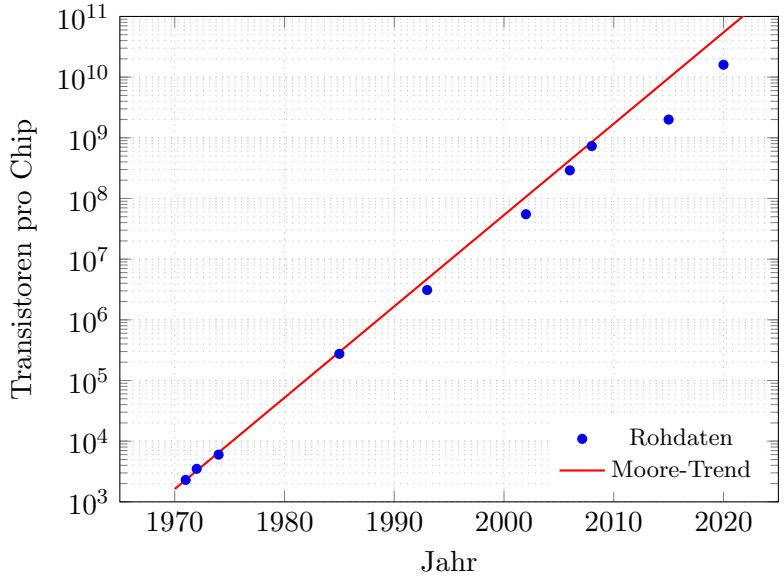


Abbildung 1: Mooresche Gesetz - Die exponentielle Zunahme der Transistordichte

Berechnung der Trendlinie

Die Trendlinie wird durch die exponentielle Ausgleichsgerade approximiert:

$$y(x) = y_0 \cdot 2^{\frac{x-x_0}{T}}$$

$y(x)$ Anzahl der Transistoren im Jahr x

y_0 Startwert der Transistoranzahl im Basisjahr x_0

T Verdopplungszeit in Jahren (hier ca. 2 Jahre)

2 Einführung in das Thema

2.1 Grundlagen des IC- und VLSI-Designs

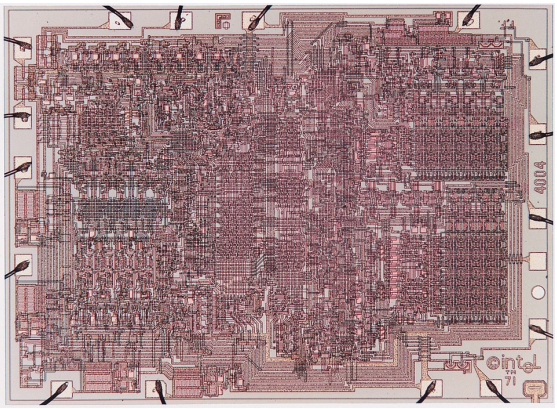


Abbildung 2: erster digital entwickelter IC, Intel 4004

Integrierte Schaltungen (Integrated Circuits, ICs) sind Grundlage moderner Elektronik, welche uns überall im Alltag begegnet. Sie ermöglichen die Funktionalität von Computern, Smartphones und einer Vielzahl weiterer Systeme. Ein IC ist hierbei im Wesentlichen eine Ansammlung von elektronischen Bauelementen - wie Transistoren, Widerständen und Kondensatoren - die auf einem einzigen Halbleiterchip integriert sind. Eben diese Integration erlaubt eine kompakte Bauweise, hohe Leistungsfähigkeit und geringe Kosten.

Mit der zunehmenden Miniaturisierung und steigenden Anforderungen an Rechenleistung entwickelte sich das VLSI-Design (Very Large-Scale Integration) zu einer Schlüsseltechnologie. VLSI bezeichnet die Integration von Millionen bis Milliarden Transistoren auf einem einzigen Chip. Diese hohe Integrationsdichte ermöglicht nicht nur leistungsstarke Prozessoren und Speicher, sondern auch energieeffiziente Lösungen für mobile und eingebettete Systeme. Die Komplexität solcher Designs erfordert den Einsatz spezialisierter Methoden und Werkzeuge, um sowohl Funktionalität als auch Zuverlässigkeit und Herstellbarkeit sicherzustellen.

Die Entwicklung integrierter Schaltungen erfolgt heute in einem strukturierten Design-Flow, der sowohl Hardwarebeschreibung als auch automatisierte Werkzeuge umfasst. Typischerweise beginnt der Prozess mit der Spezifikation der gewünschten Funktionalität, gefolgt von der Beschreibung in Hardwarebeschreibungssprachen wie VHDL oder Verilog. Anschließend wird das Design durch Synthese in eine Netzliste überführt, die die logische Struktur der Schaltung beschreibt.

Darauf folgt die Place-and-Route-Phase, in der die logischen Elemente physisch auf dem Chip angeordnet und die Verbindungen hergestellt werden. Moderne Tools berücksichtigen dabei Aspekte wie Timing, Leistungsaufnahme und Flächenoptimierung. Nach umfangreichen Verifikations- und Signoff-Schritten (Simulation, Timing-Analyse, DRC und LVS) wird das finale Layout als GDSII-Datei an die Fertigung übergeben.

Dieser Flow wird heute stark durch EDA-Tools (Electronic Design Automation) und Methoden wie Design-for-Test, Low-Power-Optimierung sowie IP-Reuse unterstützt, um die Komplexität beherrschbar zu machen und die Time-to-Market zu verkürzen.

2.2 Startpunkt des Design-Flows

Wie in Abbildung 3 dargestellt, lässt sich der gesamte IC-Designprozess grob in zwei Hauptbereiche unterteilen: das **Frontend** und das **Backend**. Das Frontend umfasst die frühen Phasen wie Systemspezifikation, Architekturdesign sowie das funktionale und logische Design. Hier wird die gewünschte Funktionalität des Chips beschrieben und in einer Hardwarebeschreibungssprache (z. B. VHDL oder Verilog) modelliert. Anschließend erfolgt die **Synthese**, bei der diese abstrakte Beschreibung in eine Gate-Level-Netzliste überführt wird. Diese Netzliste bildet die logische Struktur der Schaltung auf Basis standardisierter Zellen aus der verwendeten Technologie-Bibliothek.

Das Backend beginnt dort, wo die logische Beschreibung vorliegt und in eine physische Implementierung überführt werden muss. Der Ausgangspunkt unseres Backend-Designprozesses ist daher eine vollständig verifizierte Gate-Level-Netzliste, die aus der Synthesephase stammt. Sie enthält alle funktionalen Informationen und die logischen Verbindungen, jedoch noch keine physische Anordnung der Komponenten auf dem Chip.

Neben der Netzliste liegen zu diesem Zeitpunkt auch die relevanten Design-Constraints vor. Dazu gehören unter anderem Timing-Anforderungen, Vorgaben zur Leistungsaufnahme sowie Flächenbeschränkungen. Diese Randbedingungen sind entscheidend, um die physische Implementierung so zu gestalten, dass die funktionalen und technologischen Spezifikationen eingehalten werden.

Damit ist die Ausgangslage klar definiert: eine funktional korrekte, technologiegebundene Netzliste sowie die zugehörigen Constraints, die im weiteren Verlauf des Backend-Flows berücksichtigt werden müssen.

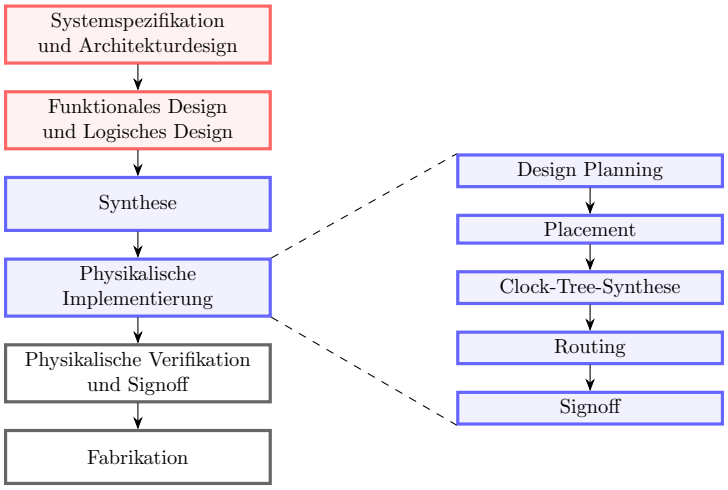


Abbildung 3: Allgemeiner Design-Flow beim IC-Design

3 Physischer IC-Designflow

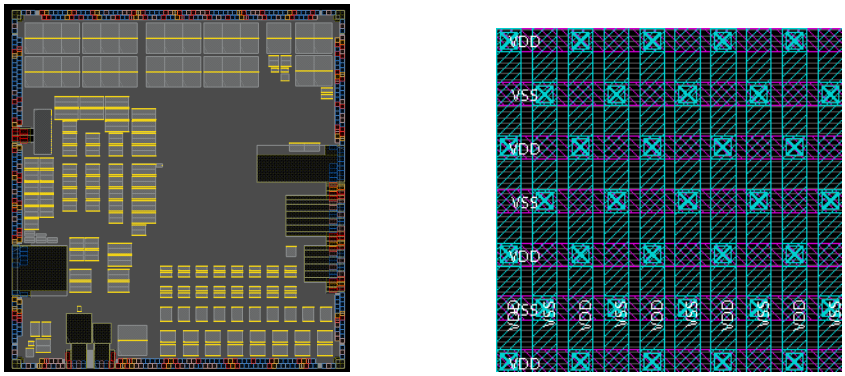
3.1 Floorplaning: Strukturierung des Chips

Mit der Ausgangslage einer verifizierten Gate-Level-Netzliste und den zugehörigen Design-Constraints, beginnt die physische Implementierung mit dem **Floorplanning**. Beim Floorplanning wird die grundlegende Struktur des Chips festgelegt. Ziel ist es, eine effiziente Anordnung der funktionalen Blöcke, Makrozellen und Standardzellen zu planen, sodass spätere Schritte wie Platzierung und Verdrahtung optimal durchgeführt werden können.

Wichtige Aspekte beim Floorplanning sind:

- **Chipgröße und Form**: Die Dimensionen müssen den Flächenvorgaben entsprechen und gleichzeitig genügend Platz für alle Komponenten sowie die Verdrahtung bieten.
- **Makroplatzierung**: Große Blöcke wie Speicher oder IP-Cores werden strategisch positioniert, um kurze Signalwege und gute Timing-Eigenschaften zu gewährleisten.
- **Power-Grid-Design**: Die Stromversorgung wird früh geplant, um eine stabile Versorgung aller Bereiche sicherzustellen.
- **I/O-Pads und Schnittstellen**: Die Positionierung der Ein- und Ausgänge beeinflusst die Signalführung und die Integration ins Gehäuse.

Abbildung 4a zeigt ein Design nach dem Floorplanning: Die IO-Pads sind entlang des Chiprandes angeordnet, während große IP-Blöcke und Makrozellen im Inneren platziert sind. Freie Bereiche dienen später der Platzierung von Standardzellen. Eine weitere Detailansicht in Abbildung 4b verdeutlicht das Power-Grid-Design, das für eine stabile Stromversorgung sorgt. Hier sind die VDD- und VSS-Leitungen sowie die zugehörigen Strukturen zu erkennen, die bereits in dieser frühen Phase berücksichtigt werden müssen. Diese sind über das gesamte Design verteilt in einem Grid angeordnet, aber aufgrund von Übersichtsgründen in der Abbildung 4a ausgeblendet (Chakravarthi 2022, Siehe S. 20 ff.).



(a) Chip-Layout nach dem Floorplanning

(b) Detailansicht des Power-Grid-Designs

Abbildung 4: Darstellung des Floorplanning und des Power-Grid-Designs

3.2 Placement: Anordnung der Standardzellen

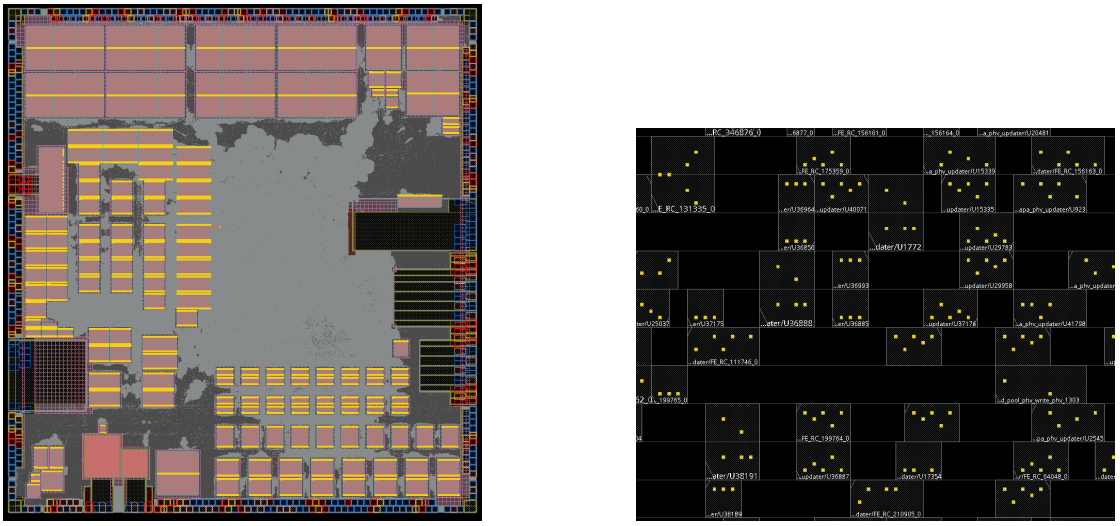
Nach dem Floorplanning, bei dem die grobe Struktur des Chips und die Position der Makrozellen festgelegt wurden, folgt der Schritt des **Placements**. Hier werden die Millionen von Standardzellen, die die eigentliche Logik des Designs bilden, in den dafür vorgesehenen Bereichen des Chips platziert. Ziel ist es, eine Anordnung zu finden, die sowohl die funktionalen Anforderungen als auch die physikalischen Randbedingungen erfüllt.

Beim Placement müssen mehrere Aspekte berücksichtigt werden:

- **Timing-Optimierung**: Die Positionierung der Zellen beeinflusst die Länge der Verbindungen und damit die Signallaufzeiten. Kurze kritische Pfade sind entscheidend für die Einhaltung der Timing-Constraints.
- **Congestion-Vermeidung**: Eine gleichmäßige Verteilung der Zellen verhindert Engpässe bei der späteren Verdrahtung.
- **Legalization**: Nach der initialen Platzierung müssen alle Zellen auf erlaubten Positionen innerhalb des Standardzellenrasters liegen, ohne Überlappungen.

Abbildung 5a zeigt das Design nach dem Placement. Die großen Blöcke im Inneren sind die bereits beim Floorplanning positionierten Makrozellen und IP-Blöcke. Die graue Fläche, die wie eine „Wolke“ wirkt, repräsentiert die Vielzahl der platzierten Standardzellen, die nun die logische Funktionalität des Chips abbilden. Am Rand sind weiterhin die IO-Pads sichtbar, die die Schnittstellen zur Außenwelt bilden.

Um die Struktur dieser „Wolke“ besser zu verdeutlichen, zeigt Abbildung 5b einen vergrößerten Ausschnitt. Hier wird deutlich, dass die Wolke aus einer Vielzahl einzelner Standardzellen besteht, die eng nebeneinander angeordnet sind. Jede dieser Zellen enthält Pins (gelbe Punkte), die später über das Routing miteinander verbunden werden.



(a) Design nach dem Placement

(b) Vergrößerter Ausschnitt der „Wolke“

Abbildung 5: Übersicht (links) und Zoom (rechts) des Placements.

3.3 Clock Tree Synthesis: Taktverteilung

Nach der Platzierung aller Standardzellen folgt die **Clock Tree Synthesis (CTS)**, ein entscheidender Schritt zur Sicherstellung einer korrekten und synchronen Taktverteilung im gesamten Design. Während das Taktsignal im logischen Design als ideal angenommen wird, muss es in der physischen Implementierung über eine Vielzahl von Pfaden verteilt werden. Ziel der CTS ist es, einen Baum aus Taktleitungen und Pufferzellen zu erzeugen, der alle Takt-Sinks (z. B. Flip-Flops) erreicht und dabei die zeitlichen Anforderungen erfüllt.

Die wichtigsten Herausforderungen bei der CTS sind:

- **Minimierung des Skews**: Alle Takt-Sinks sollen das Signal möglichst gleichzeitig erhalten, um Setup- und Hold-Verletzungen zu vermeiden.
- **Optimierung der Latenz**: Die Gesamtlaufzeit des Taktsignals vom Ursprung bis zu den Sinks muss innerhalb der vorgegebenen Grenzen liegen.
- **Energieeffizienz**: Durch den Einsatz von Clock-Gating-Zellen kann die dynamische Leistungsaufnahme reduziert werden.

Abbildung 6 zeigt die Struktur eines Clock-Trees nach der CTS-Phase. Die grünen Dreiecke repräsentieren Puffer, die das Taktsignal verstärken und verteilen. Die türkisen Kreise sind Clock-Gating-Zellen, die eine selektive Abschaltung des Takts ermöglichen, um Energie zu sparen. Die roten Quadrate an den Blättern des Baumes sind die Takt-Sinks, also die Endpunkte des Taktsignals, typischerweise Flip-Flops. Auf der Y-Achse ist die Zeit aufgetragen, was die zeitliche Verteilung des Taktsignals und den Unterschied an den Takt-Sinks verdeutlicht. Anhand des Höhenunterschieds zweier Takt-Sinks lässt sich so der zeitliche Versatz (Skew) des Taktsignals zwischen zwei Flipflops bestimmen.

Diese Baumstruktur ist nicht nur eine theoretische Darstellung, sondern wird im Design physisch umgesetzt. Das bedeutet, dass die in der Abbildung gezeigten Puffer und Clock-Gating-Zellen tatsächlich als zusätzliche Zellen in das Layout eingefügt werden. Sie werden nach der CTS-Phase im Chip platziert und später wie alle anderen Netze geroutet. Dadurch entsteht ein physischer Clock-Tree, der die logische Struktur widerspiegelt und die Taktverteilung im realen Chip sicherstellt.

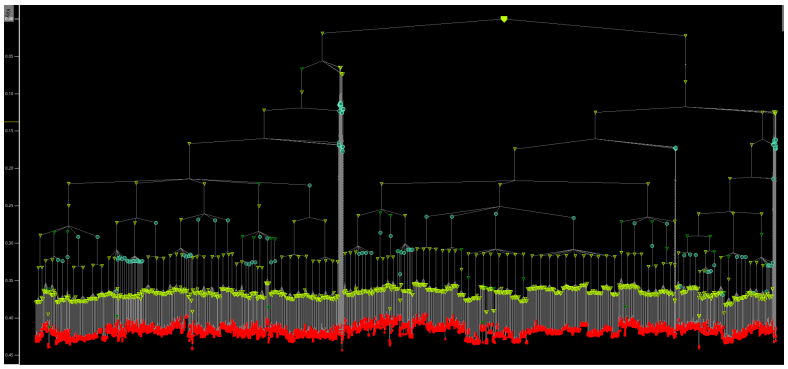


Abbildung 6: Visualisierung des Clock-Trees nach der CTS-Phase: Puffer (grün), Clock-Gating-Zellen (türkis) und Sinks (rot)

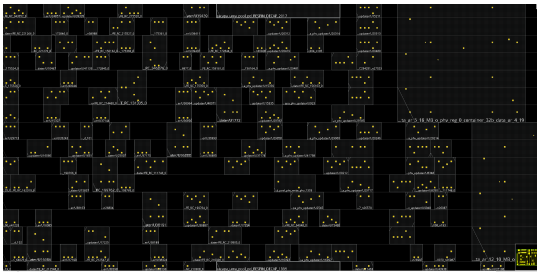
3.4 Routing: Physische Verbindung der Netze

Nach der erfolgreichen Clock Tree Synthesis folgt das **Routing**, bei dem alle logischen Verbindungen des Designs physisch realisiert werden. Während die Platzierung und CTS die Position der Zellen und die Taktverteilung festgelegt haben, sorgt das Routing dafür, dass sämtliche Netze über die verfügbaren Metall-Layer verbunden werden. Dieser Schritt ist besonders komplex, da Millionen von Verbindungen unter Berücksichtigung zahlreicher Randbedingungen erstellt werden müssen.

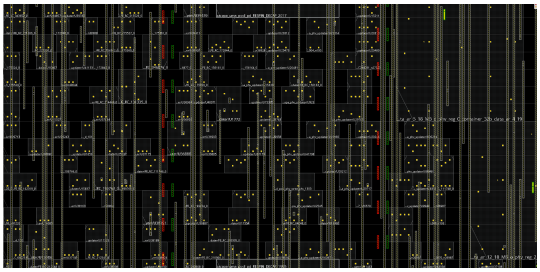
Die wichtigsten Ziele beim Routing sind:

- **Einhaltung der Design-Regeln (DRC)**: Alle Leitungen müssen den technologischen Vorgaben entsprechen, z. B. Mindestabstände und Breiten.
- **Timing-Optimierung**: Kritische Netze werden bevorzugt geroutet, um Verzögerungen zu minimieren.
- **Vermeidung von Crosstalk und EM-Problemen**: Die Anordnung der Leitungen beeinflusst Signalstörungen und die Zuverlässigkeit.

Abbildung 7 zeigt einen stark vergrößerten Ausschnitt des Designs nach dem Routing. In Bild (a) sind alle Metall-Layer deaktiviert, sodass nur die Standardzellen und ihre Pins sichtbar sind. Die gelben Punkte markieren die Anschlussstellen der Zellen. In Bild (b) wurde ein einzelner Metall-Layer aktiviert, wodurch die vertikalen Leitungen erkennbar sind, die einige der Pins verbinden. Farblich hervorgehobene Segmente (rot und grün) kennzeichnen spezielle Netze oder Prüfpunkte.



(a) Ausschnitt ohne aktive Metall-Layer: nur Zellen und Pins sichtbar



(b) Mit aktivem Metall-Layer: vertikale Leitungen und Verbindungen

Abbildung 7: Detailansicht des Designs nach dem Routing

3.5 Signoff: Finale Validierung

Nachdem alle logischen Verbindungen geroutet und die physische Implementierung abgeschlossen ist, folgt die **Signoff-Phase**. Sie stellt sicher, dass das Design nicht nur funktional korrekt, sondern auch herstellbar und zuverlässig ist. Während die vorherigen Schritte das Layout erstellt haben, dient Signoff als finale Validierung vor der Übergabe an die Fertigung.

Typische Prüfungen im Signoff sind:

- **Design Rule Check (DRC)**: Überprüfung, ob alle geometrischen Vorgaben der Technologie eingehalten werden, z. B. Mindestabstände und Breiten.
- **Layout versus Schematic (LVS)**: Sicherstellung, dass das physische Layout der logischen Netzliste entspricht.
- **Static Timing Analysis (STA)**: Analyse aller Pfade, um die Einhaltung der Timing-Constraints zu garantieren.
- **Power- und EM-Checks**: Untersuchung auf Stromdichteprobleme, Elektromigration und IR-Drop, um die Zuverlässigkeit sicherzustellen.

Erst wenn alle Signoff-Prüfungen erfolgreich abgeschlossen sind, wird das finale Layout in das GDSII-Format exportiert und an die Fertigung übergeben. Damit endet der physische Design-Flow normalerweise, und der Chip kann in die Produktion gehen.

Es ist jedoch wichtig zu beachten, dass der Signoff nicht immer den endgültigen Abschluss des Designprozesses darstellt. In dieser Phase können Probleme wie Timing-Verletzungen, DRC-Fehler oder Stromversorgungsprobleme auftreten, die eine Anpassung des Layouts erforderlich machen. In solchen Fällen müssen einzelne Schritte des Backend-Flows erneut durchlaufen werden – beispielsweise eine Optimierung der Platzierung, eine Anpassung des Clock-Trees oder ein erneutes Routing. Dieser iterative Charakter des Prozesses ist typisch für komplexe IC-Designs und stellt sicher, dass alle Spezifikationen vor der Fertigung zuverlässig erfüllt werden.

4 Cadence Innovus im Design-Flow

4.1 Überblick über Cadence-Innovus

Cadence Innovus ist ein Electronic Design Automation (EDA)-Tool, das speziell für die physische Implementierung von integrierten Schaltungen entwickelt wurde. Es ist Teil der Cadence Digital Implementation Suite und wird in der Industrie als Standardlösung für den Backend-Design-Flow eingesetzt. Innovus deckt alle wesentlichen Schritte der physischen Designphase ab, darunter Floorplanning, Placement, Clock Tree Synthesis (CTS), Routing und Signoff.

Das Tool bietet leistungsfähige Optimierungsalgorithmen, die darauf ausgelegt sind, Timing-Anforderungen einzuhalten, die Leistungsaufnahme zu reduzieren und die Chipfläche effizient zu nutzen. Innovus ist für Designs mit sehr hoher Komplexität geeignet, die Millionen bis Milliarden von Standardzellen umfassen. Neben einer grafischen Benutzeroberfläche (GUI) unterstützt Innovus auch eine skriptbasierte Steuerung über Tcl, was eine flexible Automatisierung des Design-Flows ermöglicht.

Ein weiterer Vorteil ist die enge Integration mit anderen Cadence-Produkten wie Genus für die logische Synthese und Tempus für die Timing-Analyse. Diese Kombination erlaubt einen durchgängigen und konsistenten Design-Flow von der RTL-Beschreibung bis zum finalen Layout. Innovus wird aufgrund seiner Skalierbarkeit, Effizienz und umfangreichen Funktionen in der Industrie breit eingesetzt und gilt als eines der führenden Werkzeuge für die physische Implementierung moderner ICs.

4.2 EDE-Tool: Rolle und Integration

Nachdem die grundlegenden Funktionen und die Bedeutung von Cadence Innovus im physischen Design-Flow erläutert wurden, stellt sich die Frage, wie die Interaktion zwischen den verschiedenen Werkzeugen und Prozessen effizient gestaltet werden kann. Hier kommt das EDE-Tool ins Spiel. Es bildet die Schnittstelle zwischen den Design-Teams und den eingesetzten EDA-Werkzeugen und sorgt für eine konsistente, automatisierte und kontrollierte Umgebung.

Das EDE-Tool ist eine GUI-basierte Layout-Design-Umgebung, die einen hohen Grad an Automatisierung bietet. Es dient als zentrale Plattform, um den gesamten Place-and-Route-Flow effizient zu steuern und konsistent zu halten. Alle technologie-, bibliotheks- und signoff-relevanten Einstellungen werden zentral gepflegt und sind für alle Nutzer einheitlich verfügbar, was die Qualität und Reproduzierbarkeit des Designs sicherstellt. Innerhalb des EDE können einzelne Aufgaben des Design-Flows entweder als separate Schritte ausgeführt oder zu einem vollständigen beziehungsweise teilweisen PnR-Flow zusammengefasst werden. Für jede Designpartition steht ein eigenes „EDE-Cockpit“ zur Verfügung, das die Steuerung und Überwachung der jeweiligen Prozesse ermöglicht. Dabei können verpflichtende und optionale Schritte flexibel gewählt werden. Einige dieser Schritte müssen in einer festgelegten Reihenfolge ausgeführt werden, während andere unabhängig voneinander gestartet werden können. Durch diese Struktur bietet das EDE-Tool nicht nur eine benutzerfreundliche Oberfläche, sondern auch eine robuste Grundlage

für die Automatisierung komplexer Abläufe. Dies reduziert manuelle Fehler, beschleunigt den Designprozess und erleichtert die Zusammenarbeit in großen Projekten.

Abbildung 8 zeigt die grafische Benutzeroberfläche des EDE-Tools. Auf der linken Seite sind die einzelnen Schritte des physischen Design-Flows dargestellt, die innerhalb des Tools ausgeführt werden können, wie beispielsweise **PreCTS (Initialisierung)**, **Routing**, **Signoff** und weitere Phasen. Diese Schritte können einzeln oder in Kombination gestartet werden. Auf der rechten Seite sind die zugehörigen Skripte sichtbar, die in den jeweiligen Phasen verwendet werden. Sie enthalten die spezifischen Befehle und Parameter, die den Ablauf der einzelnen Schritte steuern. Auf die Inhalte dieser Skripte sowie die wichtigsten Befehle wird im nächsten Abschnitt detailliert eingegangen.

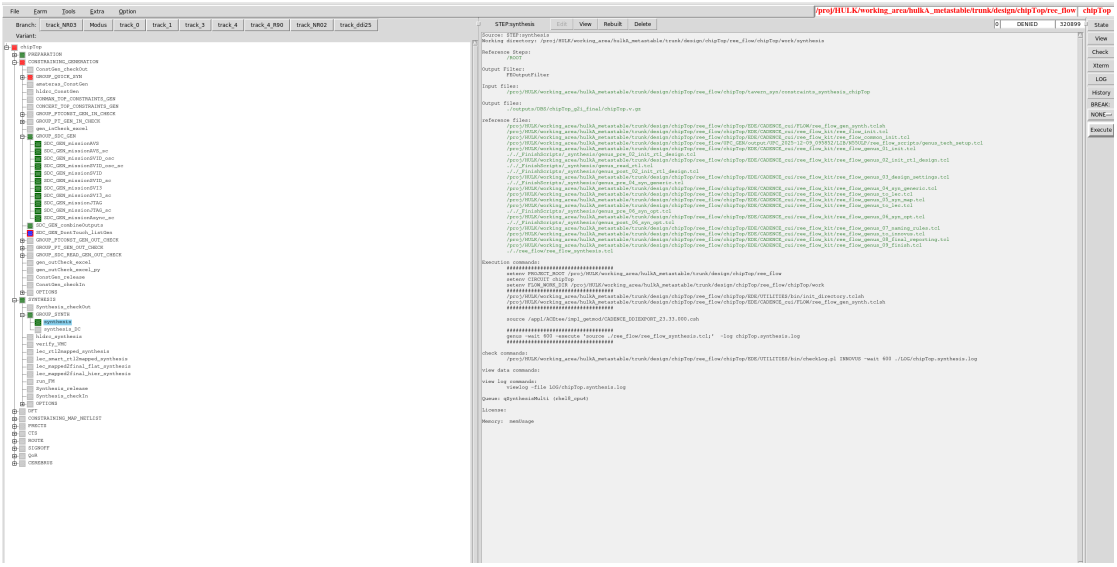


Abbildung 8: EDE-GUI mit Flow-Schritten (links) und zugehörigen Skripten (rechts)

4.3 Implementierung der Designphasen im Innovus

Die Steuerung des physischen Design-Flows erfolgt in unserer Umgebung über das EDE-Cockpit und phasenbezogene Skripte, die Cadence Innovus (und ggf. angebundene Tools) automatisiert ansteuern. Jede Phase lädt zunächst allgemeine Einstellungen über *Always-Source*-Blöcke (Technologie-, Timing- und Constraint-Setup), bevor die spezifischen Schritte der Phase ausgeführt werden. Im Folgenden werden die wesentlichen Tätigkeiten pro Phase zusammengefasst; die zugehörigen Befehle/Steps sind in der Tabelle am Ende dieses Abschnitts aufgeführt.

Initialisierung (init) Die Initialisierung bereitet das Design auf die physische Implementierung vor.

- Allgemeines Projekt-Setup: Pfade, Variablen und Skript-Bindungen; Innovus-Session wird konfiguriert.

- Einlesen der Gate-Level-Netzliste und ggf. vorhandener `DEF`-Informationen (Start-Floorplan, Pins).
- Verbinden der Power-Netze (`VDD`/`VSS`) und Laden technologie-/bibliotheksrelevanter Dateien.
- Einlesen der RAM-/Memory-Instanzen und zugehöriger Definitionen.
- Einfügen von Endcaps und Well-Taps zur Gewährleistung korrekter Randbedingungen und Substratanbindung.
- Erstellen der Power-Routing-Struktur (Power-Grid); Festlegen von `pg_keepout`-Zonen für Versorgung.
- Hinzufügen von *respin cells* (falls projektspezifisch erforderlich).
- Koordinaten-Normalisierung, z. B. *Shift Origin* auf die linke untere Ecke (*LL*).

Placement (place) In der Platzierungsphase werden logische Zellen positioniert und der Kontext für nachfolgende Optimierungen geschaffen.

- Laden der *Always-Source*-Einstellungen: u. a. OCV-Modelle, globale Constraints, `dontTouch`-Sets.
- Einstellen von *NDRs* (Non-Default Rules) für kritische Netze.
- Festlegen von *VT-Gruppen* gemäß UPC-Definitionen (Leckstrom/Timing-Zielsetzung).
- DFT-Vorplatzierung: Einfügen von *OCCs*, Haupt-DFT-Bereich; Vorplatzierung von *SIBs*, Isolationsmodulen und *MBIST*-Gruppen.
- Erneutes Setzen von `pg_keepout`-Zonen (Versorgungsnetz-Reserven).
- Einlesen der *Scan*-Definitionen; Schutz abuttierter Partition-Pins; Hinzufügen von Interface-Buffern.
- *Cell Padding* für Register (Abstandsregeln zur Entzerrung).
- (Re-)Connect der Power-Netze, falls erforderlich; *cgate*-Platzierung (Clock-Gating-Zellen).
- Definition von *Path Groups* für die Timing-Analyse und Optimierungspriorisierung.
- Platzierung und kombinierte Optimierung via `place_opt_design`.
- *save design*.

Clock Tree Synthesis (CTS) Die CTS erzeugt und optimiert den physikalischen Taktbaum.

- Laden der *Always-Source*-Einstellungen (CTS-spezifische Parameter).
- `ccopt` Konfiguration: Clock-Domain-Spezifikation, Puffer-/Inverter-Auswahl, Skew-/Latency-Ziele.
- Ausführen der Taktbaum-Erzeugung und -Optimierung mit `clock_opt_design`.

- Entfernen vorher gesetzter *cell padding* (sofern nicht mehr erforderlich).
- *Hold*-Optimierung zur Einhaltung kurzer Pfade (Setup/Hold-Balance).
- *save design*.

Routing (route) Das Routing verbindet die Netze über die verfügbaren Metall-Layer unter Einhaltung der Designregeln.

- Laden der *Always-Source*-Einstellungen; Setzen eines Genauigkeit/Laufzeit-Tradeoffs (Routing-Strategie).
- (Re-)Connect der Power-Netze; Feinanpassung der *NDRs* für ausgewählte Netze.
- Kombination aus globalem und detailliertem Routing über `route_opt_design`.
- *ADDCAP*: Kapazitive Ergänzungen/Decaps zur Stabilisierung der Versorgung und Reduktion von IR-Drop/Noise.
- *save design*.

Signoff (signoff) Die Signoff-Phase validiert Herstellbarkeit, Timing und Zuverlässigkeit und bereitet die Übergabe an die Fertigung vor.

- RC-Extraktion (`extract_rc`) für akkurate parasitäre Modelle.
- *Static Timing Analysis* und Abschluss-Timing (`signoff_timing`); ggf. ECO-Schleifen.
- Design Rule Checks (`signoff_dr`) und ergänzende DFM-Prüfungen.
- Optional: Power-/EM-Analysen (IR-Drop, Elektromigration) je nach Projekt-Setup.
- Export der finalen Daten (z. B. `write_gds` für GDSII).

Tabelle 1: Zuordnung zentraler Befehle zu den Phasen

Phase	**Wichtige Befehle**
Initialisierung (init)	`read_netlist`, `read_def`, `create_power_nets`, `pg_keepout`
Placement (place)	`place_opt_design`, `pg_keepout`
CTS	`clock_opt_design`
Routing (route)	`route_opt_design`
Signoff (signoff)	`<placeholder>`

5 Fazit und Ausblick

Die vorliegende Ausarbeitung hat den physischen Design-Flow von integrierten Schaltungen sowie die Rolle von Cadence Innovus und dem EDE-Tool beleuchtet. Dabei wurde gezeigt, wie die einzelnen Phasen – vom Floorplanning über Placement und Clock Tree Synthesis bis hin zu Routing und Signoff – ineinandergreifen und durch moderne EDA-Werkzeuge automatisiert werden können. Innovus bietet in Kombination mit dem EDE-Framework eine leistungsfähige Plattform, um komplexe Designs effizient umzusetzen und die Qualität durch konsistente Prozesse sicherzustellen.

Gleichzeitig ist zu betonen, dass diese Arbeit bewusst auf einer hohen Abstraktionsebene geblieben ist. Detaillierte Aspekte wie die zugrunde liegenden Optimierungsalgorithmen, komplexe Timing- und Power-Analysen sowie die vollständige Parametrisierung der Tool-Kommandos wurden nicht behandelt. Ebenso fehlen praktische Ergebnisse oder Benchmarks, da der Fokus auf dem konzeptionellen Verständnis des Design-Flows lag.

Für eine weiterführende Vertiefung empfiehlt sich die Auseinandersetzung mit den Optimierungsstrategien innerhalb der einzelnen Phasen, insbesondere Timing-Closure, Low-Power-Methoden und Multi-Corner-Analysen. Darüber hinaus bieten Themen wie Design-for-Test, ECO-Flows und der Einsatz von KI-gestützten Tools (z. B. Cadence Cerebrus) spannende Perspektiven für die Zukunft. Eine praktische Arbeit mit realen Projektdaten und die Teilnahme an spezialisierten Schulungen sind sinnvolle nächste Schritte, um das hier vermittelte Grundlagenwissen in die Praxis zu übertragen.

Literatur

Chakravarthi, Veena S. (2022). *SoC Physical Design: A Comprehensive Guide*. eng. 1st ed. Cham: Springer International Publishing AG. ISBN: 978-3-030-98112-9.