

Zusammenfassung: Digital Design

Basics:

Vorteile von Digitalisierung:

- speicherbarkeit
- programmierbarkeit
- störresistenz

SR-latch / D-Latch:

- zwei nor gatter
- nächste ausbaustufe: SR mit enable
- nächste ausbaustufe: $(s \neq r) \rightarrow D\text{-Latch}$

latch vs flip-flop:

- latch: pegel-gesteuert
- flip-flop: flanken-gesteuert

D-FLip-Flop:

- zwei D-Latches hintereinander mit jeweils invertierter clock
- clock enable:
 - dont mess with the clock
- mux in datenpfad \rightarrow entweder d übernehmen, oder früheren ausgang
- Preset Clear:
 - asynchron
 - wichtig für initialisierung

Toggle-FF:

- D-FF bei dem negierter Ausgang immer zu D zurückgeführt wird
- kann verwendet werden um taktfrequenz zu halbieren

JK-FF:

- J: "Jump", K: "Keep"
- D-FF, Q und \bar{Q} werden zurückgeführt mit und-gattern und dann oder-gatter

Seriell-Parallel-Konverter:

- Reihe von D-FF
- an ersten wird serielles signal angelegt
- mit jeder flanke wird speicher eins weitergeschoben

Parallel-Seriell-Konverter:

- ebenfalls schieberegister

- am anfang werden D-FFs entsprechend gesetzt
- mit jedem takt wird ein wert rausgeschoben
- Muxe um zwischen Load und Shift zu schalten

LFSR (Linear Feedback Shift Register):

- schieberegister mit rückführung über parity-gatter
- dnf/knf mit kv-diag. üben

Asic-Fertigung

vorteile von ICs:

- billiger
- schneller
- stromsparender
- störsicherer
- billiger
- schlecht kopierbar

Moores Law: Komplexität verdoppelt sich alle 1.5 Jahre

Bestandteile:

- Verbindung: Poly-Si, Aluminium oder Kupfer
- Schalter: dotiertes Silizium
- Isolatoren: oxidiertes Silizium

Aufbau:

- Die ("Chip")
- Bonding
- Packaging

Silizium Dotieren

- Bor: p
- Phosphor: n

MOS-FET:

- Metall-Oxid-Halbleiter-Feldeffekttransistor
- p-dotiertes Substrat
- n-dotierte Wannen
- Isolator an Gate

Fertigungsprozess:

- Silicon Ingots
- Silicon Wafers
- Patterned Wafers
- Individual Dies

- Testing
- Packaging
- Testing (again)

Gehäuse - wozu:

- mechanischer schutz
- chemischer schutz
- Wärmeabfuhr
- besserer Zugang zu Kontakten

Moderne Formen:

- SoC (System on a Chip)
 - alle Komponenten befinden sich auf einem einzigen Chip
 - Komponenten heißen "IP-Module" → Intellectual Property
 - Vorteile: billig, klein, schnell
 - Nachteile: Komplexe Fertigung, nicht alles integrierbar
- SiP (System in Package)
 - mehrere Dies in einem Gehäuse
 - ⇒ leichter in Fertigung
 - Stacked Die Package ⇒ kürzere Verbindungen zwischen Chips
 - Through Silicon Via ⇒ Verbindungen gehen durch mehrere Schichten von Dies

Grenzen der Technologien:

- einzelne elektronen machen Unterschiede
- zu dünne Isolatoren ⇒ Tunnelströme
- Größe der Atome ⇒ Moleküle des Photolacks
- Wellenlänge des Lichts bei Belichtung
- Lichtgeschwindigkeit zu langsam
- Thermische Leitfähigkeit zu gering
- Ladevorgänge dauern zu lang

Transistor der Zukunft:

- Strained Silicon ⇒ Kristallgitter wird auseinandergezogen
- Gate aus Metall
- bessere Isolatoren
- beidseitiges Gate ⇒ 3d-Struktur, FinFET
- Silicon on Insulator
 - Substrat wird mit Siliconoxid überzogen um Störeffekte zu mindern

Grenzen der Komplexität:

- Aufwand für Testen wird zu groß
- "Design Crisis": Dinge können nicht schnell genug entwickelt werden
- Anzahl der Pins am Gehäuse
- Leistungsverbrauch

Sequenzielle Logik

FET:

- wird als Schalter verwendet, ist aber im Kern analog mit allen Nachteilen
- An Allen P-N-Übergänge bildet sich Sperrsicht
– Gleichgewicht zwischen elektrischer Kraft und Gitterkraft kann verändert werden \Rightarrow Bildung eines Kanals
- Gleichgewicht stark temperaturabhängig
- Schwellspannung über diverse Parameter einstellbar, zB dotierung oder Formfaktor \Rightarrow Treiberstärke
 - * Tradeoff zwischen Geschwindigkeit und Energieeffizienz

N- vs P-MOSFET:

- n-mosfet gut in Sourceschaltung \Rightarrow treibt 0-er gut
- p-mosfet gut in Sourcefolger \Rightarrow teibt 1-er gut
- Kombination führt zu CMOS-Logik

CMOS-Logik:

- p-Stack an VDD, n-Stack an GND
- die beiden Stacks sind komplementär zueinander, dh. Serienschaltung werden zu Parallelschaltungen und anders herum

Open Drain:

- p-Stack wird ausgelassen, nur 0-er kann gut getrieben werden
- kann verwendet werden um "Wired And" herzustellen

AOI/OAI

- And-Or-Invert / Or-And-Invert
- entspricht DNF- / KNF-Darstellung

Transmission Gate

- realisierung durch zwei MOSFETs
- intuitiv: Schaltbare Verbindung
- ermöglicht Multiplexer

Multiplexer:

- entweder mit Transmission Gates oder in AOI

Getakteter Inverter:

- Transmissiongate mit Inverter
- wird öfters geraucht, daher eigenes Schaltsymbol
- einfache realisierung durch vier Transistoren
- wird verwendet für vereinfachten Aufbau von Latches und in weiterer Folge auch Flip-Flops

Design Flow eines ASICs

Y-Diagramm:

- Achsen:
 - Verhalten
 - Struktur
 - Geometrie
- Schichten:
 - system
 - algorithmic (auf dieser Ebene wollen wir beschreiben, können es aber noch nicht wirklich)
 - rtl (Beschreibung durch VHDL)
 - gate (auf dieser Ebene können wir beschreiben, wollen wir aber nicht mehr)
 - circuit

Design Flow:

- Spezifikation
- Design Entry
- Kompilieren
- Technology Mapping ← Library
- Partition, Place and Route
- Manufacturing / Download

Design-Hilfen:

- Design-Hierarchien: vgl. Funktionen in Software
- Vectored Instances: man muss nicht 16 FlipFlops angeben sondern kann sagen, man möchte einen Vektor aus 16 FFs

Partition, Place and Rout:

- Partitioning: Aufteilen auf mehrere Asics, falls nötig. Möglichst wenige Verbindungen
- Place: Elemente so anlegen, dass möglichst geringer Interconnect-Delay entsteht
 - Zwei Delays spielen Rolle: Gate-Delay (Propagation-Delay) und Interconnect-Delay
 - Interconnect-Delay wesentlich schlechter vorhersagbar als Gate-Array, wegen Abhängigkeit von Placement
 - Mittlererweise ist Interconnect-Delay der Überwiegende
- Route: Interconnect auslegen: vergleichsweise leicht (bis auf Clock)

Skew:

- Phasenverschiebung zwischen Signalen, die eigentlich gleiche Phase haben sollten

Back-Annotation:

- Nach Partition, Place and Route können Delays relativ exakt berechnet werden
- Diese Delays werden in Simulation eingespeist (= Back-Annotation)
- Danach Post-Layout-Simulation

Simulationsarten:

- Mixed-Level-Simulation
 - man kann manche Teile sehr detailliert simulieren und andere nur grob
 - große Zeitersparnis
- Sign-Off Simulation
 - dient zur Vertragsschließung zwischen Designer und Fabrik

Logikpegel in Simulation:

- 9-Wertige Logik
 - Low, High und Unknown jeweils in Weak und Strong
 - High Impedance, Dont Care, Uninitialized

Eventgesteuerte Simulation:

- Tabelle mit Events und deren Auswirkungen wird sukzessive befüllt
- Delta-Time für "zeitgleiche" Ereignisse

Statische Timinganalyse:

- Welchen Takt darf man Anlegen, damit alle Berechnungen fertig sind innerhalb eines Taktes
- kritischer Pfad: Pfad mit größtem Delay
- $f_{clk}^{-1} = \text{kritischer Pfad} + T_{su} - \text{Clock delay}$

Library-Angaben:

- Kapazitäten des Bauteils
- intrinsisches Delay und Extrinsisches Delay
- Delay = intrinsisch + Kapazität * extrinsisch
- Derating-Factor: Spannungs- und temperaturabhängiger Skalierungsfaktor von Delay
 - wenig Spannung und hohe Temperaturen machen Chip langsam
 - hohe Spannung und niedrige Temperaturen machen Chip schnell

Speichertechnologien

Unterscheidung nach...

- Medium: optisch, magnetisch, topologisch, elektrisch, etc
- ROM / read-write
- Zugriffsmöglichkeiten: random access / sequentiell (fifo/filo)
- Volatile / non-volatile

ROM:

- $2^n \times b$
- wird verwendet um kombinatorische Funktionen abzubilden
- masked-Rom: bei Fertigung programmierbar, danach nicht mehr
 - Decoder mit invertierten Ausgängen
 - "wired and" mit Dioden über Ausgänge
 - Am Ende wieder Inversion
 - bei größerem Datenraum: Am Ende Mux mit invertierten Dateneingängen, zweite Hälfte der Adressbits als Steuereingänge
- OTP-Rom: einmal programmierbar
 - wie mask-Rom
 - bei Fertigung werden an allen Kreuzungspunkten Dioden gesetzt, die zum Programmieren weggebrannt werden können
- (UV)-EPRom: löschbar und wiederprogrammierbar
 - Transistoren mit Floating-Gate
 - * Floating-Gate-Transistoren: Wie MOS-FET aber mit zwei Gates → eines davon floatet. Durch elektrisches Feld kann Ladung in Floating Gate gespeichert werden.
 - Ladung kann durch Einwirkung von UV-Strahlung aus Floating Gate "herausgeschlagen werden"
 - ⇒ UV-EPRoms müssen Fenster haben
- EEPROM: elektrisch löschbar
 - durch elektrisches Feld nicht nur beschreibbar, sondern auch wieder löschbar
 - begrenzte Zahl an Schreibzyklen ⇒ nicht geeignet für ständiges Umprogrammieren
- Steuersignale:
 - Chip Select: steuert Stromversorgung des ROMs
 - Output enable: schaltet Ausgänge ein/aus (Tri-State)

S-RAM (static RAM):

- besteht aus einfachen Speicherzellen
- 6D-Zelle:
 - rückgekoppelte Inverter (=Speicher) und jeweils ein Transistor am Ein- und Ausgang um zu schreiben/lesen
 - ⇒ nur 6 Transistoren; vgl. D-FF: mehr als 20 Transistoren
 - mehr Ähnlichkeit zu Latch als zu Flip-Flop, aber durch richtige Einbettung stört das nicht
- Steuersignale: Daten Ein-/Ausgang, Chip Select, Write-Enable, Output-Enable
- Bidirekionaler Datenbus: Ein Bus sowohl für Input, als auch Output
- SSRAM: synchronous static RAM
 - man designed RAM synchron mit pipelining
 - burst mode: Adressen werden inkrementiert statt wahlfrei eingelesen → schneller

D-RAM (dynamic RAM):

- Aufbau: ein Transistor und ein Kondensator
- Kondensator trägt Ladung und kann über Transistor angesprochen werden
- Kondensatoren sehr klein \Rightarrow entladen sich schnell \Rightarrow müssen dauernd refreshed werden
- Stuersignale:
 - Adressbits, nur halb so viele, weil beim ersten Takt nur die Zeilenadresse angelegt wird und im zweiten nur die Spaltenadresse
 - RAS/CAS: Row Adress Strobe / Column Adress Strobe
 - Write Enable
- Page Mode: Man bleibt in selber Zeile und ändert nur Spalte \rightarrow doppelt so schnell
- synchronous D-RAM
- DDRAM: Double Clock Rate DRAM
 - Es wird nicht nur steigende sondern auch fallende Flanke genutzt: doppelt so schnell

M-RAM (magnetic RAM):

- Speicherelemente sind zwei Magnetschichten (eine weich-, die andere hartmagnetisch) und einer dünnen Isolationsschicht
- Beschaltung ähnlich wie bei D-RAM
- durch polarisation der Magneten wird Tunnerstrom durch Isolationsschicht verändert
- Eigenschaften:
 - non-volatile (!)
 - sehr viele Schreibzyklen
 - 40% kleiner als D-RAM
 - stromsparend

Multiport Memory:

- ein einziges Speicherarray, aber es kann von mehreren Seiten gleichzeitig zugegriffen werden
- Arbitration Logic um Kollisionen zu verhindern
 - häufig mit Semaphoren (vgl. Betriebssysteme)

FIFO:

- First in, first out
- auf einer Seite nur hineinschreiben, auf der anderen Seite nur lesen
- Empty-Flag, Full-Flag, (Half-Full-Flag)

Error Detection & Correction:

- Parity:
 - ein Bit wird hinzugefügt, sodass Anzahl der 1-er entweder gerade (even parity) oder ungerade (odd parity) ist
- Hamming Code
 - zu Datenbits werden Prüfbits hinzugefügt
 - für jedes Prüfbit gibt es eine Prüfgleichung

- ein Fehler kann korrigiert werden
- CRC (Cyclic Redundancy Check):
 - Bildung in Hardware durch LFSR
 - Nach einfügen des letzten Datenbits kann der Inhalt des LFSRs als Prüfwort herangezogen werden
 - je größer das LFSR ist, desto mehr Fehler können erkannt werden

Zieltechnologien

ASICs:

- Full Custom:
 - alles wird selbst designet
 - alles kann optimiert werden
 - sehr aufwändig und kostspielig
 - keine Garantie bei Fertigung
- Standard-Cell ASIC / BSIC:
 - vordefinierte Zellen aus Library
 - * Library: fertig entwickelt
 - * Hard Makros: fertig geroutete "Black Box", Soft Makros: nur Netzliste \Rightarrow Technologieunabhängig
 - "Mega Cells": RAM, ROM, IP-Cores
 - Entwicklung viel effizienter
 - Fertigung immer noch aufwändig, weil Masken gefertigt werden müssen
 - weniger Optimierungsmöglichkeiten
- (Metal-)Gate-Arrays (MGAs):
 - vorgefertigte Wafer mit Basiszellen
 - nur Interconnect benutzerdefiniert
 - channelled/channel-less Gate-Arrays: Kanäle für Interconnect freigeben oder eben nicht
 - schnelle Entwicklung und schnelle Fertigung
 - noch weniger Optimierungsmöglichkeiten, Position der Gatter bereits vorgegeben
- Programmable Logic Devices (PLDs):
 - ROMs können als programmierbare Logikbausteine verwendet werden
 - Programmable Array Logic: Kombination aus And- und Or-Gattern
 - * PAL: nur And-Gatter programmierbar
 - * PLA: And- und Or-Gatter programmierbar
 - FPGA (Field Programmable Gate Array)
 - * programmierbare Makro-Blöcke mit programmierbarem Interconnect und programmierbarem I/O

FPGA:

- Programmierbare Verbindungen:
 - Antifuse: Isolationsschicht wird durchgebrannt \Rightarrow OTP (One Time

- Programmable); Bsp: Actel
- SRAM-Konfiguration: kleine SRAM-Elemente steuern Transmission-gates an \Rightarrow vor jeder Inbetriebnahme beschreiben; Bsp: Xilinx Virtex, Altera Stratix
 - EPROM-Konfiguration: wie SRAM-Konfiguration, aber non-volatile
 - Programmierbare Logik:
 - Mux-basiert: Grundprinzip ist Shannon'sches-Erweiterungstheorem
 - LUT-basiert (Look Up Table): RAM/ROM-Bausteine als Wahrheitstabellen
 - PAL/PLD: Darstellung durch DNF (disjunktive Normalform)
 - * Raster aus Leitungen. Jede Variable steht einmal normal und einmal invertiert zur Verfügung. And-Gatter müssen nur noch richtige Kreuzungspunkte abgreifen. Erleichterung durch Wired-And
 - * Logic Expander: manchmal stehen zu wenige Oder-Terme zur Verfügung. Aber manche der Und-Terme können invertiert wieder in dieses Raster rückgeführt werden und dann weiterverarbeitet werden.
 - * Manchmal wäre invertierte Funktion leichter zu implementieren \Rightarrow am Ausgang programmierbarer Inverter
 - An Ausgängen häufig Flip-Flops oder Carry-Logik, um deren Verwendung zu erleichtern
 - Programmable Interconnect (PIA):
 - entweder von jeder Source zu jeder Sink: gut vorhersehbares timing, aber unter umständen langsamere Verbindungen zwischen nahegelegenen Blöcken
 - oder unterschiedliche Kanäle für unterschiedlich nahegelegene Blöcke: zwischen nahegelegenen Blöcken sehr schnelle Verbindungen, dafür zu weiter entfernten Blöcken weniger gut vorhersehbares Timing

Datenblattangaben

Absolut Maximum Ratings: sind keine Angaben zum idealen Betriebszustand, sondern geben an, wann man den Chip mit Sicherheit zerstört hat

Temperatoreinfluss:

- hohe Temperatur führt dazu, dass der Chip langsamer wird und die Fehlerrate exponentiell ansteigt.
- entscheidend ist Junction-Temperatur – also die Temperatur des Dies, messbar ist aber nur die Gehäuse- oder Gehäusetemperatur
- vereinfachte Wärmeleitgleichung: $T_{junction} = T_{ambient} + P * \theta_{JA}$
 - $T_{junction}$... Temperatur am Die
 - $T_{ambient}$... Umgebungstemperatur
 - P ... Verlustleistung des Chips
 - $\theta_{JA} = \theta_{JC} + \theta_{CA}$... Wärmewiderstand von Junction zu Ambient: Einheit K/W

- * θ_{JC} ... Wärmewiderstand von Junction zu Case
- * θ_{CA} ... Wärmewiderstand von Case zu Ambient
- Oft haben Chips Pins, deren einzige Funktion Wärmeabfuhr ist
- Durch temperaturabhängiges Verhalten von Halbleitern kann doch wieder direkt $T_{junction}$ gemessen werden

Verlustleistung:

- Ruheströme:
 - Tunnelströme durch Gate von Transistor: im nA-Bereich, aber bei 10^9 Transitoren führt das insgesamt zu Strömen im A-Bereich
 - Leckströme von Source nach Drain trotz gespernten Transistors: Trade-off zwischen Geschwindigkeit und Energieeffizienz
 - Ströme über Pull-Up-Widerstände, z.B. bei Open-Drain und in Wired-And in PLA-Strukturen
- Ladeströme bei CMOS:
 - Parasitäre Widerstände und Kapazitäten \Rightarrow bei jedem Schaltvorgang entstehen Ladeströme
 - Verlustleistung durch Ladeströme: $P_{lade} = C_{\text{äqu}} \cdot f \cdot V_{dd}^2$
 - * P_{lade} ... Verlustleistung durch Ladeströme
 - * $C_{\text{äqu}}$... Ersatzkapazität
 - * f ... Taktfrequenz
 - * V_{dd} ... Versorgungsspannung
 - * Herleitung über $E_C = \frac{C \cdot U^2}{2}$
 - Verringerung der Verlustleistung durch Senken der Frequenz oder der Versorgungsspannung \Rightarrow dynamic voltage- and frequency scaling
- Transiente Kurzschlüsse:
 - bei Schaltvorgängen in CMOS-Strukturen könnten P- und N-Stack für kurze Zeit gleichzeitig geschlossen sein \Rightarrow Kurzschluss
 - Verringerung durch geringere Asymmetrie der Schaltzeitpunkte und selteneres Schalten
- Entwicklungstrends: sowohl statische, als auch dynamische Verlustleistung steigen, aber statische Verlustleistung steigt wesentlich stärker.

Versorgungsspannung:

- unterschiedliche Spannungen für interne Versorgung und Versorgung von I/O, um intern möglichst wenig Wärme zu erzeugen aber nach außen hin dennoch mehr Spannung erzeugen zu können
- bei zu hoher Spannung: Chip wird heiß, Fehlerrate steigt, Chip kann zerstört werden
- bei zu niedriger Spannung: Chip wird langsam Pegel und Noise-Margins stimmen nicht mehr, bei Schnittstellen können hohe Ströme auftreten
- Klemmdiode: können bei I/O kurzfristig zu hohe oder zu niedrige Spannung abfangen
- Störspannungsabstände: Chip sorgt an Ausgang für stärkere 0-er und 1-er als er am Eingang verlangt

An Eingängen oft Schmitt-Trigger mit Hysteres um unsaubere Signalverläufe zu verhindern

Ausgangsströme: Wird zu viel Strom gezogen, können...

- Flanken zu flach werden
- Spannungspegel nicht mehr eingehalten werden
- Ausgangstreiber überlastet werden

Fan-Out:

- gibt an, wie viele Gatter am Ausgang eines bestimmten Gatters hängen
- Ausgangsströme nur geringes Problem, aber Kapazität erhöht sich stark
⇒ flachere Flanken

Timing-Angaben:

- Kombinatorische Logik:
 - Propagation-Delay gibt an, wie lange ein Signal braucht, um eine kombinatorische Logik zu durchlaufen
 - Output enable/disable
- Sequenzielle Logik:
 - Setup- & Hold Time: wie lange vor und nach der Taktflanke der Eingang eines Flip-Flops gehalten werden muss
 - Clock to Data Out:
- Interconnect:
 - Delay für diverse Leitungslängen, Switchelemente, Vias

Speed-Grades: Chips können unterschiedliche Geschwindigkeitsgrade haben

Timing-Optimierung:

- Temperatur und Versorgungsspannung (siehe Derating Factor)
- Treiberstärke:
 - größere Treiber kosten mehr Fläche
- Fan-Out
 - Einfluss der Lastkapazität, zur Erinnerung: $\text{Delay} = \text{intrinsisch} + \text{Kapazität} * \text{extrinsisch}$
 - Verringerung durch Verdopplung der Funktionen
- Routing

Slew Rate:

- gibt maximale Steigung des Spannungsanstiegs an (V/s)
- höhere Slew Rate kann nur durch höhere Ströme erreicht werden
- daher konfigurierbar ⇒ dort wo sie schnell sein müssen ist die Slew Rate hoch, ansonsten etwas geringer

Taktnetz:

- komplexestes Netz im gesamten Chip
- Fan-Out von einigen Tausend

- wir wollen, dass Takt überall gleichzeitig ankommt
- \Rightarrow FPGAs stellen meist viele Clock-Domains und Netze zur Verfügung \rightarrow sollen auch verwendet werden

PLLs (Phase Locked Loop):

- Spannungsgesteuerter Oszillatior gibt Takt vor
- Phasendetektor erkennt Phasendifferenz zwischen einem Referenzsignal und Ausgangssignal des Oszillators und kann sie wieder über die Eingangsspannung des Oszillators (Stellgröße) ausgleichen \Rightarrow Regelkreis
- über Totzeitglieder kann Phasendifferenz zwischen Referenzsignal und Ausgangssignal des Oszillators eingestellt werden
- Ausgangssignal und Referenzsignal können durch natürliche Zahl dividiert werden \Rightarrow Verhältnis von Referenzfrequenz zu Ausgangsfrequenz kann beliebig (aber rational) gewählt werden
- Loopfilter: Regelungstechnisches Glied um Dynamik des Regelkreises zu beeinflussen, sodass keine ungewünschten Effekte auftreten
- Spread Spectrum: verhindert, dass Ausgangssignal eine zu regelmäßige Frequenz hat, damit der Chip nicht bei bestimmten Frequenzen zu stark abstrahlt (wichtig für diverse Zertifizierungen)

Sonstige Features von FPGAs:

- automatische Upset-Detection (erkennt, wenn Bits umkippen)
- Signal Tap: "eingebauter Logikanalysator"
- Unterstützung für diverse Protokolle

Ordering Information:

- Familiensignatur
- Geräte-Typ
- Gehäuse-Typ
- Pin-Anzahl
- Temperaturbereich
- Speed Grade
- Optionales Suffix

Grenzen des synchronen Designs

Einflussfaktoren auf Delays:

- Anzahl der durchlaufenden Logikstufen (Komplexität der Operationen, Optimierung und Mapping)
- Routing: Länger der Leitungen, Geometrie, Vias und Switches
- Betriebsbedingungen: z.B.: Temperatur und Versorgungsspannung

Skew:

- Zeitverzögerungen zwischen Signalen, die eigentlich zeitgleich an einem Ziel ankommen sollten

- Konsequenzen: z.B. Glitches

Glitch:

- ist ein sehr kurzer, unerwünschter Puls
- stört nur dann, wenn er in einen Zustand übernommen wird (z.B. wenn seine Flanke eine Auswirkung hat oder der falsche Pegel in einem Speicherelement übernommen wird)
- static 1: unerwarteter negativer Glitch bei Übergang von 1 auf 1
- static 0: unerwarteter negativer Glitch bei Übergang von 0 auf 0
- dynamic Glitch: unerwarteter Glitch vor einem Zustandswechsel

Hazard: die Möglichkeit, dass Glitches auftreten können.

Datenkonsistenz: Daten, die gemeinsam interpretiert werden sollen, müssen auch zeitlich richtig korreliert sein. Ansonsten sind die Daten nicht konsistent

Bool'sche Logik kann diese zeitlichen Korrelationen nicht darstellen \Rightarrow ungeeignet um Dateninkonsistenz zu modellieren

Grundproblem:

- Es gibt Daten-Sources und Daten-Sinks. Dazwischen befindet sich eine Logik-Wolke
- Wann sind die Daten am Eingang der Sink gültig und konsistent? Wann darf die Source neue Daten durch die Wolke schicken?
- Lösung des synchronen Designs: wenn man lang genug wartet, vergehen schon die transienten Zustände \Rightarrow ausreichend langsamer Takt
- asynchrone Lösungen: bounded-delay (wie Timer) oder delay-insensitive Codes

Probleme des Taktnetzes:

- Energieverbrauch: Taktnetz braucht in etwa 50% der gesamten Leistung des Chips (wegen vieler Treiber und hohen Fan-Outs)
- bei steigender Flanke ziehen alle angeschlossenen Blöcke zeitgleich Strom \Rightarrow sehr hohe Stromspitzen
- Strahlung: die langen Leitungen des Taktnetzes wirken wie Antennen und strahlen bei Taktfrequenz besonders stark ab
- Schwindende Zeitreserven: Timing wird bis zum Geht-nicht-mehr ausgequetscht \Rightarrow einfache Grundidee des synchronen Designs wird doch wieder wahnsinnig schwer

Asynchrone Eingänge:

- innerhalb des Chips kann Timing (mehr oder weniger gut) vorhergesagt werden, aber an I/O Schnittstellen können asynchrone Ereignisse eintreten
- Wahrscheinlichkeit für Verletzung des Set-Up & Hold-Windows:

$$P_{violate} = \frac{T_0}{T_{clk}} > 0$$

$$- P_{violate} \dots$$
 Wahrscheinlichkeit für Verletzung des Set-Up & Hold-Windows

- T_0 ... Dauer des Set-Up & Hold-Windows
- T_{clk} ... Taktperiode
- Auch an Übergängen von einer Clock-Domain zu einer anderen können solche Set-Up-Hold-Violations auftreten

Metastabilität:

- Wenn Daten früh genug (also außerhalb des Set-Up & Hold-Windows) an ein Flip-Flop angelegt werden, werden sie mit der Taktflanke übernommen. Legt man sie jedoch erst später an, braucht das Flip-Flop länger, um die Daten zu übernehmen. Im Worst Case kann es sogar unendlich lange dauern und bis dahin in einem ungewünschten Zwischenzustand verharren \Rightarrow Metastabilität
- Metastabilität kann dazu führen, dass sich dieser Zwischenzustand im Chip fortpflanzt
- Metastabilität kann zu inkonsistenter Wahrnehmung führen: Ein hinter dem metastabilen Gatter liegendes Gatter kann den Zwischenzustand als 0-er interpretieren, ein anders als 1-er
- Resolution Time zwischen zwei Flip-Flops: $t_r = T_{clk} - t_{CO} - t_{comb} - t_{su}$
 - t_r ... Resolution Time
 - T_{clk} ... Taktperiode
 - t_{CO} ... Dauer von Clock zu Output
 - t_{comb} ... Dauer zum durchlaufen einer Logikwolke
 - t_{su} ... Set-Up-Zeit (ohne Hold-Zeit)
- Upset: Metastabilität hält länger an als Resolution Time \Rightarrow kann nicht aufgelöst werden
- MTBU (Mean Time Between Upset):

$$MTBU = R_{upset}^{-1} = \exp\left(\frac{t_r}{\tau_c}\right) \cdot \frac{1}{T_0 \cdot f_{clk} \cdot \lambda_{dat}}$$

– $MTBU$... Mean Time Between Upset
 – R_{upset} ... Upset-Rate
 – t_r ... Resolution Time
 – τ_c ... Bauteilparameter des Flip-Flops
 – T_0 ... Decision-Window, vgl. Wahrscheinlichkeit für Setup-Hold-Violation
 – f_{clk} ... Taktfrequenz
 – λ_{dat} ... Datenrate am Eingang (doppelte Frequenz)
 – Herleitung über $R_{upset} = \underbrace{\exp\left(-\frac{t_r}{\tau_c}\right)}_{\substack{\text{Wahrscheinlichkeit, dass man aus} \\ \text{Metastabilität nicht rechtzeitig herauskommt}}} \cdot \underbrace{\frac{T_0}{T} \cdot \lambda_{dat}}_{\substack{\text{Wahrscheinlichkeit, dass man} \\ \text{in Metastabilität hineinkommt}}}$

- Vergrößerung der MTBU durch besonders designete Flip-Flops mit geeigneteren Bauteilparametern und möglichst hoher Resolution Time
- Synchronizer: Kaskade von Flip-Flops. Durch jedes Flip-Flop steigt die Resolution Time um (fast) eine Taktperiode \Rightarrow sehr starke Auswirkung auf MTBU

- In konventionellen synchronen Designs lässt sich Metastabilität nicht vermeiden, nur hinreichend unwahrscheinlich machen. In adaptierten synchronen Designs oder asynchronen Designs (mit Handshake) lässt sie sich jedoch gänzlich vermeiden

Provozieren von Metastabilität

- asynchrone Inputs
- viele Clock-Domains
- wenig Zeitreserve (Resolution Time)
- langsame Technologie
- zu geringe Versorgungsspannung
- zu hohe Temperatur

"Metastabilität war früher ein Problem, heute muss man sich nicht mehr darum kümmern": Falsch! Technologie ist zwar besser geworden, aber Resolution Time ist auch wesentlich kleiner geworden.

Defekte und Fehler

Fehlerquellen im Aufwind:

- wachsender Zeitdruck
- zunehmender Anteil an Fremddesign
- steigende Komplexität
- kleinere Strukturen
- sinkende Versorgungsspannung
- steigende Taktraten
- Laien als Anwender

Fehler in der Fertigung:

- Wafer: Verunreinigungen, Microcracks, Kristalldefekte
- Prozesse: Masken-Alignment, Unterätzung
- Packaging: Hohlräume (schlechtere Wärmeabfuhr), Bonding-Defekte
- Transport: unsachgemäße Handhabung
- Bestückung: kalte Lötstellen, Kurzschlüsse, etc

Badewannenkurve:

- beschreibt nach welcher Zeit im Einsatz ein Chip wie wahrscheinlich ausfällt
- Am Anfang relativ hoch - "Infant Mortality", danach einige Zeit lang relativ niedrig und mit Einsetzen von Alterungserscheinungen wieder wesentlich höher

Fehlerquellen im Einsatz:

- Electrical Stress
 - Electrostatic Discharge (ESD): Bei elektrostatische Entladung kann (ziemlich sicher) Gate-Oxid durchbrochen werden. ESD kann durch

- Erdungsbänder verhindert werden oder Klemmdioden abgefangen werden
- Electrical Overstress: Spannungsspitzen in der Versorgung, Überspannung durch Blitzschlag, Out-of-Spec Benutzung. Folgen: Thermische Zerstörung oder Beschädigung des Oxids
 - Latch-Up: Durch Struktur von CMOS-Invertern bilden sich ungewünschterweise Thyristoren. Wenn dieser Thyristor einen Verstärkungsfaktor > 1 hat und aktiviert wird, verursacht er einen Kurzschluss und brennt sich selbst ab.
 - Intrinsic
 - Gate-Oxid Wear-Out: Störstellen im Oxid dienen als Stützpunkte für Ladungen \Rightarrow wesentlich höhere Tunnelströme oder gar ein Pfad für Strom (Gate-Oxid Break-Down)
 - Ionic Contamination
 - Oberflächenladung
 - Kristalldefekte
 - Piping
 - Extrinsic
 - Elektromigration: Elektronenwind bewegt Leiter
 - * Mean Time To Failure: $MTTF = \frac{A}{J^2} \cdot \exp\left(\frac{E}{k \cdot T}\right)$
 - * A ... Konstante
 - * J ... Stromdichte, Einheit A/cm²
 - * E ... Aktivierungsenergie (Materialkonstante)
 - * k ... Boltzmann-Konstante
 - * T ... Temperatur (meistens in Kelvin)
 - Kontaktmigration
 - Stress-induzierte Migration: Migration infolge mechanischer Spannung
 - Microcracks
 - Die Attach Failure: Unerwünschte Hohlräume
 - Bonding Failure: Reißen oder ablösen des Bondings, Whiskers
 - Popcorn-Effekt: Ausdehnung von Feuchtigkeit
 - Korrosion: chemische Effekte
 - Soft Errors: z.B. Bit-Flips durch Strahlung

Arrhenius-Gleichung:

- viele Fehlverhalten lassen sich durch Arrhenius-Gleichung modellieren
- Fehlerrate $F = C \cdot \exp\left(-\frac{E_{act}}{k \cdot T}\right) \approx MTTF^{-1}$
- Wichtige Beobachtung: Temperatur wirkt sich exponentiell aus

Burn In:

- Chip wird für kurze Zeit unter hohem Stress betrieben, um möglichst schnell die Phase der Infant Mortality zu durchlaufen
- Temperature Acceleration Factor: $AF_T = \exp\left(\frac{E_{act}}{k \cdot T_{normal}} - \frac{E_{act}}{k \cdot T_{stress}}\right)$
- Voltage Acceleration Factor: $AF_V = \exp(\gamma \cdot (U_{stress} - U_{normal}))$
- Acceleration Factor: $AF_{ges} = AF_T \cdot AF_V$

Fehlermanifestation:

- Kontaktprobleme, Isolationsprobleme, parametrische Fehler, dynamische Fehler (verringerte Geschwindigkeit), Speicherfehler (Bit-Flip)

Testen

"Stuck-At"-Fehlermodell:

- Modell: eine Leitung ist immer nur 1 (stuck at 1) oder immer nur 0 (stuck at 0)
- Single Stuck-At - Modell: es tritt immer nur ein einziger Stuck-At-Fehler auf
- Fehlermodell ist zwar nicht sonderbar exakt aber Test ist auch so schon komplex genug. Außerdem lassen sich anscheinend durch dieses Fehlermodell auch viele andere Fehler finden.

Rule of Ten: Die Kosten eines Fehlers oder Defekts steigen bei jedem Assemblierungsschritt (fertiger Chip, Aufbringen auf Platine, Einbringen in Computer, Einbinden in Applikation) um den Faktor 10

Testqualität:

- Defect Level: Anteil an fehlerhaften Chips, die *nicht* durch den Test ausscheiden
- Test Coverage: gibt an, wieviel der (mit dem Fehlermodell darstellbaren) Fehler erkannt werden können
- wichtige Erkenntnis: 100% Test Coverage bedeuten nicht, dass man garantiert alle Fehler findet, sondern bestenfalls die, die vom Fehlermodell modelliert werden.

Aufbau eines Testers:

- Test Pattern Generator
- Response Analysis
- Test Controller

Tests:

- exhaustive testing: alle möglichen Bit-Kombinationen
- deterministic testing:
 - Annahme eines Fehlers, Aktivieren und Rechtfertigen des Fehlers, Ausgang beobachten
 - Für alle Fehler im Fehlermodell so durchführen
 - Liste an zu überprüfenden Fehlern kann reduziert werden, indem man äquivalente Fehler streicht oder erkennt, dass man mit der Überprüfung eines Fehlers auch gewisse andere Fehler ausschließen kann
 - "hard to detect"-Fehler: z.B. Fehler in redundanter Logik

- Testbarkeit: Steuerbarkeit und Beobachtbarkeit. Weitere Pins, die Logik in der Mitte abgreifen, erhöhen Testbarkeit
- non-deterministic testing:
 - pseudozufällig Vektoren
 - kann für niedrigere Testabdeckungen sehr schnell zum Ziel führen
 - gut in Hardware realisierbar (LFSR)
 - Zufallsfolge kann so gewählt werden, dass sie bestimmte Vektoren auf jeden Fall enthält
- Scan-Test: bei sequenzieller Logik:
 - nicht funktional sondern Strukturell \Rightarrow "Wenn alle Einzelteile funktionieren, funktioniert auch das Ganze"
 - Speicherlemente werden zu Schieberegistern umfunktioniert \Rightarrow Testvektoren können einfach in Register hineingeschoben werden und liegen dann an kombinatorische Logik an
 - Test-Modus (also das Auffädeln zu Scheiberegister) wird über Multiplexer aktiviert \Rightarrow ca. 10% mehr Fläche, längerer kritischer Pfad, Verdrahtungsaufwand
 - Vorteile: Sequenzielles Problem auf kombinatorisches reduziert, direktes Setzen interner Zustände, hohe Testbarkeit, einfach automatisierbar, Zugriff auf Registerketten über wenige Pins
 - Problem: Dauer um Daten in Register zu schieben steigt mit der Länge der Scan-Chains
- March-Test: für Speicherelemente:
 - Fehlermodell Stuck-At nicht mehr ausreichend, wird ergänzt um Coupling Faults und Neighborhood Pattern Sensitive Faults
 - Schreibe in Register nur 0-er, überprüfe 0-er und schreibe 1-er, überprüfe 1-er und schreibe 0-er, überprüfe 0-er, wiederhole in umgekehrter Reihenfolge

Testen von PLDs/FPGAs:

- Grundfunktionen werden in Fabrik nach Fertigung getestet
- korrekte Programmierung kann nach Download überprüft werden (Rücklesen)
- korrektes Design muss vom Designer sichergestellt werden (Simulation)
- programmierte Bausteine lassen sich genau so testen wie nicht programmierbare

Boundary Scan:

- dient zum Testen, ob ein Chip auf einer Platine richtig eingebettet ist und die Verbindungen zu anderen Komponenten funktioniert
- Chip A schiebt Bitmuster über I/O hinaus, Chip B empfängt es. Wenn das empfangene Muster dem gesendeten spricht, geht man davon aus, dass die Verbindung funktioniert

JTAG Test Access Port:

- kleine Finite State Machine (FSM), über die Tests am bereits eingebauten

Chip abgewickelt werden können: Boundary Scan, March-Tests, Scan-Tests, etc.

Design for Test (DFT):

- Herstellungskosten für Chips fallen aber Testkosten bleiben konstant \Rightarrow irgendwann kostet Test mehr als Herstellung des Chips. Daher möchte man Tests billiger machen und designt den Chip bereits so, dass er gut getestet werden kann
- einige Design-Regeln:
 - komplexe Logik partitionieren, Testhilfen in langen Zählerketten einfügen
 - Initialisierung/Reset für sequenzielle Logik vorsehen
 - redundante Logik vermeiden
 - keine Verzögerungsglieder als funktionale Elemente
 - strikte Trennung von Takt und anderen Signalen
 - Clock Gating vermeiden
 - Bus-Strukturen bevorzugt verwenden

Built-in Self Test (BIST):

- simplere Test-Bench wird bereits auf Chip gebaut
- bevorzugt Non-Deterministic Tests mit LFSR, weil man nicht genügend Testvektoren eines deterministischen Tests speichern könnte
- Response Analyzer arbeitet mit Multiple Input Shift Register (MISR), das am Ende eine Signatur erzeugt \Rightarrow es muss nur noch Signatur überprüft werden
- Vorteile: vereinfachtes Interface, Test kann "at speed" ablaufen, erhöhte Testbarkeit weil besser Abgreifbar, geringer Overhead
- für IP-Cores müssen keine Testfälle mehr mitgeliefert werden, sondern nur BIST \Rightarrow kein Reverse-Engineering
- rasche Fehlererkennung (und Diagnose) im Feld