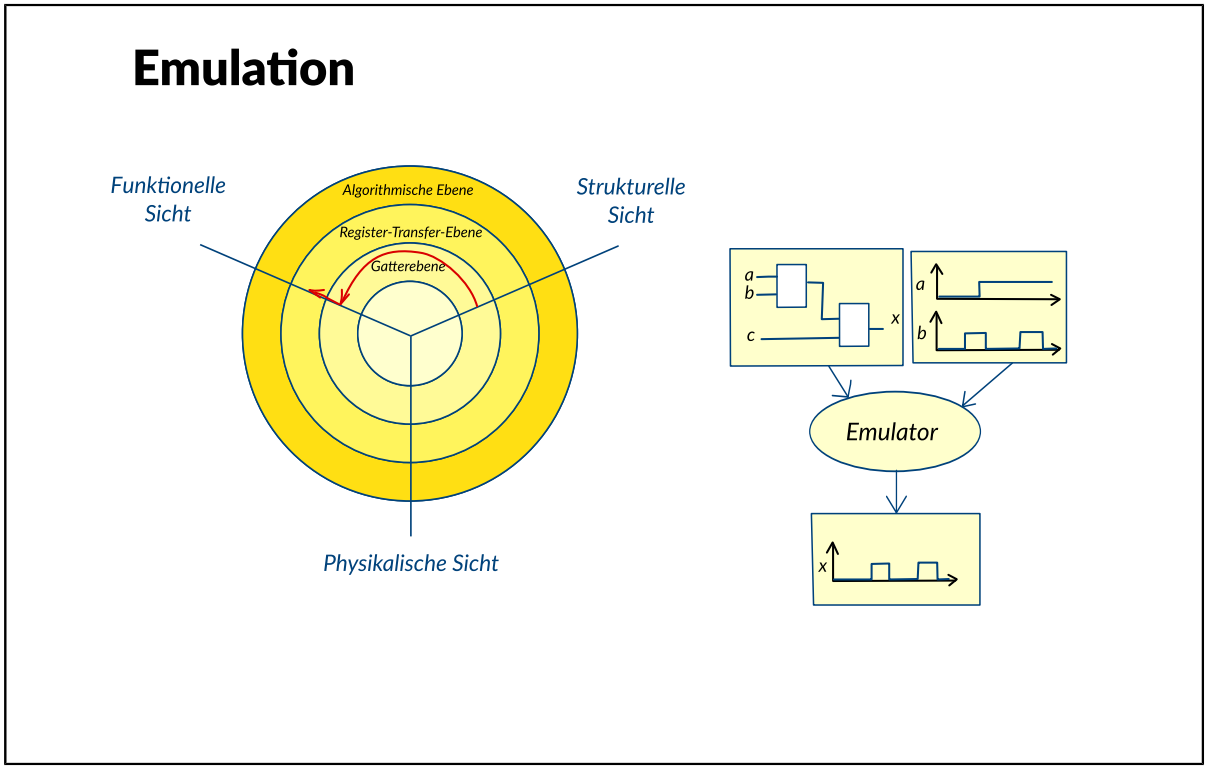
Electronic Design Automation (EDA)

Emulation

Emulation
Emulationskonzept
Schaltungsabbildung
Lookup-Tables (LUT)
CLB mit 2 Lookup-Tables (LUTs)
Rekonfigurierbare Netze
Partitionierung
Anzahl der Pins
Eigenschaften der Emulation
Simulation/Emulation/ASIC
Emulationsbetriebsarten
Vector Debug Modus
In-Circuit-Emulation

Date of generation: 26.10.2026

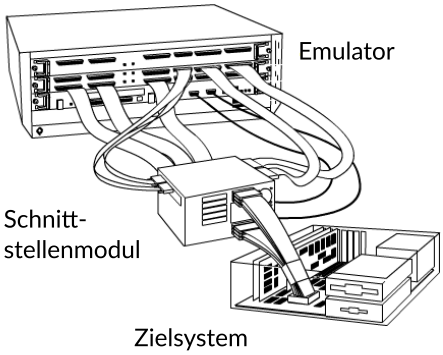
Emulation: Emulation



Moderne Schaltungen sind so komplex, dass man sie über sehr viele Taktschritte laufen lassen muss, um sie hinreichend zu simulieren. Bei einer Simulation auf Gatterebene könnte das Monate in Anspruch nehmen. Hinzu kommt die mangelnde Anbindung an die Peripherie. Jede Schaltung tauscht Daten mit ihrer Umgebung aus. Dazu muss nicht nur die Funktion der Schaltung für sich, sondern auch die Kommunikation mit ihrer Umgebung, der so genannten Peripherie, getestet werden. Fast immer ist die simulierte Schaltung aber zu langsam, um die Datenströme der Peripherie verarbeiten zu können und um schnell genug Steuersignale auszusenden. Um diese Nachteile zu umgehen, wendet man das Verfahren der Emulation an. Dabei wird das Verhalten einer Schaltung durch eine andere Schaltung nachgebildet. Das Verfahren ist also hardware-, nicht softwarebasiert. Dazu wird eine Emulator genannte Maschine verwendet, die statt der sonst üblichen festverdrahteten Schaltungen aus Einheiten rekonfigurierbarer Logik besteht. Der Emulator wird so konfiguriert, dass seine Funktion dem der zu emulierenden Schaltung entspricht; er verhält sich nach außen also wie die Schaltung selbst. Emulation findet grundsätzlich auf der Gatterebene statt, da die Grundelemente des Emulators einfache logische Funktionen realisieren.

Emulation: Emulationskonzept

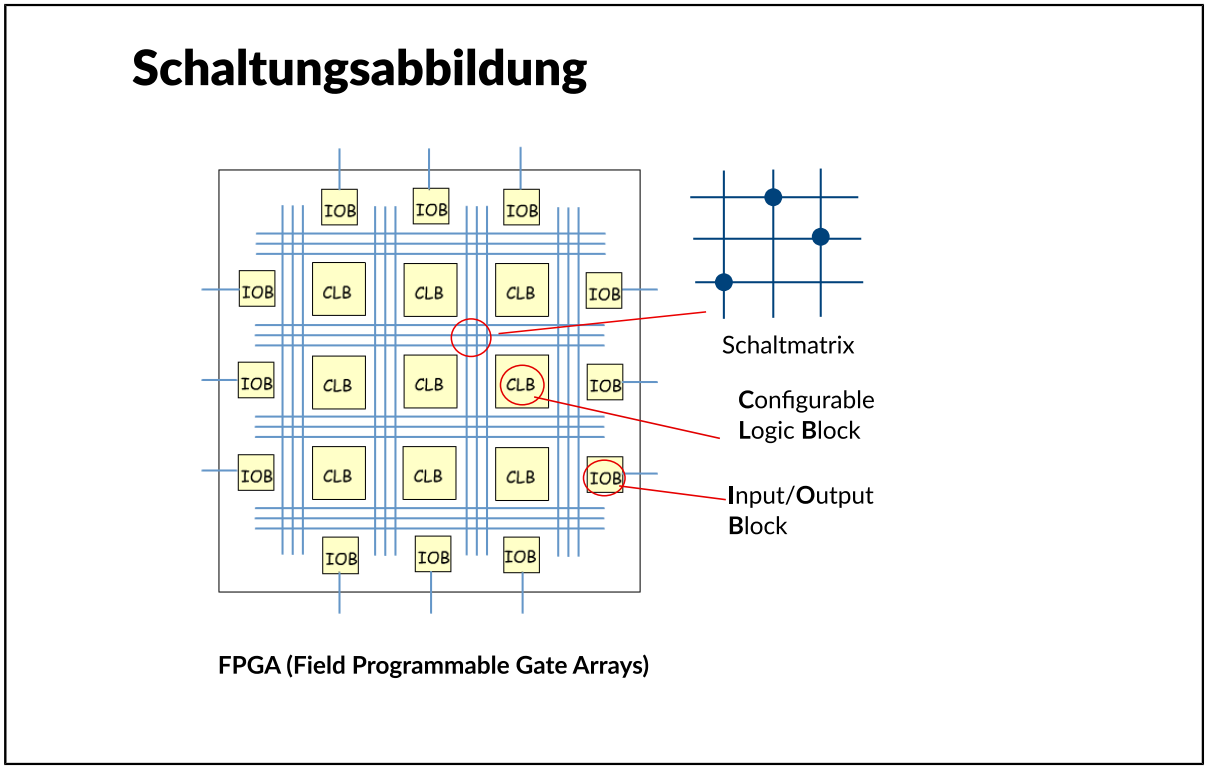
Emulationskonzept



- Virtueller Chip in programmierbarer Hardware
- In-System-Emulation möglich
- Ca. 1000fach schneller als Simulation
- Hoher Anschaffungspreis

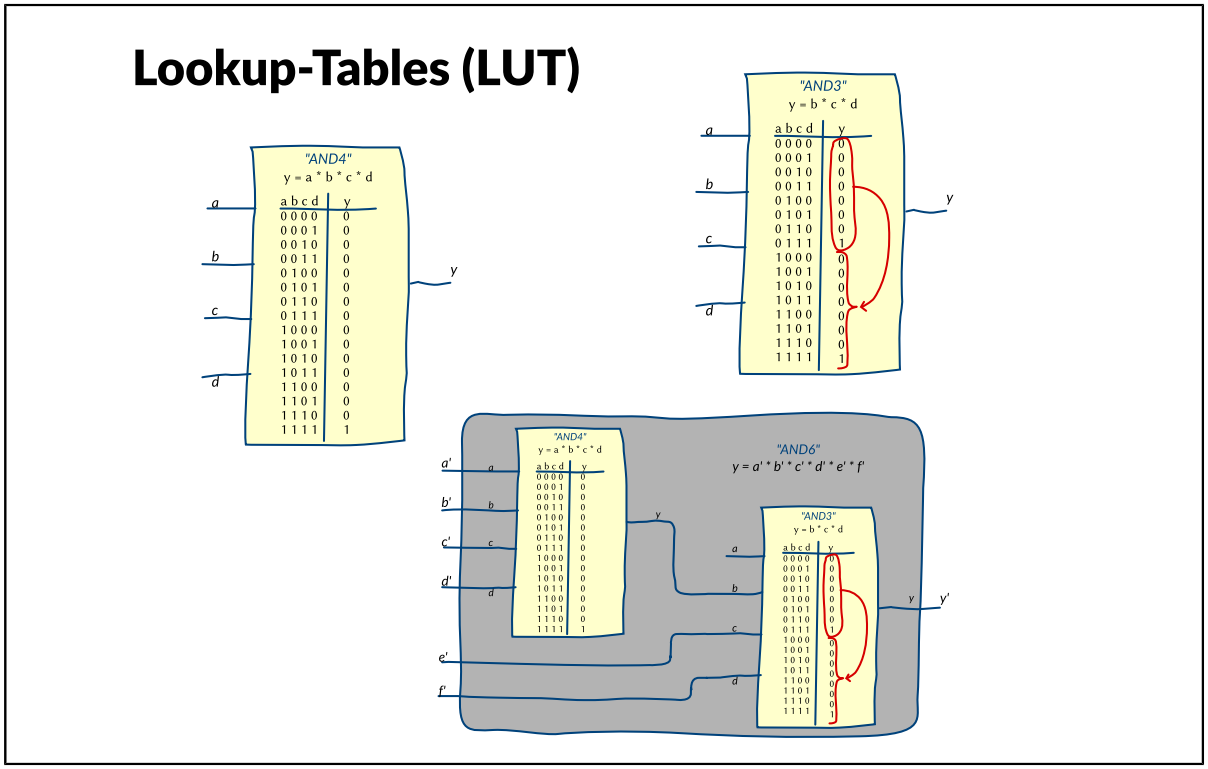
FPGAs (Field Programmable Gate Arrays) sind die Basisbauelemente für eine Emulation. Größere Emulationssysteme bestehen aus der Zusammenschaltung von Dutzenden von einzelnen FPGAs. Das funktionelle Modell des Chips wird durch Struktursynthese, Technologieabbildung und Partitionierung auf das Emulationssystem abgebildet. Es entsteht eine Art virtueller Chip in programmierter Hardware. Die Emulation einer Schaltung ist in der Regel langsamer als der fertige Chip, aber sie ist ca. eintausendmal schneller als eine Simulation. Die Geschwindigkeit reicht aus, um die meisten Chips schon vor ihrer Fertigung in ihrem späteren Zielsystem zu testen. Dies stellt einen enormen Vorteil gegenüber der Simulation dar, da das Zusammenwirken eines Chips mit dem Zielsystem in Software meist nur sehr schwer dargestellt werden kann. Ein Nachteil der Emulationssyteme ist der sehr hohe Anschaffungspreis von einigen 100.000 Euro bis hin zu mehreren Mio. Euro, da zudem wegen der steigenden Designkomplexität alle paar Jahre eine Neuanschaffung ansteht.

Emulation: Schaltungsabbildung



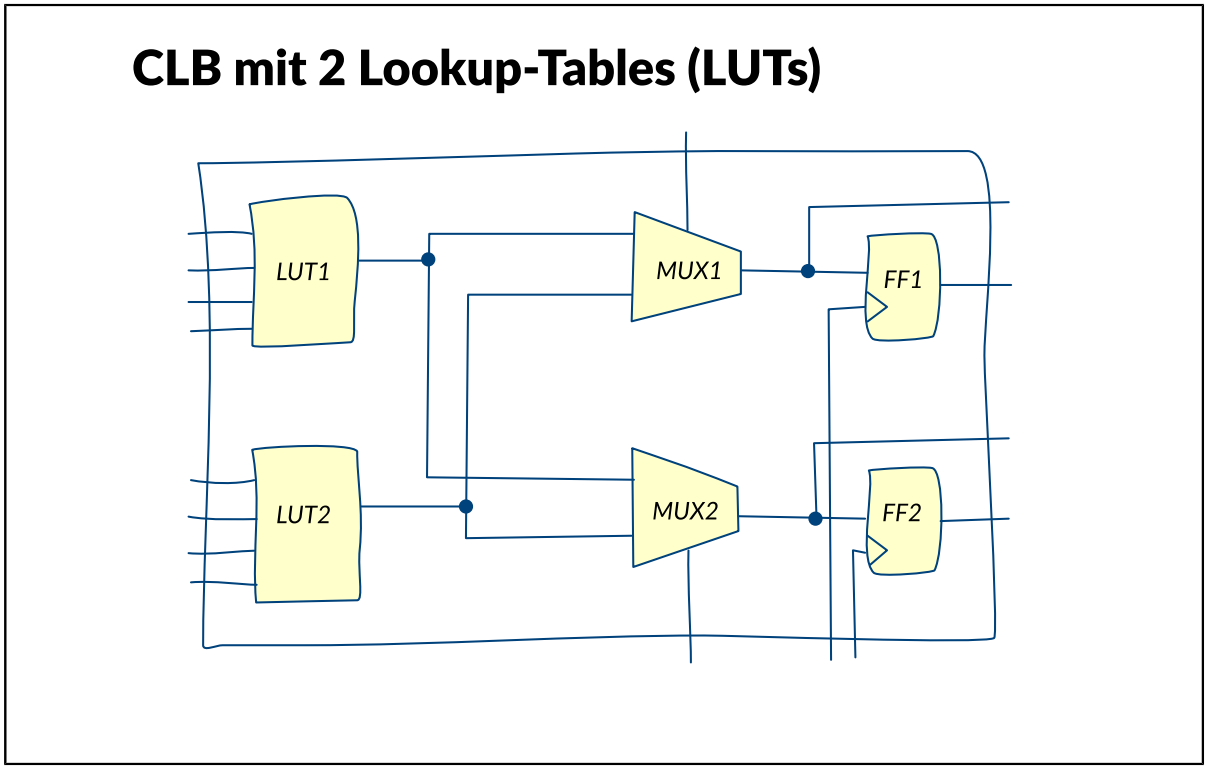
Ein Emulator besteht üblicherweise aus einer Matrix von so genannten FPGAs (Field Programmable Gate Arrays), die untereinander mit frei konfigurierbaren Leitungen über Schaltmatrizen verbunden werden können. Die FPGAs sind Bausteine, die rekonfigurierbare Logik enthalten. Eine Matrix so genannter CLBs (Configurable Logic Blocks) wird von einem Netz rekonfigurierbarer Leitungen durchzogen, mit denen die CLBs untereinander und mit den Ein- und Ausgangspins des FPGAs verbunden werden können. Die Abbildung einer Schaltung auf einen Emulator stellt eine Technologieabbildung dar. Ausgangspunkt ist die Gatterebene in funktionaler Sicht, in der noch keine Vorentscheidung über die zu verwendende Technologie getroffen ist. Der Mapping-Schritt bildet einzelne oder auch Gruppen von Logikgattern auf CLBs ab. CLBs werden in der Regel als sogenannte Lookup-Tables (LUT) realisiert.

Emulation: Lookup-Tables (LUT)



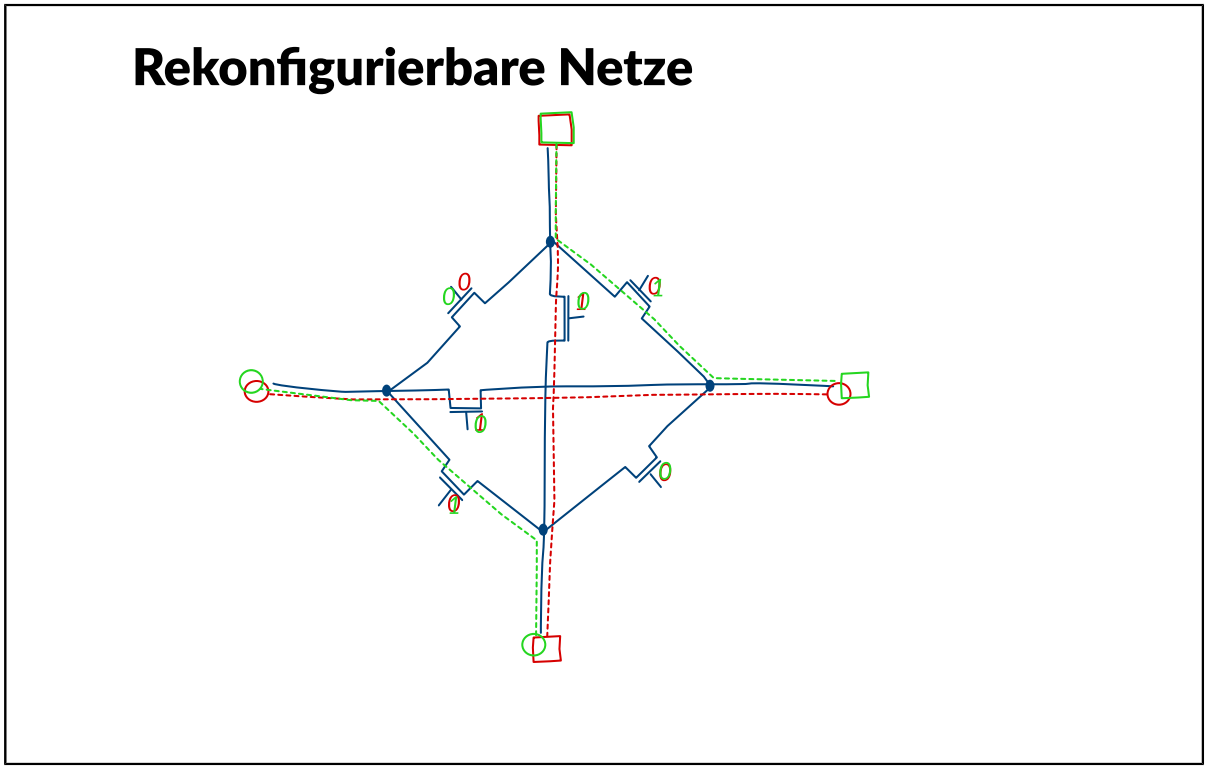
Eine LUT ist die Implementierung einer Wahrheitstabelle als Speicher. Betrachtet man die Eingänge der LUT als Adressleitungen des Speichers, kodiert jede Kombination von Eingangssignalen eine Speicheradresse. Am Ausgang der LUT liegt immer der Inhalt der ausgewählten Speicherzelle an. Dieser entspricht dem Wert der logischen Funktion für diese Eingangsbelegung. Durch Festlegen der Speicherinhalte lässt sich die logische Funktion, die die LUT modelliert, frei bestimmen. Um ein Logikgatter auf eine LUT abzubilden, wird die Wahrheitstabelle des Gatters auf die LUT übertragen. Hat das Logikgatter weniger oder gleich viele Eingänge wie die LUT, ist die Abbildung direkt möglich. Bleibt ein Eingang unbelegt, muss die Wahrheitstabelle doppelt in der LUT abgelegt werden, so dass das Ergebnis am Ausgang vom Wert des nicht belegten Signals unbeeinflusst bleibt. Sieht man die LUT als Speicher, halbiert sich der verwendete Adressraum. Hat ein Logikgatter mehr Eingänge als eine LUT, muss es auf mehrere LUTs aufgeteilt werden. Diese enthalten dann jeweils einen Teil der logischen Funktion des Gatters und bilden zusammengenommen die Funktion korrekt ab.

Emulation: CLB mit 2 Lookup-Tables (LUTs)



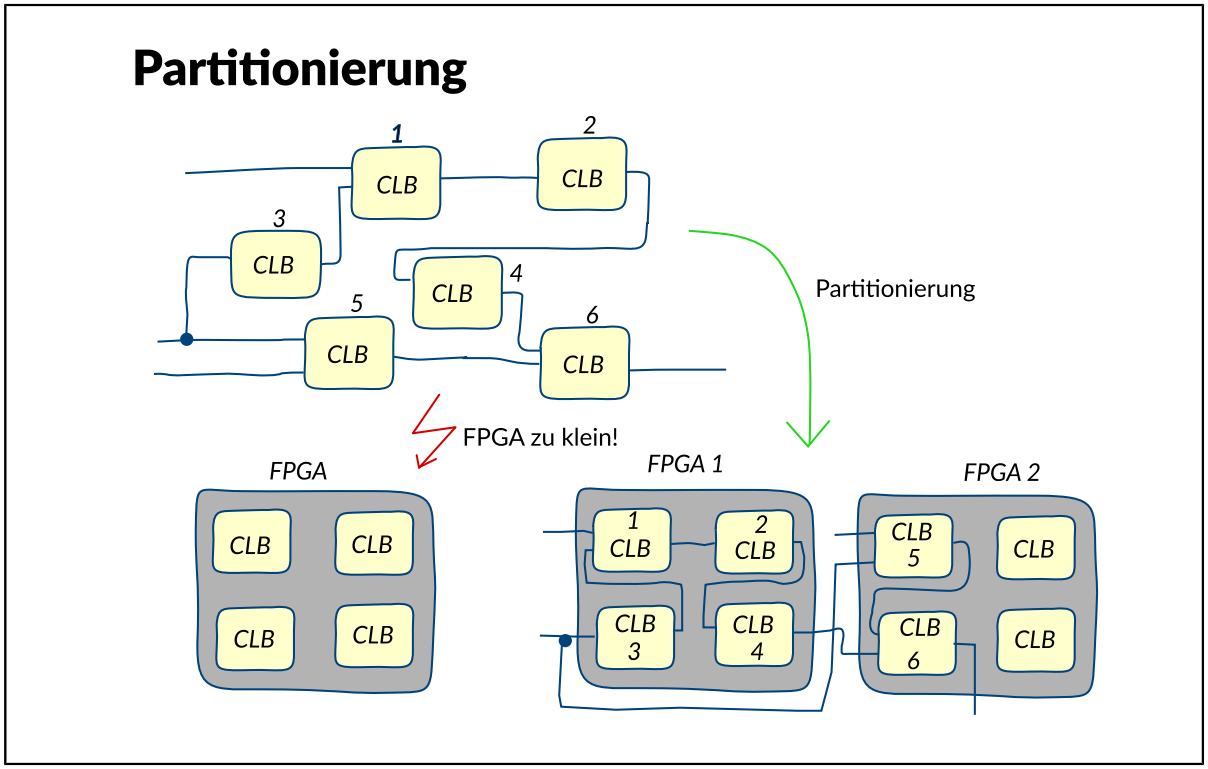
In der Praxis haben LUTs meist vier Eingänge und einen Ausgang, verfügen also über 16 Eingangssignalkombinationen. Neben einer oder oft auch zwei LUTs enthält ein CLB auch eine entsprechende Anzahl Flip-Flops. Diese sind notwendig, um sequentielle Logik modellieren zu können. Der Ausgang der LUTs kann dazu jeweils auf ein Flip-Flop geführt werden. Die Abbildung zeigt die Ausführung eines in der Praxis gebräuchlichen CLBs mit zwei LUTs, zwei Flip-Flops und der sie verbindenden Logik.

Emulation: Rekonfigurierbare Netze



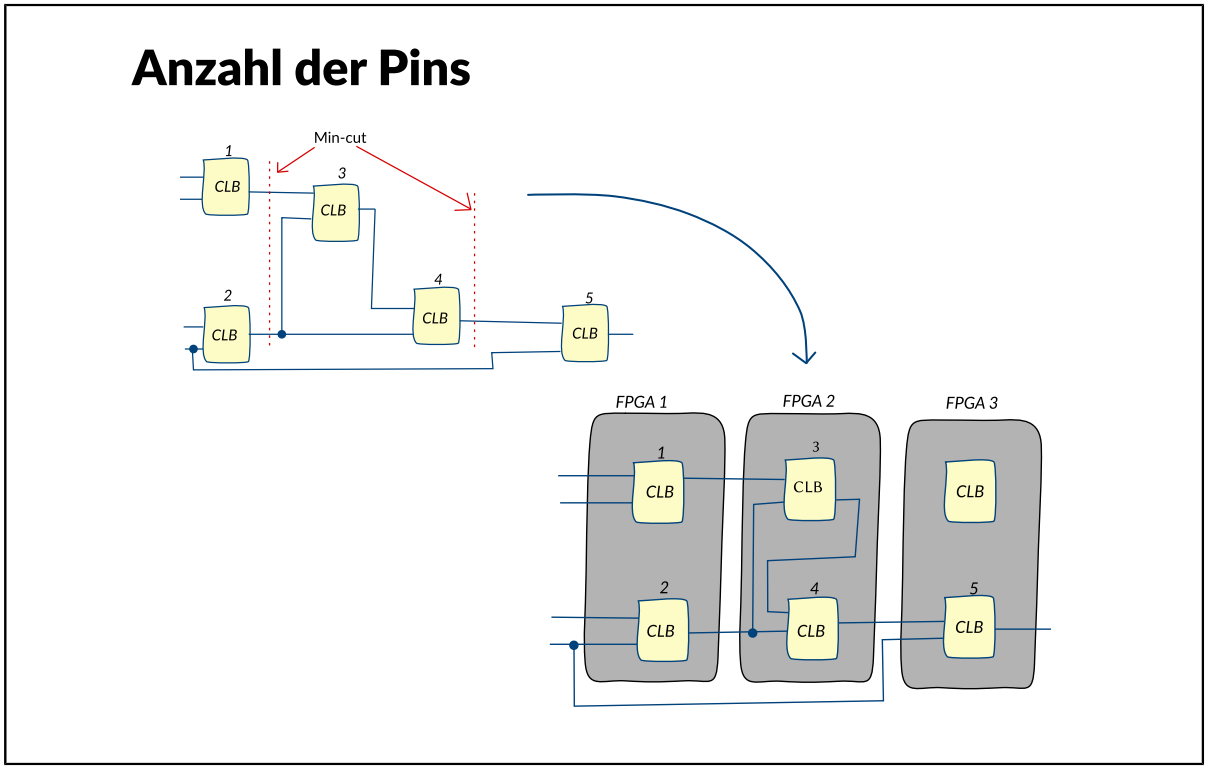
Um die rekonfigurierbaren Logikblöcke untereinander verbinden zu können, sind rekonfigurierbare Netze notwendig. Dazu werden Verdrahtungskanäle zwischen den CLBs vorgesehen, in die Bündel von Signalleitungen gelegt werden. An den Stellen, wo Verbindungen zu den Ein- und Ausgängen der am Kanal gelegenen CLBs geschaffen werden müssen und vor allem an den Kreuzungspunkten der senkrecht zueinander verlaufenden Kanäle befinden sich Schaltmatrizen, die eine freie Konfiguration der Verbindungen ermöglichen. An jeder Stelle, wo zwei Leitungen sich kreuzen, werden so genannte Passtransistoren zwischen den Leitungen angebracht. Zudem werden die Leitungen selbst auch jeweils mit einem Passtransistor unterbrochen. Dadurch treffen sich vier getrennte Leitungen miteinander, von denen jede mit jeder über einen Passtransistor verbunden werden kann. Auf die Art können Überkreuzungen, Abzweigungen und Richtungswechsel realisiert werden. Die Konfiguration eines Knotenpunkts wird dadurch vorgenommen, dass man festlegt, welche Passtransistoren sperren und welche ein Signal durchschalten sollen. Jedes Gate eines Passtransistors ist mit einem Latch verbunden, das den Konfigurationswert speichert. Durch die Passtransistoren werden Leitungen von einem CLB zu einem anderen aus mehreren Leitungsteilen zusammengesetzt. Der Anzahl an Leitungen, die in einem Leitungskanal untergebracht werden können, sind jedoch Grenzen gesetzt. Es kommt daher in der Praxis vor, dass nicht alle CLBs eines FPGAs benutzt werden können, da bereits alle Leitungsresourcen für die Verbindungen anderer CLBs aufgebraucht wurden. Meist sind solche Engpässe nur auf einen Bereich eines FPGAs beschränkt. Analog zum Straßenverkehr spricht man auch von einer "Signal Congestion" (congestion (engl.) = Stau). Dann lässt sich Abhilfe meist dadurch schaffen, dass man die zu implementierenden Funktionen anders auf die CLBs im FPGA verteilt und damit die Leitungsdichte in dem Bereich verringert.

Emulation: Partitionierung



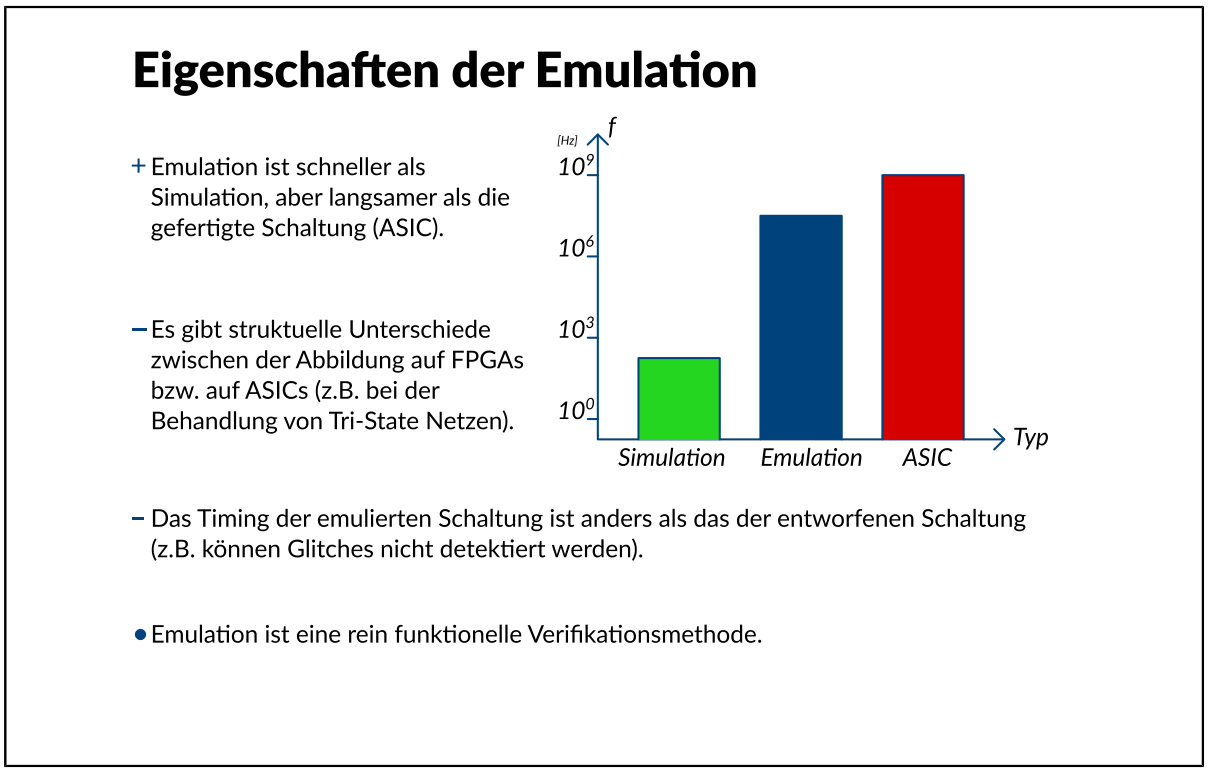
In vielen Fällen ist eine Schaltung zu groß, um sie vollständig auf ein einzelnes FPGA abbilden zu können. Dafür kann es mehrere Gründe geben: 1.Es gibt nicht genug CLBs im FPGA, um die Logik abzubilden. 2.Es sind mehr Leitungen zur Verbindung der CLBs nötig, als das FPGA insgesamt oder in einem bestimmten Bereich hat. 3.Die abzubildende Schaltung erfordert mehr Ein- und Ausgangspins, als das FPGA zur Verfügung stellen kann. In solchen Fällen muss die Schaltung auf mehrere FPGAs aufgeteilt werden. Dieser Vorgang heißt Partitionierung. Dabei sind vor allem zwei Randbedingungen zu beachten: die Länge des kritischen Pfads und die Anzahl der Ein- und Ausgänge. Jedes Schaltungselement und jede Leitung verzögern die Signale, die sie durchlaufen. Besonders groß ist die zusätzliche Verzögerung, wenn das Signal von einem FPGA zu einem anderen geführt wird. Die Leitungen außerhalb der FPGAs sind im Vergleich zu den internen Leitungen sehr lang und haben eine sehr große Kapazität. Beides erhöht die Verzögerungszeiten. Es muss deshalb darauf geachtet werden, dass der kritische Pfad und andere Pfade mit ähnlich großer Verzögerungszeit möglichst nicht über mehrere FPGAs führen, da sonst durch die zusätzliche Verzögerung die mögliche Taktfrequenz der Schaltung deutlich herabgesetzt wird.

Emulation: Anzahl der Pins



Ein FPGA hat nur eine begrenzte Anzahl an Pins, über die Eingangs- und Ausgangssignale geführt werden können. Wenn man eine Schaltung auftrennt, um sie auf mehrere FPGAs abzubilden, braucht man für jedes Signal, das von einem FPGA zu einem anderen führt, jeweils einen Pin an beiden FPGAs. Trennt man die Schaltung nun an einer ungünstigen Stelle auf, kann es passieren, dass man wesentlich mehr Signale aus einem FPGA zu- und abführen will, als dieses Pins hat. Dann kann die gewählte Partitionierung nicht durchgeführt werden. Aus diesem Grund sucht man bei der Aufteilung einer Schaltung nach dem so genannten Min-Cut. Das ist die Trennung einer Schaltung in zwei Teile, die die wenigsten Verbindungen zwischen den beiden Teilen aufweist.

Emulation: Eigenschaften der Emulation



Die Emulation bietet offenkundig einige Vorteile gegenüber der Simulation. Trotzdem existieren beide nebeneinander, denn die Emulation hat auch einige Nachteile. - Emulation ist schneller als Simulation, aber langsamer als die gefertigte Schaltung. - Das Timing der emulierten Schaltung ist anders als das der entworfenen Schaltung (z.B.können Glitches nicht detektiert werden). - Es gibt strukturelle Unterschiede bei der Abbildung auf FPGAs bzw. auf ASICs (z.B. bei der Behandlung von Tri-State-Netzen) . - Emulation ist eine rein funktionelle Verifikationsmethode. Während Simulatoren heute mit Taktraten zwischen einigen Hertz und wenigen Kilohertz laufen, werden Emulatoren üblicherweise im Bereich von einem bis zu zweihundert Megahertz betrieben. Trotzdem sind sie immer noch langsamer als die gefertigte Schaltung. Das liegt an den zusätzlichen Verzögerungszeiten, die ein Emulator durch die rekonfigurierbare Logik hat. Dazu tragen sowohl die rekonfigurierbaren Netze mit ihren Passtransistoren als auch die CLBs mit ihren Konfigurationselementen bei. Besonders groß werden die Verzögerungszeiten, wenn die Schaltung zusätzlich auf mehrere FPGAs partitioniert werden muss. Um eine Schaltung schon vor der Fertigung verifizieren zu können, ist es wichtig, ihr späteres Zeitverhalten zu kennen. Dazu müssen die Verzögerungszeiten der einzelnen Gatter und nach Möglichkeit auch der Signalleitungen bekannt sein. Dann kann in der Schaltungssimulation die Verzögerung durch die einzelnen Schaltungskomponenten berücksichtigt und das zeitliche Verhalten der Schaltung realistisch nachgebildet werden. In der Emulation ist dies höchstens sehr eingeschränkt möglich. Da die Schaltung direkt auf die Hardware des Emulators abgebildet wird, arbeitet sie auch mit deren Verzögerungen. Ein CLB hat aber nur in Ausnahmefällen die gleiche Verzögerung wie das Gatter, was darauf abgebildet wurde, in einem ASIC. Genauso verhält es sich mit den Leitungen. Eine Emulation wird sich bei richtiger Wahl der Taktrate taktgenau verhalten. Effekte, die in ihrer Größenordnung darunter liegen, können nicht von einer Emulation abgeleitet werden. Dazu gehören zum Beispiel Glitches. FPGAs und damit die aus ihnen aufgebauten Emulatoren sind darauf ausgelegt, binäre Logik abzubilden. Das führt zu mehreren Einschränkungen: Heutige Schaltungen enthalten häufig so genannte dreiwertige Logik (engl. Tristate-Logic). Diese Art von Logik ist insbesondere für den Aufbau von Bussystemen geeignet. An einem solchen Bus sind die Ausgänge von mehr als einem Gatter angeschlossen. Normalerweise würden diese Ausgänge häufig verschiedene logische Pegel treiben und damit elektrische Kurzschlüsse verursachen. Die Tristate-Logik schafft hier Abhilfe. Nur ein Gatter treibt das Tristate-Netz, alle anderen sind im hochohmigen Zustand, wodurch der Signalpegel auf dem Netz eindeutig bleibt. Eine solche Anordnung stellt für einen Emulator allerdings ein Problem dar. Lookup-Tables müssen immer einen binären Wert

an ihrem Ausgang treiben. Multiplexer müssen immer eines ihrer Eingangssignale zum Ausgang durchleiten. Eine direkte Abbildung von Tristate-Logik auf Emulatoren ist also nicht möglich. Um dieses Problem zu umgehen, verfügen einige FPGAs über zusätzliche Tristate-Logik-Resourcen. Damit ist eine direkte Abbildung möglich, jedoch wird der Abbildungsvorgang komplizierter, da das Abbildungswerkzeug die verschiedenen Logiktypen verwalten und verknüpfen muss. Eine andere Möglichkeit besteht darin, die Tristate-Logik in binäre Logik zu transformieren. Das ist dann möglich, wenn man einen Defaultwert vorgibt, den ein Tristate-Netz haben soll, wenn es von gar keinem Gatterausgang getrieben wird. Die resultierende binäre Logik ist umfangreicher als die ursprüngliche dreiwertige, lässt sich aber auch auf gewöhnliche FPGAs abbilden . Für diesen Fall wird besonders deutlich, dass eine Schaltung auf einem Emulator mit einer gleichwertigen als ASIC gefertigten Schaltung nichts weiter gemein hat als die gleiche Funktion. Durch Einführung zusätzlicher logischer Pegel zwischen "high" und "low" können in der Schaltungssimulation die Auswirkungen von nicht-digitalen Effekten wie unterschiedliche Treiberstärken modelliert werden. Bei Verwendung eines Emulators ist das nicht möglich.

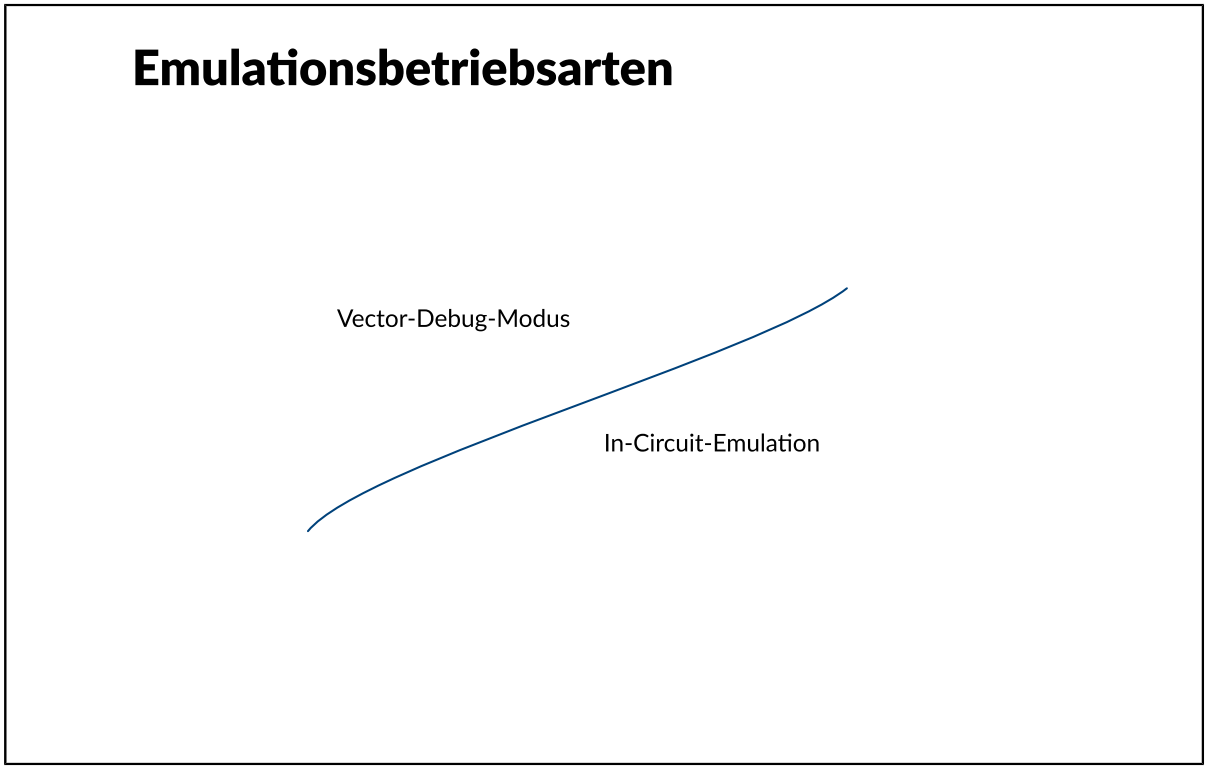
Emulation: Simulation/Emulation/ASIC

Simulation / Emulation / ASIC

	Simulation	Emulation	ASIC
Zieltechnologie	Verhaltensmodell	Rekonfigurierbare Logik (FPGA)	Transistorlogik (z.B. in CMOS)
Geschwindigkeit	langsam	schnell	sehr schnell
Verzögerungen	modelliert	eigene	real
Tristate-Netze	modelliert	eingeschränkt	real
Beobachtbarkeit interner Signale	voll	eingeschränkt	sehr aufwendig
In-Circuit-Betrieb	nein	ja	ja

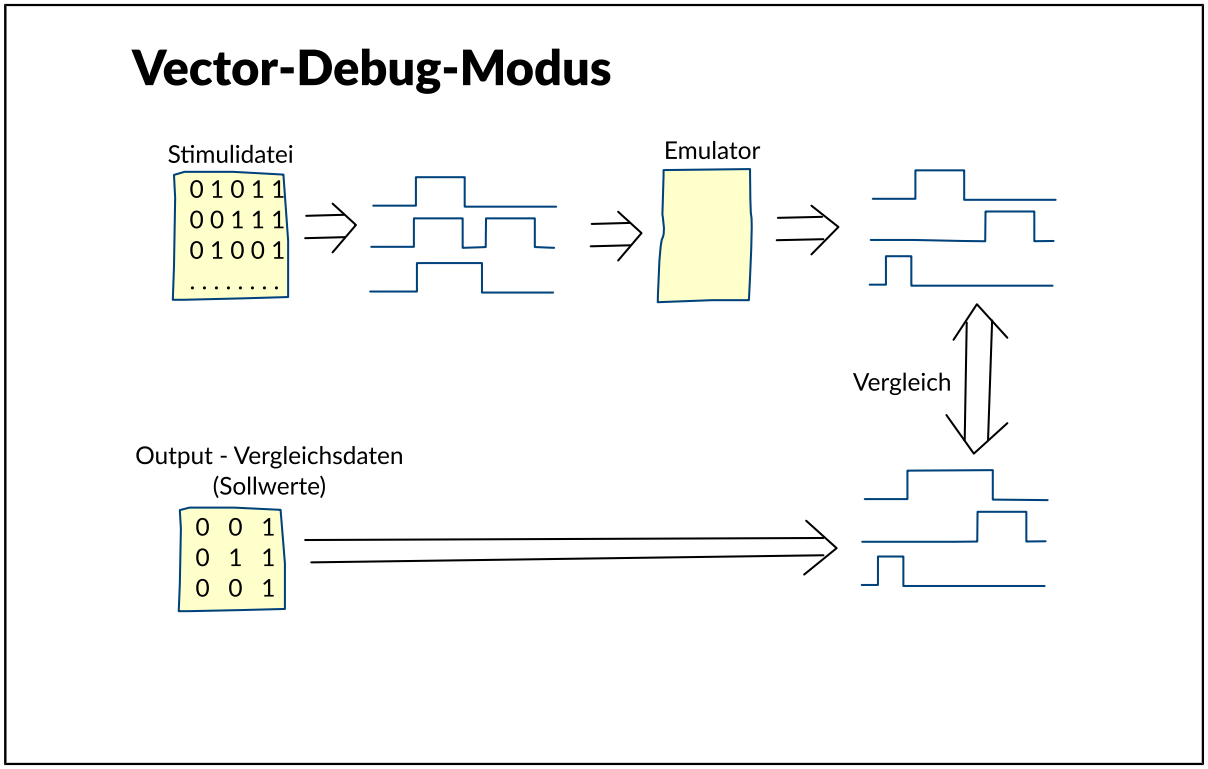
Die Tabelle zeigt eine Gegenüberstellung der Eigenschaften von Simulation, Emulation und gefertigtem ASIC einer Schaltung.

Emulation: Emulationsbetriebsarten



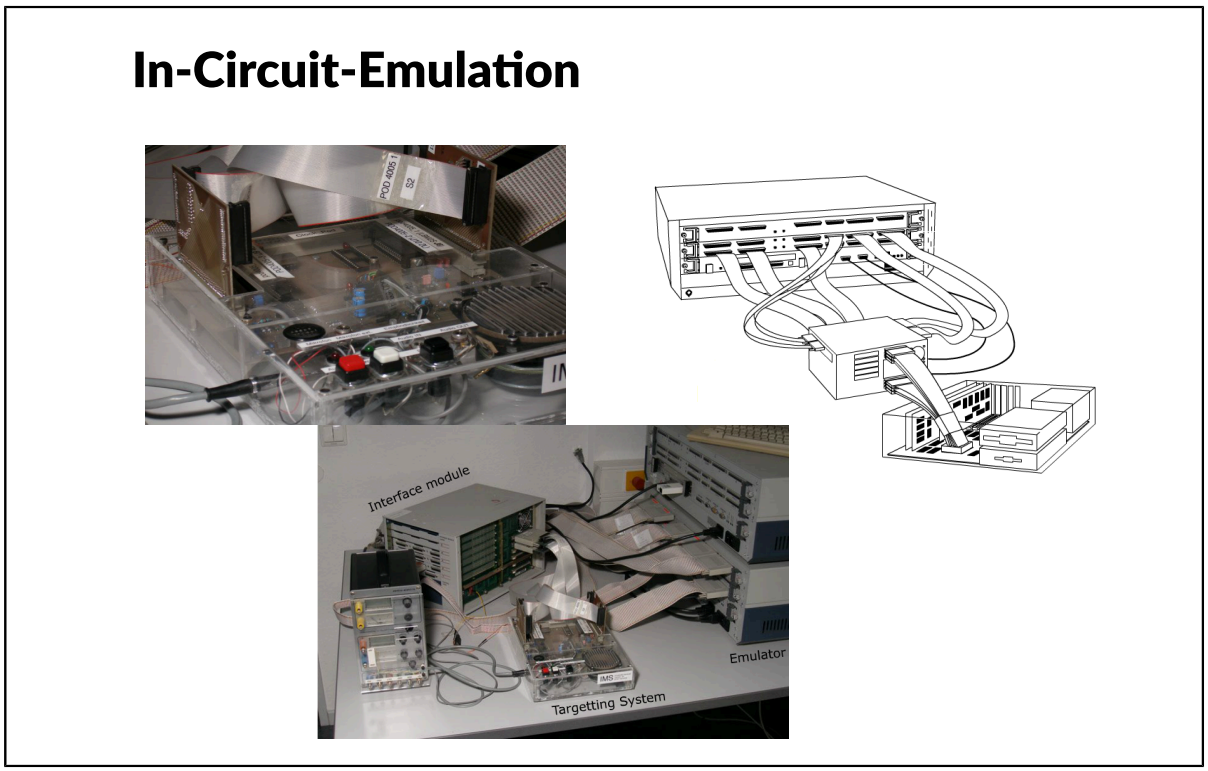
Ein Emulator kann in zwei Betriebsarten eingesetzt werden: - Vektor-Debug-Modus ("beschleunigte Simulation") - In-Circuit-Modus

Emulation: Vector Debug Modus



Im Vector Debug Mode wird die auf den Emulator abgebildete Schaltung genau wie bei einer Simulation mit festgelegten Stimuli angesteuert. Oftmals ist die Betrachtung der Schaltungsausgänge allein unzureichend, um das Verhalten der Schaltung nachvollziehen zu können. Darum bieten viele Emulationssysteme die Möglichkeit, in begrenztem Umfang auch Signale aus dem Inneren der emulierten Schaltung abzugreifen und zu speichern. In der Simulation sind dagegen alle Signale zu jedem Zeitpunkt zugänglich. Bei einem gefertigten ASIC ist dies nur mit einem sehr hohen Aufwnd möglich.. Dieser Emulationsmodus wird häufig als "beschleunigte Simulation" zur Verifikation der Schaltungsfunktion eingesetzt. Die Ausgangssignale der Emulation werden dazu mit vom Schaltungsentwickler festgelegten Sollsignalen verglichen, die seinen Erwartungen an die Funktion der Schaltung entsprechen. Stimmen die Emulationsergebnisse mit den Sollwerten überein, hat die Schaltung das vom Entwickler gewünschte Verhalten, ihre Funktion ist verifiziert.

Emulation: In-Circuit-Emulation



Bei der In-Circuit-Emulation wird der Emulator direkt an das Zielsystem angeschlossen, das für die zu emulierende Schaltung vorgesehen ist. Es ersetzt also den meist noch nicht gefertigten ASIC. Die Eingangsstimuli des Emulators sind die realen Signale des Systems. Es wird ebenso direkt von den Ausgangssignalen des Emulators angesteuert. Ein Problem stellt meist die Emulationsgeschwindigkeit dar. Obwohl sie der einer als ASIC gefertigten Schaltung nahekommt, liegt sie doch praktisch immer darunter. Die Peripherie ist aber meist für einen eng begrenzten Taktbereich entworfen. In dem Fall kommt es zu Fehlverhalten, wenn der Emulator die Peripherie mit zu geringer Geschwindigkeit ansteuert oder Daten von dort nicht schnell genug verarbeiten kann. Daher muss die Peripherie für eine In-Circuit-Emulation meist heruntergetaktet oder anderweitig neu mit der Schaltung synchronisiert werden. Trotzdem ist die In-Circuit-Emulation zur Verifikation der Kommunikation einer Schaltung mit ihrem Umfeld praktisch unerlässlich, gerade wenn weitere komplizierte Komponenten angesteuert werden, deren Verhalten ebenfalls nicht vollständig analytisch zu erfassen ist. Mit der In-Circuit Emulation wird also ebenso wie mit der Vector-Debug-Emulation ein funktioneller Test der Schaltung durchgeführt.