NEUE SKALIERTE TM–DUAL–RAIL–COMPUTERARCHITEKTUR

Der Speicher (Flip-Flop) wird zum Prozessor – das Ende des Von-Neumann-Flaschenhalses

Software/Algorithmen werden algebraisch

Deterministische KI, minimale Energieaufnahme und sicherheitskritische Echtzeit

Autor: Peter Hettich
Patentanmeldung: WO2025061229
Version: 2.0
Datum: 18 -12-2025

Zusammenfassung

Der exponentielle Anstieg des Rechenbedarfs, getrieben durch künstliche Intelligenz, autonome Systeme, industrielle Automatisierung, Telekommunikation sowie Hochleistungsrechner, hat die fundamentalen, physikalischen und architektonischen Grenzen klassischer Computersysteme offengelegt. Die Trennung von Speicher- und Recheneinheiten, wie sie in der Von-Neumann-Architektur formalisiert ist, führt zu exzessiver Datenbewegung, hohen Latenzen, thermischer Instabilität und geringer Energieeffizienz.

Dieses Whitepaper stellt **TM–Dual–Rail Computing** (Thue Morse-Dual-Rail Computing) vor – eine neuartige, skalierte Boolesche Architektur, die auf einem neuen fundamentalen Logikelement basiert: dem **bidirektionalen algebraischen Dual-Rail-Flip-Flop (BFF)**. Die Architektur vereint Speicher, Steuerung und Operation/Berechnung in einer einzigen physikalischen Struktur und ermöglicht damit **geringe Datenbewegung**, **deterministisches Zeitverhalten**, **ultraniedrigen Energieverbrauch** und **hohe Rechendichte**. Die Software/Algorithmen werden algebraisch. Ein typischer Anwendungsfall der TM–Dual–Rail-Architektur ist der sicherheitskritische Regelkreis, wie er zum Beispiel in autonomen Systemen, der Industrieautomatisierung, der Energieinfrastruktur oder der Telekommunikation vorkommt. Vorgestellt werden die theoretischen Grundlagen, die Implementierung auf Transistorebene, die Systemarchitektur, die Leistungsmerkmale, validierte Simulationsergebnisse, sowie die Anwendungsfelder in den Bereichen KI, Automotive, Industrieautomatisierung, Telekommunikation, Medizintechnik, Energie und kritische Infrastrukturen.

1. Einleitung

Das Ende der klassischen Skalierung

Über mehr als 70 Jahre hinweg wurden Leistungssteigerungen in der Rechentechnik durch Prozessskalierung (Mooresches Gesetz) und Frequenzsteigerungen (Dennard-Skalierung) erzielt. Beide Mechanismen haben sich mittlerweile grundlegend verlangsamt:

- Transistorskalierung führt nicht mehr zu proportionalen Leistungsgewinnen
- Frequenzskalierung ist thermisch begrenzt
- Speicherbandbreite wächst deutlich langsamer als die Rechenleistung

- Der Energieverbrauch der Interconnects dominiert die Chip-Energiebudgets

Gleichzeitig wachsen KI-Workloads überproportional in Bezug auf Datenbewegung, Speicherzugriffe und Parameteranzahl. Das Ergebnis ist eine **strukturelle Krise der Recheneffizienz**.

2. Der Von-Neumann-Flaschenhals

Die Architektur als limitierender Faktor

In allen klassischen Architekturen gilt:

- Der Speicher ist physikalisch vom Prozessor getrennt
- Jede Operation erfordert:
    1. Laden aus dem Speicher
    2. Transport über Interconnects
    3. Berechnung in der ALU
    4. Rückschreiben in den Speicher

Dies führt zu:

- **70–90 % des Gesamtenergieverbrauchs durch Datenbewegung**
- **10–30 Taktzyklen Latenz pro Speicheroperation**
- Dominanz der Interconnects bei Chipfläche und Wärmeentwicklung
- Skalierungsgrenzen unabhängig von der Transistoranzahl

Moderne GPUs, TPUs und NPUs bleiben somit **fundamental speichergebundene Systeme (memory-bound)**.

3. Warum KI, Industrie & Infrastruktur dieses Modell sprengen

Die folgenden Domänen sind mit klassischen Architekturen strukturell nicht kompatibel:

- Autonome Systeme (Echtzeitbedingungen)
- Medizinische Bildgebung (deterministische Sicherheits-Zeitpfade)
- Telekommunikation & 5G/6G-RAN (ultraniedrige Latenz am Edge)
- Industrieautomatisierung (24/7-deterministische Regelkreise)
- Energieinfrastruktur (thermische & EMV-Beschränkungen)

Diese Anwendungsfelder erfordern:

- Deterministisches Zeitverhalten
- Geringe Wärmeentwicklung

- Höchste Zuverlässigkeit
- Extreme Energieeffizienz
- Lokale (On-Device) Rechenverarbeitung

3.1 Typischer Use Case – Deterministische KI im sicherheitskritischen Regelkreis

Ein typischer Anwendungsfall der TM–Dual–Rail-Architektur ist der **sicherheitskritische Regelkreis**, wie er zum Beispiel in autonomen Systemen, der Industrieautomatisierung, der Energieinfrastruktur oder der Telekommunikation vorkommt. Solche Systeme lassen sich formal als **endliche Automatenmodelle** beschreiben, bestehend aus Zuständen, Zustandsübergängen und Rückkopplung. Klassische Architekturen realisieren diese Automaten durch eine Trennung von sequenziellem Speicher (Flip-Flops) und kombinatorischer Logik, was zwangsläufig zu wiederholten Lade-, Verarbeitungs- und Rückschreibzyklen führt.

Die TM–Dual–Rail-Architektur ersetzt diese Trennung durch eine **homogene Regelkreisstruktur**. Die kombinatorischen Blöcke eines Automatenmodells werden durch **algebraische Dual-Rail-Flip-Flops (BFF)** realisiert, die Speicherung, Zustandsübergang und logische Operation in einer einzigen physikalischen Struktur vereinen. Der Regelkreis wird dadurch zu einem lokal geschlossenen, deterministischen Zustandsraum.

Das BFF kann bidirektional betrieben werden: Im **IN/OUT-Modus** fungiert es als sequentieller Speicherautomat, im **OUT/IN-Modus** als algebraische Funktion. Die Umschaltung zwischen beiden Betriebsarten erfolgt rein elektrisch durch Vertauschung der Ein- und Ausgangsvariablen und ist in einem Schritt möglich. Dadurch kann derselbe Regelkreis unmittelbar sowohl als **Reaktor** auf Eingangssignale reagieren als auch als **Aktor** Zustandsänderungen erzeugen.

Auf diese Weise werden **KI-basierte Entscheidungsfunktionen direkt in den Regelkreis integriert**. Die Entscheidung ist kein separater Rechenschritt mehr, sondern ein deterministischer algebraischer Zustandsübergang mit garantierter Latenz von wenigen Takten und minimalem Energieverbrauch durch rein lokale Umschaltung. Die vollständige algebraische Beschreibung der Zustandsräume ermöglicht zudem eine formale Verifikation und Zertifizierbarkeit für sicherheitskritische Anwendungen.

Zur Systemintegration kann der Dual-Rail-Regelkreis als gekapselte **Safety Island** in bestehende Single-Rail-Architekturen eingebettet werden. Damit wird eine signifikante Erhöhung von Determinismus, Energieeffizienz und funktionaler Sicherheit erreicht, ohne die bestehende Systemarchitektur grundlegend zu verändern.

4. TM–Dual–Rail Computer: Architekturüberblick

TM–Dual–Rail Computing beseitigt die Trennung von Speicher und Logik vollständig. Es stellt die erste skalierte Realisierungsebene des TM Computing dar und basiert auf dem Booleschen CNOT/XOR-Grundelement. Dieses erzeugt ein neues physikalisches Dual-Rail-Shannon-Bit (01,10) als Konkatenation der beiden Single-Rail-Bits 0 und 1.

Das neue Dual-Rail-System bildet damit das formal einfachste, zugleich jedoch komplexe Zweiteilchensystem, das zu einem koordinierten, algebraisch steuerbaren Zustandswechsel fähig ist.

Die Skalierung und Vereinheitlichung kombinatorisch-algebraischer Schalteranordnungen mit sequenziellen Schalteranordnungen (Flip-Flops) führt zu einer neuen technologischen Grundlage, in der Speicherung, Logik und Zustandsautomat erstmals physikalisch vereinheitlicht werden – als fundamentale Vereinheitlichung der Informatik.

Grundprinzip:

Der Speicher, das Flip-Flop ist der neue Prozessor (speichern, operieren, steuern).

Im Zentrum der Architektur steht das:

BFF – Bidirektionale algebraische Dual-Rail-Flip-Flop

Ein einziges physikalisches Bauteil übernimmt zwei Betriebsarten:

- **Sequentielle Speicheroperationen**
- **Kombinatorische algebraische Logikoperationen**

Die Umschaltung zwischen den Betriebsmodi erfolgt rein elektrisch durch den einfachen Wechsel der Ein- und Ausgangsvariablen. Der Geschwindigkeitsvorteil des BFF ergibt sich primär aus der Eliminierung des klassischen Datentransfers zwischen Speicherwerk und Rechenwerk. Während in der klassischen Technologie ein Zustands-Update (Setzen/Löschen/Halten) mindestens die Abfolge „Lesen → kombinatorische Entscheidung → Zurückschreiben“ durchlaufen muss, erfolgen bei der BFF-Speicherung, Steuerung und Operation im selben physikalischen Element. Dadurch entfallen Transfer- und Verteilungszeiten (Interconnect, Fanout, MUX-Gating), die in konventionellen Datenpfaden häufig einen großen Anteil der Zykluszeit dominieren.

Die Zeilen in den Wertetabellen für Setzen, Löschen und Halten (Speicherung) bleiben dabei in beiden Modi unverändert. Im kombinatorisch-algebraischen Betrieb stehen mehrere Varianten, durch Tausch der Ein/Aussgangsvariablen zur Verfügung (z. B. BFF1, BFF2, BFF3). Das BFF ist physikalisch aus 16 Transmission Gates (32 Transistoren) aufgebaut. Funktional verfügt es über vier Eingangsvariablen sowie zwei Ausgangsvariablen. Abhängig von der jeweiligen Kombination aus Eingangs-, Ausgangs- und Steuervariablen ergibt sich die hohe Funktionsvielfalt.

Die Funktionsmöglichkeiten aller algebraischer BFF lassen sich wie folgt darstellen:

- 4 Eingangsvariablen, 2 Ausgangsvariablen: Anzahl der Funktionen: 2
- 3 Eingangsvariablen, 2 Ausgangsvariablen, 1 Steuervariable: Anzahl der Funktionen: 4
- 2 Eingangsvariablen, 2 Ausgangsvariablen, 2 Steuervariablen: Anzahl der Funktionen: 8
- 1 Eingangsvariable, 2 Ausgangsvariablen, 3 Steuervariablen: Anzahl der Funktionen: 16Das elektronische Grundelement des BFF sowie des zugrundeliegenden CNOT/XOR ist das Transmission Gate, ein seit Jahrzehnten bewährtes elektronisches Schaltelement. Die Funktionsweise dieses Bauteils ist technologieunabhängig und kann prinzipiell auch in anderen physikalischen Realisierungen umgesetzt werden – etwa in physikalischen, hydraulischen, pneumatischen, biologischen oder atomaren Systemen –, sofern dort ein Schalter (IDENT) und dessen Komplement (NOT) realisierbar sind.

4.1 Architekturvergleich

Das BFF wird mit den klassischen Computerarchitekturen (CPU, GPU, TPU, PIM) verglichen. Alle klassischen Architekturen sind dadurch gekennzeichnet, **dass der Speicher und der Prozessor (Recheneinheit) getrennt sind.**

Erläuterung:

CPU (Central Processing Unit)
Universeller Prozessor für allgemeine Programme, flexibel, aber energie- und speichergebunden.

GPU (Graphics Processing Unit)
Massiv paralleler Prozessor für viele gleiche Rechenoperationen gleichzeitig (z. B. KI, Grafik), aber stark speicherabhängig.

TPU (Tensor Processing Unit)
Spezialprozessor für KI-Matrixoperationen, sehr schnell, aber durch **HBM-Speicherzugriffe limitiert** und nur eingeschränkt programmierbar.
HBM (High Bandwidth Memory)
Extrem schneller, gestapelter Hochleistungsspeicher, direkt neben GPU/TPU, mit sehr hoher Bandbreite, aber hoher Leistungsaufnahme.

PIM (Processing In Memory)
Rechnen direkt im Speicher, reduziert Datenbewegung, ist aber meist **fest verdrahtet (Fixed-Function)** und nicht frei programmierbar. Bei der klassischem PIM sind Speicher und Rechenwerk weiterhin getrennt, nur näher zusammengebracht.

BFF – (Bidirektionale algebraische Dual-Rail-Flip-Flop) Im TM–Dual–Rail-BFF sind Speicherung (IN/OUT) und Berechnung (OUT/IN) identisch in einem einzigen physikalischen Bauelement vereinigt.

Der entscheidende Unterschied zu BFF/TM–Dual–Rail

Architektur	**Speicherzelle**	**Recheneinheit**	**Physikalisch identisch?**
CPU	getrennt	getrennt	Nein
GPU	getrennt	getrennt	Nein
TPU	getrennt	getrennt	Nein
PIM	im selben Chip	daneben	**Nein**
BFF / TM–Dual–Rail	Ja	Ja	**Ja – identisch**

Nur beim BFF ist die *gleiche physikalische Struktur gleichzeitig Speicher UND Recheneinheit*.

4.2 Grundprinzip der Architekturen (physikalisch)

Architektur	Speicher	Rechenwerk	Physische Trennung?	Grundproblem
CPU	extern (Cache/DRAM)	ALU/FPU	Ja	Datenverkehr dominiert
GPU	extern (HBM)	Tausende ALUs	Ja	Speicherbandbreite limitiert
TPU	extern (HBM)	Matrix Units	Ja	**HBM-Bottleneck**
klassisches PIM	im DRAM	Mini-ALUs	Teilweise	**Fixed-Function**
BFF/TM–Dual–Rail	identisch mit Rechenwerk	identisch	**Nein**	**Keine**

Nur beim BFF sind Speicher und Rechenwerk physikalisch dasselbe Bauteil.

Abkürzungen:

- Cache (Cache Memory), DRAM (Dynamic Random Access Memory)
  Zwischenspeicher (Cache) für extrem schnellen Zugriff im Prozessor; DRAM ist der Hauptarbeitsspeicher für Programme und Daten.
- ALU (Arithmetic Logic Unit) /FPU (Floating Point Unit)
  ALU rechnet mit Ganzzahlen (Addieren, Vergleichen, Logik), FPU rechnet mit Gleitkommazahlen (z. B. für KI, Physik).
- DRAM (Dynamic Random Access Memory)
  Flüchtiger Hauptspeicher, in dem Programme und Daten während des Betriebs liegen (langsam als Cache, schneller als SSD).
- HBM (High Bandwidth Memory)
  Extrem schneller, gestapelter Hochleistungsspeicher, direkt neben GPU/TPU, mit sehr hoher Bandbreite, aber hoher Leistungsaufnahme.

4.3 Datenbewegung (zentraler Engpass moderner Systeme)

Architektur	Datenbewegung pro Operation	Busse nötig?	Cache nötig?
CPU	sehr hoch	Ja	Ja
GPU	extrem hoch	Ja	Ja
TPU	extrem hoch	Ja	Ja
klassisches PIM	reduziert	teilweise	meist nicht
BFF / TM–Dual–Rail	nahe 1	Nein	Nein

Beim BFF **existiert kein Load/Store-Zyklus mehr**.

4.4 Programmierbarkeit & Rechenmodell

Architektur	Allgemein programmierbar?	Zustandsautomaten?	Kontrolllogik?	KI + Steuerung auf einem System?
CPU	Ja	Ja	Ja	Ja
GPU	Ja	schlecht	schlecht	getrennte Systeme
TPU	stark eingeschränkt	Nein	Nein	Nein
klassisches PIM	Nein	Nein	Nein	Nein
BFF / TM–Dual–Rail	**Ja, algebraisch vollständig**	**Ja, direkt algebraisch**	**Ja, direkt im Zustand**	**Ja, vollständig**

Der **BFF enthält in einem Bauteil:**

- Operator
- Speicher
- Zustandsmaschine
- Steuerlogik
- Große Teile der Algorithmen/Software

4.5 Latenzvergleich (physikalisch bedingt)

Architektur	Speicherlatenz	Logiklatenz	Gesamtpfad
CPU	10–30 Zyklen	1–3 Zyklen	20–100 Zyklen
GPU	200–800 Zyklen (HBM)	1–4	sehr hoch
TPU	200–800 Zyklen (HBM)	1	sehr hoch
klassisches PIM	2–10 Zyklen	1–2	mittel
BFF / TM–Dual–Rail	**1–2 Zyklen**	**1 Zyklen**	**minimal**

Für **Regelkreise, KI-Echtzeit, Sicherheit** ist das ein **quantitativer Sprung**.

Die Logiklatenz bezeichnet die Zeit (bzw. Anzahl der Taktzyklen), die ein logisches Rechenelement (z. B. AND, NOT, ALU, Gatter) benötigt, um ein Eingangssignal in ein stabiles Ausgangssignal umzuwandeln.

4.5.1 Formale Herleitung des Geschwindigkeitsvorteils

In klassischen Computerarchitekturen besteht ein Zustands-Update eines Regelkreises oder Automatenmodells aus mehreren logisch getrennten Phasen: dem Lesen des aktuellen

Zustands aus dem Speicher, der kombinatorischen Auswertung der Übergangsfunktion sowie dem Rückschreiben des neuen Zustands. In modernen Prozessorarchitekturen dominiert dabei häufig der Zeitanteil der Datenbewegung zwischen Speicher, Cache-Hierarchien, Bussen und Recheneinheiten den Gesamtpfad, während die eigentliche logische Auswertung nur einen vergleichsweisen kleinen Anteil der Latenz verursacht.

Dieser Zusammenhang lässt sich formal beschreiben, indem der Gesamtzeitbedarf eines Zustands-Updates in einen Logikanteil und einen Transferanteil zerlegt wird. Sei $p$der Anteil der Datenbewegung an der Gesamtlatenz klassischer Architekturen und $r$der Reduktionsfaktor dieses Transferanteils durch die physikalische Vereinigung von Speicher und Logik. Der resultierende Geschwindigkeitsvorteil Slässt sich näherungsweise angeben als:

$$S \approx \frac{1}{(1-p) + \frac{p}{r}}$$

Da in speichergebundenen Architekturen typische Werte für p im Bereich von 0,5 bis über 0,8 liegen, ergibt sich bereits bei moderaten Reduktionsfaktoren ein Speedup von mehreren Faktoren. Im TM–Dual–Rail-BFF entfällt der klassischen Load/Store-Zyklus vollständig, da Speicherung, Zustandsübergang und logische Operation im selben physikalischen Element stattfinden. Der Zustandswechsel erfolgt dadurch lokal in ein bis zwei Takten, was den im Latenzvergleich beobachteten quantitativen Sprung systemisch erklärt.

4.6 Energie pro Operation

Architektur	**Energie/Operation**	**Hauptverlustquelle**
CPU	10–100 pJ	Busse + Cache
GPU	50–300 pJ	HBM + Interconnect
TPU	30–150 pJ	HBM
klassisches PIM	5–20 pJ	interner DRAM-Transport
BFF / TM–Dual–Rail	**1–2 pJ**	**nur lokales Umschalten**

Das ist ein wesentlicher **physikalische Vorteil der BFF-Architektur**.

Physikalische Bedeutung -Kalkulation „Energie pro Operation“

Das zukünftige **BFF besteht lediglich aus 16 Hochleistungs-Transmission-Gates bzw. 32 Transistoren**.
Damit findet **Speicherung und Berechnung lokal in exakt demselben physikalischen Bauelement statt**. Es gibt:

- **keine Busse**
- **keine Cache-Hierarchien**
- **keine externen Speicherzugriffe**
- **keine langen Leitungen**

- **keinen globalen Datenverkehr**

Die Schaltenergie entsteht **ausschließlich durch das lokale Umsteuern dieser 32 Transistoren**.

Dies erklärt die erwartete extrem niedrige Energie pro Operation von **nur 1–2 pJ**.

Warum andere Architekturen so viel mehr Energie benötigen

CPU: 10–100 pJ

- Daten müssen ständig zwischen:

  Cache

  DRAM

  ALU/FPU
  über lange Leitungen transportiert werden

- **Hauptverlustquelle: Busse + Cache**
- Physikalisch: große parasitäre Kapazitäten → hoher Lade-/Entladestrom

GPU: 50–300 pJ

- Massive Parallelität
- Ständiger Zugriff auf **HBM**
- Sehr breite Interconnects
- **Hauptverlustquelle: HBM + Interconnect**
- Physikalisch: hohe Bandbreite = extrem hohe Schaltenergie

TPU: 30–150 pJ

- Matrix-Recheneinheiten sind effizient

  Aber: Gewichte und Aktivierungen kommen ständig aus **HBM**

- **Hauptverlustquelle: HBM**
- Physikalisch: Speicherzugriff dominiert die Energiebilanz

Klassisches PIM: 5–20 pJ

- Rechnen näher am Speicher
- Aber: Recheneinheit **immer noch getrennt von der Speicherzelle**

  interner DRAM-Transport bleibt notwendig

- **Hauptverlustquelle: interner DRAM-Transport**

Warum das BFF bei nur 1–2 pJ liegt

Beim **BFF / TM–Dual–Rail** gilt:

- **Speicher = Rechenwerk**
- **Operation = lokales Umschalten von 32 Transistoren**
- **Kein Daten-Transport über Distanzen**
- **Keine Busse**
- **Kein Cache**
- **Kein externer Speicher**
- **Keine Arbitrierung**
- **Kein Speicherzugriff im klassischen Sinn**

Bus-Arbitrierung ist der Entscheidungsmechanismus, der festlegt, welches von mehreren gleichzeitig anfragenden Geräten einen gemeinsamen Datenbus zuerst benutzen darf.

In klassischen CPU-Architekturen erfordert selbst ein einfacher Flip-Flop-Setzvorgang zahlreiche Taktzyklen (20–100 Zyklen) für Adressierung, Datenladen, Instruktion, Verarbeitung und Rückschreiben. Die dabei dominierende Energie entsteht durch Busse, Cache- und DRAM-Zugriffe. Im BFF erfolgt derselbe Vorgang als direkte lokale Zustandsschaltung in einem einzigen Zyklus mit minimaler Schaltenergie.

Die gesamte Energie wird **nur für die lokale Zustandstransformation im BFF verbraucht**.
Das ist der **fundamentale physikalische Ursprung des 10- bis 100-fachen Effizienzvorteils**.

4.6.1 Formale Herleitung der Energie pro Operation

Bei einer Versorgungsspannung im Bereich von $V = 0{,}6$–$0{,}8$ V entspricht ein Energiebedarf von 1–2 pJ pro Operation einer effektiven Schaltkapazität von $C_{\text{eff}} \approx 1{,}6$–$5{,}6$ pF. Umgelegt auf ein BFF mit 16 Transmission Gates (32 Transistoren) ergibt sich daraus eine äquivalente effektive Kapazität von etwa 100–350 fF pro Transmission Gate bzw. 50–175 fF pro Transistor. Diese Größenordnung ist konsistent mit moderner CMOS-Low-Capacitance-Logik und reflektiert ausschließlich lokale Gate-, Diffusions- und Verdrahtungskapazitäten ohne Beteiligung globaler Interconnects oder Speicherzugriffe.

4.7 Determinismus & Sicherheit

Architektur	Zeitverhalten vorhersagbar?	Cache-Effekte?	Bus-Arbitrierung?	Safety-Zertifizierbarkeit
CPU	Nein	Ja	Ja	teuer

Architektur	Zeitverhalten vorhersagbar?	Cache-Effekte?	Bus-Arbitrierung?	Safety-Zertifizierbarkeit
GPU	Nein	Ja	Ja	sehr schwierig
TPU	Nein	Ja	Ja	kaum möglich
klassisches PIM	eingeschränkt	meist nicht	teils	schwierig
BFF / TM–Dual–Rail	**Ja, vollständig deterministisch**	**Nein**	**Nein**	**Ja, optimal für ASIL/SIL**

Das BFF ermöglich eine SIL (Safety Integraty Level) Klassifizierung.

4.8 Softwareabhängigkeit

Architektur	Klassische Software nötig?	Compiler komplex?	Laufzeit-Fehler möglich?
CPU	Ja	Ja	Ja
GPU	Ja	extrem	Ja
TPU	spezial	Ja	Ja
klassisches PIM	kaum	kaum	trotzdem
BFF / TM–Dual–Rail	**stark reduziert**	algebraisch	**massiv reduziert**

Beim **BFF „wandert“ die Logik von der Software in die Hardware-Zustandsräume**.

Die Maschinensprache und damit die Algorithmen und die Software lassen sich in algebraische Strukturen transformieren.

4.9 Strategische Einordnung

- **CPU:** flexibel, aber ineffizient
- **GPU:** massiv parallel, aber speichergebunden
- **TPU:** KI-spezialisiert, aber HBM-limitiert
- **PIM:** speichernah, aber unprogrammierbar
- **BFF / TM–Dual–Rail:** ***Die erste Architektur, die Speicher, Rechnen, Zustände und Steuerung physikalisch in einem einheitlichen, Element vereint (Vereinheitlichung der Informatik).***

5. Dual-Rail-Kodierung & algebraische Logik

Die theoretische Grundlage bildet die Boolesche TM-Skalierung der BITs, Variablen und der Funktionen.

Dual-Rail-Symbole:

- Logische 0 → (01)
- Logische 1 → (10)

Eigenschaften:

- Kein ungültiger Logikzustand
- Symmetrische Darstellung
- Reversible Logik möglich
- Extrem geringe Verlustleistung

Die Logik basiert auf dem physikalischen **Shannon-Dual-Rail-Bit** und der neuen skalierten **TM-Algebra** und unterstützt:

- NOR
- NAND
- AND
- NXOR
- NOT
- Identität

6. Das BFF: Funktionale Betriebsarten

Modus 1 – IN/OUT (sequentiell)

- Setzen
- Rücksetzen
- Halten
- Zustandsbehaftete Speicheroperation

Modus 2 – OUT/IN (algebraisch)

- Ausgang = algebraische Transformation
- Logikfunktionen werden über Steuervariable gewählt

Damit werden **Zustandsautomaten und kombinatorische Logik algebraisch vereinheitlicht**.

7. Implementierung auf Transistorebene

Komponente	Transmission Gates	Transistoren
Dual-Rail-AND	8	16
BFF (sequentiell)	16	32
BFF (algebraisch)	16	32

Eigenschaften:

- CMOS-kompatibel
- Keine exotischen Materialien erforderlich

8. Reversibilität, Involution & Thermodynamik

Das algebraische BFF unterstützt:

- Reversible Zustandsübergänge
- Involutive Transformationen
- Geringe Entropieproduktion
- Nahezu adiabatisches Schaltverhalten

Reversibles Computing erfordert eine bijektive Abbildung mit gleicher Anzahl von Ein- und Ausgangsvariablen, bei der eine eindeutige Inversfunktion existiert. Involutive Logik stellt eine strengere Unterklasse dar, bei der die Abbildung identisch mit ihrer eigenen Inversfunktion ist, so dass durch erneute Anwendung derselben Operation der ursprüngliche Zustand exakt wiederhergestellt wird (Gleiche Operation vor & zurück).

Der Unterschied (formal)

Eigenschaft	Reversibel	Involutiv
Anzahl Ein-/Ausgänge gleich	ja	ja
Eindeutig rückrechenbar	ja	ja
Separate Inversfunktion nötig	ja	nein
Gleiche Operation vor & zurück	nein	ja
$f(f(x)) = x$	nicht zwingend	immer
Physikalische Effizienz	hoch	maximal

Involutiv ist eine strengere Spezialform der Reversibilität.

Daraus resultieren:

- Ultrageringe Wärmeentwicklung
- Reduziertes thermisches Rauschen
- Hohe Langzeit-Betriebsstabilität
- Robustheit gegenüber Strahlung und Soft-Errors

9. Systemarchitektur

Ein TM–Dual–Rail-System besteht aus:

- Großen Arrays von BFF-Zellen
- Lokalen algebraischen Verknüpfungsstrukturen anstelle von ALUs
- Keinen globalen Bussen
- Keiner Cache-Hierarchie
- Keiner Trennung von Instruktionen und Daten

Das Gesamtsystem wird zu:

Einem kontinuierlichen algebraischen Rechengewebe mit eingebettetem Zustand.

10. Leistungsmerkmale

Latenz

Operation	**Klassisch**	**Dual-Rail**
Speicherzugriff	10–30 Takte	1–2 Takte
Logikoperation	1–4 Takte	1 Takt
Load/Store	20–100 Takte	eliminiert

Energie

- Klassisch: 10–100 pJ pro Operation
- TM–Dual–Rail: ca. **2–3 pJ pro Operation**

Dichte

- **5x–20x höhere effektive Rechendichte pro mm^2**

11. Determinismus & Sicherheit

TM–Dual–Rail bietet:

- Vollständig deterministisches, nachvollziehbares Zeitverhalten
- Keine Cache-Unvorhersagbarkeit
- Keine Bus-Arbitrierung
- Keine Race-Conditions durch gemischte Speicherhierarchien

Ideal geeignet für:

- ASIL- (Automotive Safety Integrity Level), D-Automobilsysteme

- SIL-4-Industrieanwendungen
- Medizinische Sicherheitssysteme
- Kerntechnik & Luft-/Raumfahrt

12. Validierung durch SPICE-Simulation

Extern validiert wurden:

- Dual-Rail-AND mit CNOT/XOR-Implementierung
- IN/OUT Sequentielles BFF
- OUT/IN Algebraisches BFF 1 und Algebraisches BFF 2
- Zustandsübergänge
- Stabile Zustand Speicherung

Ergebnisse:

- Elektrische Korrektheit bestätigt
- Algebraische Wahrheitstabellen verifiziert
- Keine Metastabilität festgestellt
- Die BFF-Funktionen Setzen, Rücksetzen und Speichern sind jeweils eindeutig durch entsprechende Zeilen in der Flip-Flop-Wertetabelle definiert. Diese Zeilen sind sowohl im IN/OUT-Betrieb als auch im OUT/IN-Betrieb identisch. Der funktionale Unterschied der beiden Betriebsarten ergibt sich ausschließlich aus der jeweiligen Zuordnung und Variation der Ein- und Ausgangsvariablen.
- Alternative Steuerungsmöglichkeiten der BFF sind möglich.

13. Auswirkungen auf KI & Maschinelles Lernen

Transformation der Maschinensprache in Algebra:

Die Maschinensprache – und damit auch Algorithmen und Software – lassen sich systematisch in algebraische Strukturen überführen und physikalisch direkt im Dual-Rail-Rechengewebe abbilden.

Eliminierung speichergebundener Trainingsprozesse:

Durch die Vereinigung von Speicher und Rechnen im BFF entfallen klassische, speichergebundene Trainingsprozesse, die heute durch Datenbewegung limitiert sind.

Hochlokalisierte Inferenz:
KI-Berechnungen (Inferenz) erfolgen direkt am Ort der Datenerzeugung – im Sensor, im Gerät oder im Edge-System – ohne Übertragung in zentrale Rechenzentren oder die Cloud. Dadurch werden Latenz, Energieverbrauch und Datenschutzrisiken drastisch reduziert.

Wegfall der HBM-Abhängigkeit:

Die Architektur macht Hochleistungs-KI unabhängig von externem High-Bandwidth Memory (HBM), das heute den dominierenden Energie- und Kostenfaktor moderner KI-Beschleuniger darstellt.

On-Device-LLMs:
Große Sprachmodelle (LLMs) können direkt auf Endgeräten und Edge-Systemen betrieben werden – ohne Cloud-Anbindung, mit deterministischem Zeitverhalten und hoher Energieeffizienz.

Wirtschaftliche KI-Skalierung:
Die Architektur ermöglicht durch höhere Boolesche Skalierung erstmals eine ökonomisch und energetisch skalierbare Umsetzung von KI (Artificial Intelligence), also Künstlicher Intelligenz.

14. Auswirkungen auf Automotive

- ADAS (Advanced Driver Assistance Systems) - in 1–2 Taktzyklen
- Zentrale Fahrzeug-HPC bei geringerer Leistungsaufnahme
- KI-Modelle direkt auf ECUs (Electronic Control Unit)
- Deterministische Sicherheitspipelines
- Erhöhte Reichweite von Elektrofahrzeugen durch geringeren Rechenenergiebedarf

15. Telekommunikation, Edge & 6G

- Ultraniedrig-latenzfähiges RAN (Radio Access Network)-Scheduling
- Edge-KI ohne thermische Überlastung
- MEC (Mobile Edge Computing). **Dezentrales Rechnen direkt am Rand des Mobilfunknetzes**, nahe bei den Endgeräten, statt im entfernten Cloud-Rechenzentrum. -Beschleunigung
- Netzwerksicherheit inline auf Paketebene
- Deterministische URLLC (Ultra-Reliable Low-Latency Communication) - Unterstützung

16. Industrieautomatisierung & Prozessregelung

- Ersatz klassischer SPS (Speicherprogrammierbare Steuerung / Programmable Logic Controller) - Systeme
- Echtzeit-Robotik
- Prädiktive Wartung auf Sensorebene
- Deterministische DCS-Rechenleistung (DCS: Distributed Control System / Verteiltes Leitsystem)
- Edge-Steuerung ohne Cloud-Abhängigkeit

17. Medizintechnik, Biotechnologie & Pharma

- Portable Diagnostik
- Echtzeit-Bildgebung
- Beschleunigte Sequenzierung
- Deterministische OP-Robotik
- On-Device KI, HIPAA (Health Insurance Portability and Accountability Act) - & DSGVO (Datenschutz-Grundverordnung) - konform

18. Energie, Öl & Gas

- Turbinen- & Netzregelung
- Pipeline-Monitoring
- Inline-chemische Analytik
- Explosionsgeschützte Ultra-Low-Heat-Computer-Systeme
- 24/7-Langzeit-Controller mit extrem hoher Zuverlässigkeit

19. Fertigungs- & Lizenzierungsmodell

Die TM–Dual–Rail-Technologie adressiert zwei grundsätzliche Anwendungsfälle:

a.) Hardware-Anwendung & Hardware-Optimierung

Überführung und Optimierung kundenspezifischer Chips auf Basis der TM–Dual–Rail-Architektur.

b.) Software-, Algorithmus- & Rechenmodell-Transformation

Überführung bestehender Algorithmen und Software in algebraische Dual–Rail-Strukturen zur massiven Effizienzsteigerung.

Geschäfts- und Lizenzierungsmodell

- Architekturlizenzierung
  Lizenzierung der gesamten Dual Rail TM–Dual–Rail-Architektur zur direkten Integration in kundenspezifische Chips (ASICs / SoCs).
- IP-Cores (Funktionsblöcke)
  Vorgefertigte, lizenzierbare Bausteine
- Branchenspezifische Varianten

20. Fazit & Ausblick

Visionäre Perspektive – Der Beginn eines neuen Computerzeitalters

TM–Dual–Rail Computing markiert **einen fundamentalen Umbruch in der Computerphysik** – nicht als inkrementelle Verbesserung, sondern als **erste wirklich neue Rechenarchitektur seit über 70 Jahren**. Wo bisher Speicher und Berechnung physikalisch getrennt waren, verschmelzen sie nun in **einem einzigen algebraischen, physikalischen Grundbaustein**. Damit wird ein Grundprinzip beendet, das seit den Anfängen der digitalen Rechentechnik alle Systeme limitiert hat: der **Von-Neumann-Flaschenhals**.

Diese Architektur bedeutet nicht weniger als eine **Neudefinition dessen, was "Rechnen" physikalisch ist**:

- **Vereinigung von sequenzieller Zustandslogik (Flip-Flops) und algebraischer Berechnung in einer einzigen Struktur**
- **Rückführung der gesamten Maschinensprache auf reine Algebra**
- **Transformation von Algorithmen und Software in physikalisch berechenbare algebraische Zustandsräume. Software/Algorithmen werden algebraisch**
- **Überführung bestehender Single-Rail-Hardwarearchitekturen in die neue Dual-Rail-Technologie**
- **Ein neuer Skalierungspfad für Rechenleistung – jenseits klassischer Takt- und Transistorgrenzen**
- **Energieeinsparungen um Größenordnungen**
- **Deterministische Echtzeitsysteme als Standard statt Ausnahme**
- **Völlig neue KI-Einsatzparadigmen – vom Sensor bis zur planetaren Infrastruktur**
- **Nachvollziehbare, physikalisch erklärbare und skalierte neue Rechenfähigkeiten**

TM–Dual–Rail ist damit nicht nur eine neue Architektur –
es ist der **Übergang von softwaregetriebenem Rechnen zu physikalisch-algebraischem Rechnen**.

Es ist der Beginn einer neuen Epoche der digitalen Systeme:
schneller, deterministischer, nachvollziehbarer, energieärmer – und erstmals wieder physikalisch skalierbar.

21. Zeichnungen

Die wesentlichen technischen Aussagen der obigen Ausführungen lassen sich in Zeichnungen darstellen.

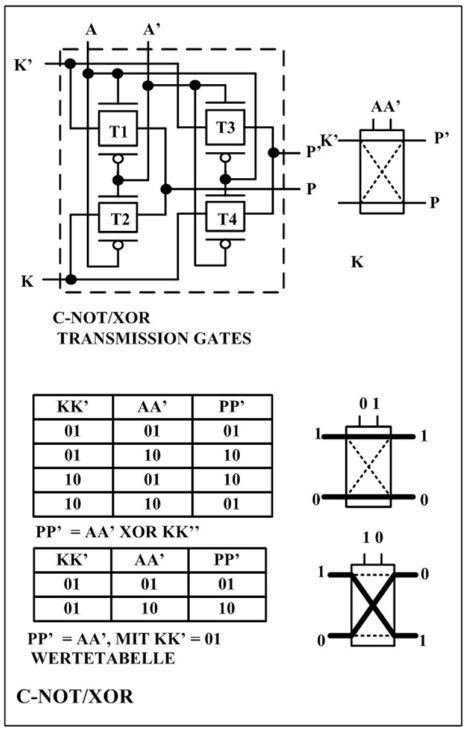
Die Figur 1 zeigt das skalierte Boolesche CNOT/XOR.

Das Dual Rail CNOT/XOR: PP‘ = AA‘ XOR KK‘ ist aus 4 Transmission Gate aufgebaut. Die Eingangsvariable sind KK‘ und AA‘. Die Ausgangsvariable ist PP‘. Bei AA‘ = 01 werden die Signale KK‘ an den Ausgang PP‘ durchgeleitet. Bei AA‘ = 10 werden die Signalpfade vertauscht.

Mit KK‘ = 01 = konstant wird das CNOT/XOR zum Dual Rail Schalter PP‘ = AA‘.

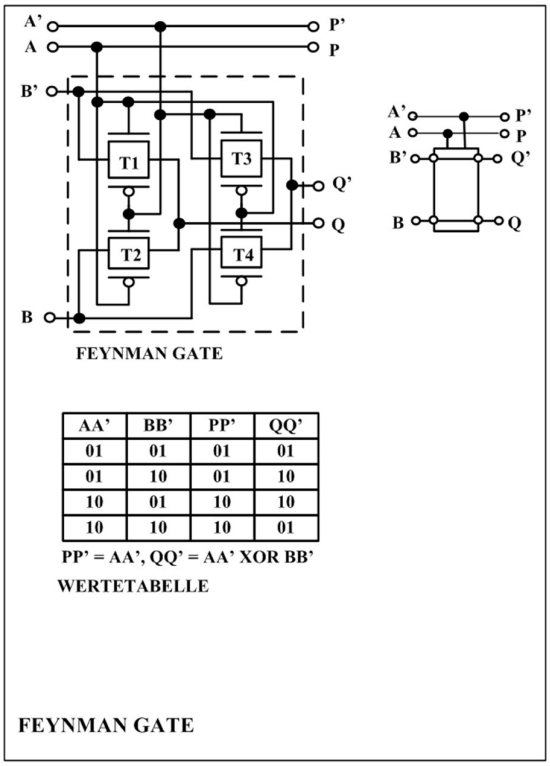
Diese Funktionalität kann durch ein einfaches Symbol dargestellt werden.

Ein komplementäres Signal verbindet oder vertauscht zwei Leitungen. Dies macht das CNOT/XOR zu einem Booleschen Schalter der Dimension 2, Dual Rail.



Das Dual Rail Feynman Gate: PP' = AA', QQ' = AA' XOR BB' ist eine Variation des Dual Rail CNOT/XOR. Die Eingangsvariablen sind AA' und BB', die Ausgangsvariablen sind PP' und QQ'. Die Eingangsvariable AA' wird an den Ausgang geführt. Die Anzahl der Ein und Ausgangsvariablen ist gleich. In der Darstellung entspricht dies der Funktion von links (L), (Eingänge) nach rechts (Ausgänge), (A). Das Feynman Gate (LR) ist in diesem Betrieb reversibel.

Das Feynman Gate kann auch von rechts nach links betrieben werden (RL), **AA' = PP', BB' = AA' XOR QQ'.** Die Eingangsvariablen sind PP' und QQ', die Ausgangsvariablen sind AA' und BB'. Dadurch wird das Dual Rail Feynman Gate **involutisch.** Involutive Logik stellt eine strengere Unterklasse dar, bei der die Abbildung identisch mit ihrer eigenen Inversfunktion ist, so dass durch erneute Anwendung derselben Operation der ursprüngliche Zustand exakt wiederhergestellt wird (Gleiche Operation vor & zurück).



FEYNMAN GATE

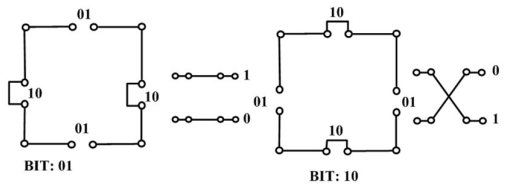
AA'	BB'	PP'	QQ'
01	01	01	01
01	10	01	10
10	01	10	10
10	10	10	01

PP' = AA', QQ' = AA' XOR BB'

WERTETABELLE

FEYNMAN GATE

Das CNOT/XOR als Schalter lässt sich in eine mechanische Version transformieren, wodurch das **physikalische Dual Rail Shannon Bit** bestimmt wird. Dies ist das erste skalierte BIT (01,10), es ist eine Konkatenation (Zusammenfassung) der einstelligen komplementären BIT (0,1) und (1,0), es ist eine Metastruktur. Das Dual-Rail-System bildet damit das formal einfachste Zweiteilchensystem, das zu einem koordinierten, algebraisch steuerbaren Zustandswechsel fähig ist.



BIT: 01

BIT: 10

SYMBOL

MERKMALE:

PHYSICALISCHES SHANNON DUAL RAIL BIT

DAS ERSTE SKALIERTE BIT

PHYSIKALISCHE KONKANTENATION VON ZWEI ELEMENTEN.

ERSTE SKALIERTE METASTRUKTUR.

ZWEI KOMPLEMENTÄRE ELEMENTE ERZEUGEN DAS BIT

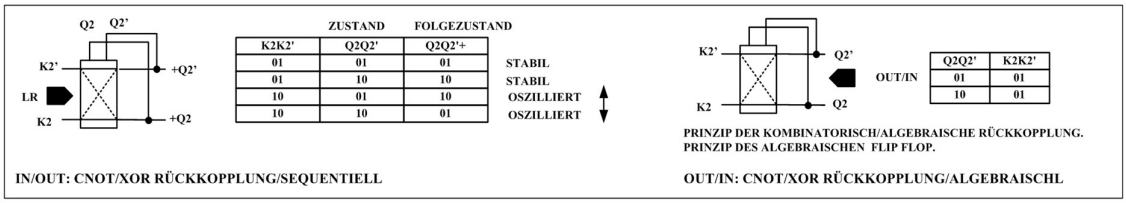
DAS EINFACHSTE PHYSIKALISHE ZWEI TEILCHEN SYSTEM DAS ZU EINEM KOORDINIERTEN ZUSTANDSWECHSEL IN DER LAGE IST

PHYSIKALISCHES DUAL RAIL SHANNON BIT

IN/OUT: CNOT/XOR RÜCKKOPPLUNG/SEQUENZIELL. Als rückgekoppelte Funktion Q2Q2‘+ = Q2Q2‘ XOR K2K2‘ verdeutlicht das CNOT/XOR am einfachsten die algebraische Form der

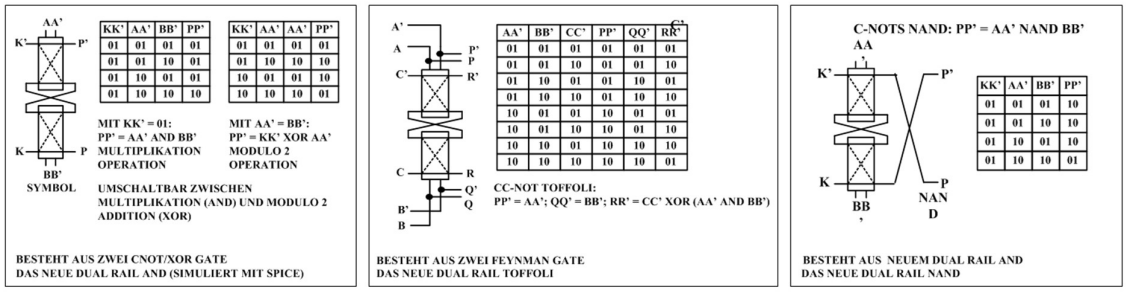
Rückkopplung. Bei K2K2‘ ist diese Funktion stabil, bei K2K2‘ = 10 oszilliert sie. Sie ist eine sequenzielle Struktur.

Betreibt man Sie jedoch umgekehrt OUT/IN: K2K2‘ = Q2Q2‘+ XOR Q2Q2‘ so ergibt sich K2K2‘ = 01, eine algebraische Funktion.

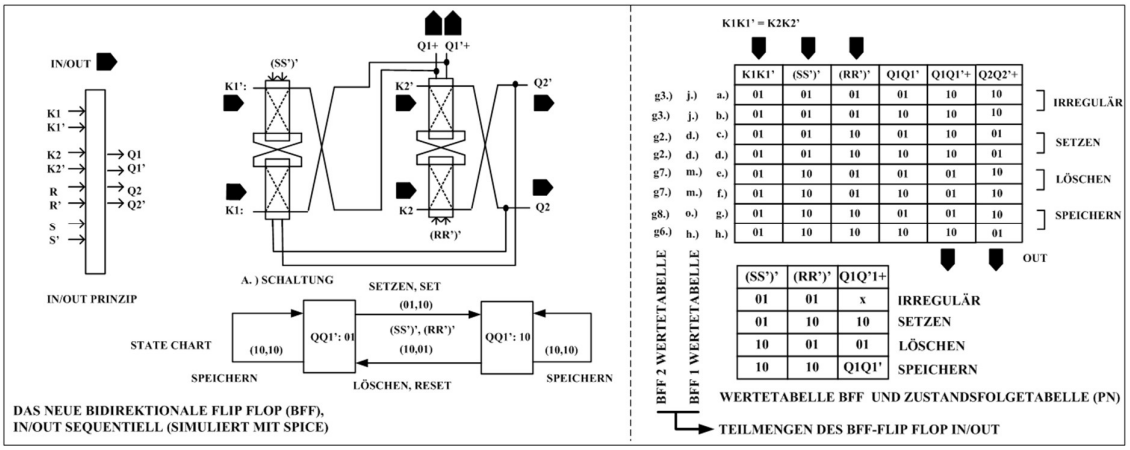


Diese Prinzipien sind die Grundlage für das neue bidirektionale Flip-Flop (BFF), IN/OUT im sequenziellen Betrieb, der auch mit SPICE simuliert wurde, FIGUR 2.

Das Flipflop besteht aus zwei rückgekoppelten neuen Dual Rail NAND-Gatter, die wiederum aus dem neuen Dual Rail AND aufgebaut werden. Das reversible Dual Rail Toffoli kann aus zwei Feynman Gate aufgebaut werden.

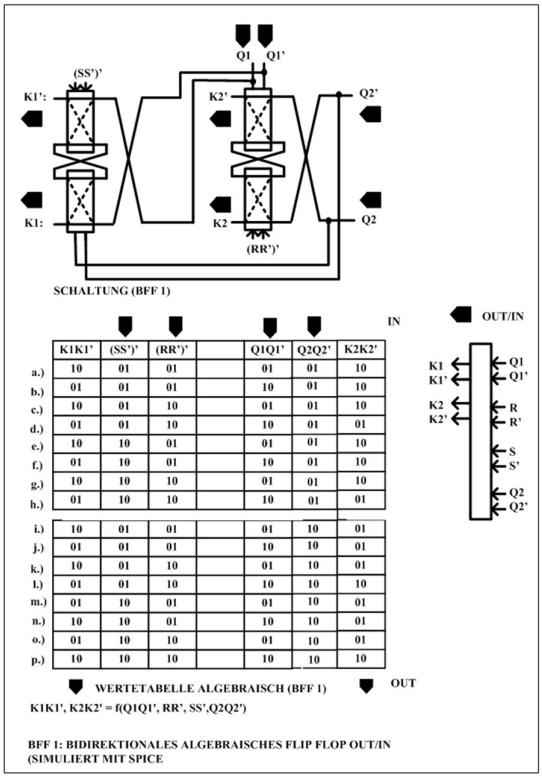


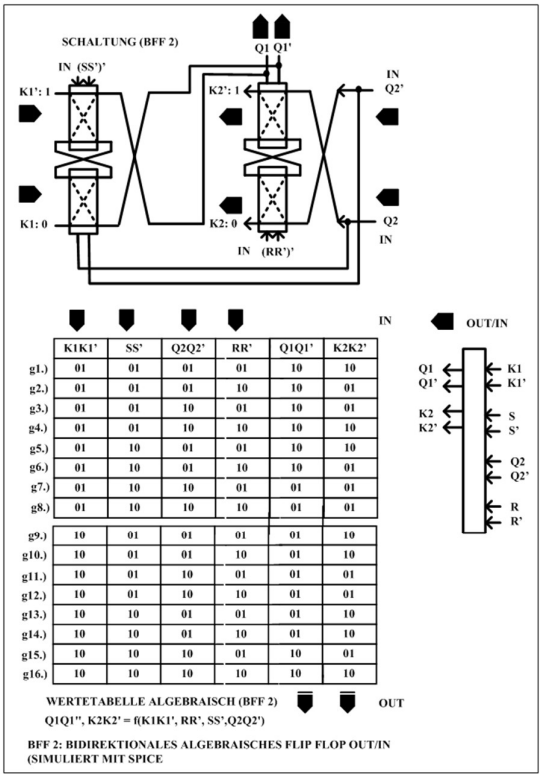
Die SPICE Simulation bestätigt die grundlegende Funktion des Flip-Flops (Setzen, Löschen Speichern) wie es im State Chart und der Wertetabelle dargestellt wird.



Die Zeilen a.) bis h.) der Wertetabelle können auf die Wertetabellen von BFF1 und BFF2 abgebildet werden, was durch eine Gegenüberstellung der Zeilen gezeigt wird. Sie sind

eine Teilmenge der algebraischen Flip-Flops BFF1 und BFF2 im OUT/IN Betrieb, wie Sie in der FIGUR 3 gezeigt werden.

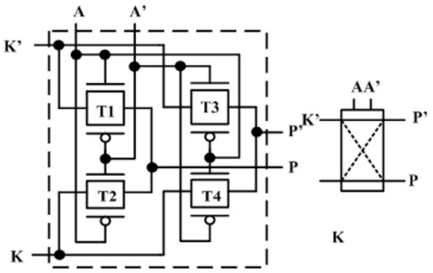




FIGUR 3

Die Umschaltung zwischen den Betriebsmodi des sequentiellen Flip Flop auf das algebraische BFF2 erfolgt durch das Vertauschen **der Ein- und Ausgangsvariablen, FIGUR 4**. Die Zeilen in den Wertetabellen für Setzen, Löschen und Halten (Speicherung) bleiben dabei in beiden Modi unverändert.

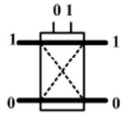
Durch das Vertauschen der Ein/Ausgangsvariablen werden die unterschiedlichen algebraischen Flip Flops (BFF1, BFF2,BFF3) erzeugt. Die 24 algebraischen Funktionen mit 2 Ein und 2 Ausgangsvariablen erhält man mit 2 Steuervariablen, **FIGUR 5**, **FIGUR 6** und **FIGUR 7**.



C-NOT/XOR
TRANSMISSION GATES

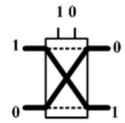
KK'	AA'	PP'
01	01	01
01	10	10
10	01	10
10	10	01

PP' = AA' XOR KK''

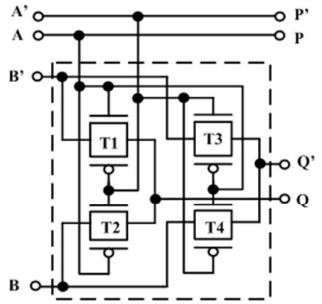


KK'	AA'	PP'
01	01	01
01	10	10

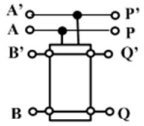
PP' = AA', MIT KK' = 01
WERTETABELLE



C-NOT/XOR



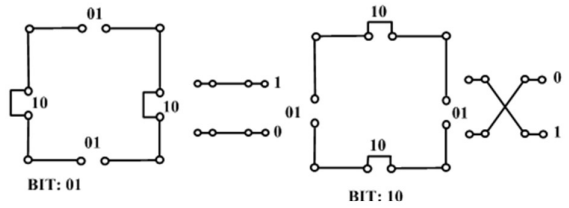
FEYNMAN GATE



AA'	BB'	PP'	QQ'
01	01	01	01
01	10	01	10
10	01	10	10
10	10	10	01

PP' = AA', QQ' = AA' XOR BB'
WERTETABELLE

FEYNMAN GATE



SYMBOL

MERKMALE:

PHYSICALISCHES SHANNON DUAL RAIL BIT

DAS ERSTE SKALIERTE BIT

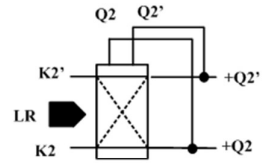
PHYSIKALISCHE KONKANTENATION VON ZWEI ELEMENTEN.

ERSTE SKALIERTE METASTRUKTUR.

ZWEI KOMPLEMENTÄRE ELEMENTE ERZEUGEN DAS BIT

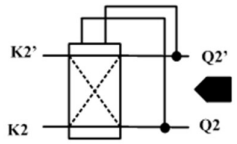
DAS EINFACHSTE PHYSIKALISHE ZWEI TEILCHEN SYSTEM DAS ZU EINEM KOORDINIERTEN ZUSTANDSWECHSEL IN DER LAGE IST

PHYSIKALISCHES DUAL RAIL SHANNON BIT



ZUSTAND		FOLGEZUSTAND	
K2K2'	Q2Q2'	Q2Q2'+	
01	01	01	STABIL
01	10	10	STABIL
10	01	10	OSZILLIERT
10	10	01	OSZILLIERT

IN/OUT: CNOT/XOR RÜCKKOPPLUNG/SEQUENTIELL



OUT/IN

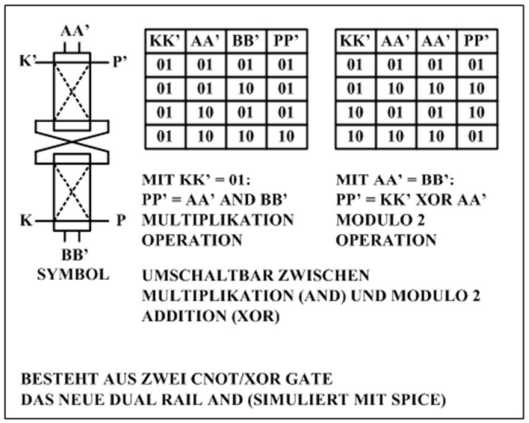
Q2Q2'	K2K2'
01	01
10	01

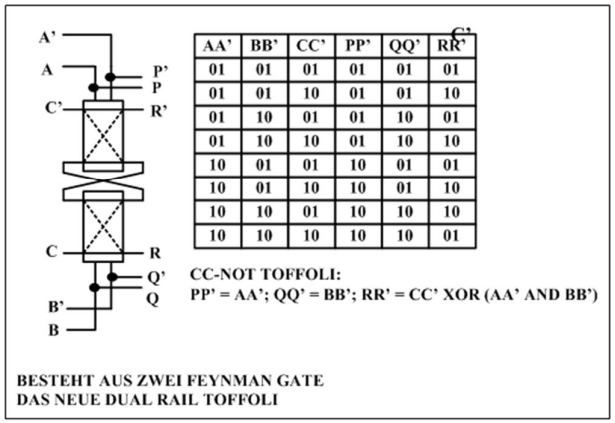
PRINZIP DER KOMBINATORISCH/ALGEBRAISCHE RÜCKKOPPLUNG.
PRINZIP DES ALGEBRAISCHEN FLIP FLOP.

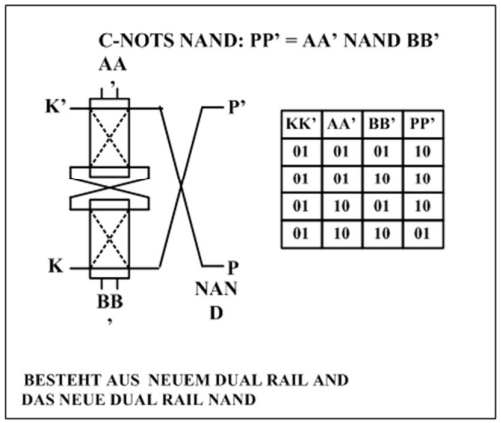
OUT/IN: CNOT/XOR RÜCKKOPPLUNG/ALGEBRAISCHL

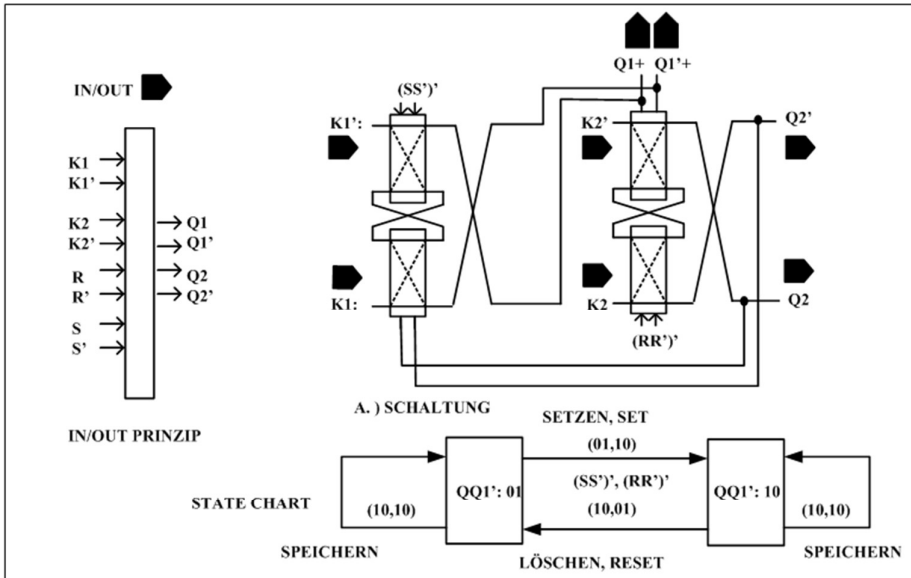
SKALIERTES BOOLESCHE CNOT/XOR, DATE: 07-12-2025

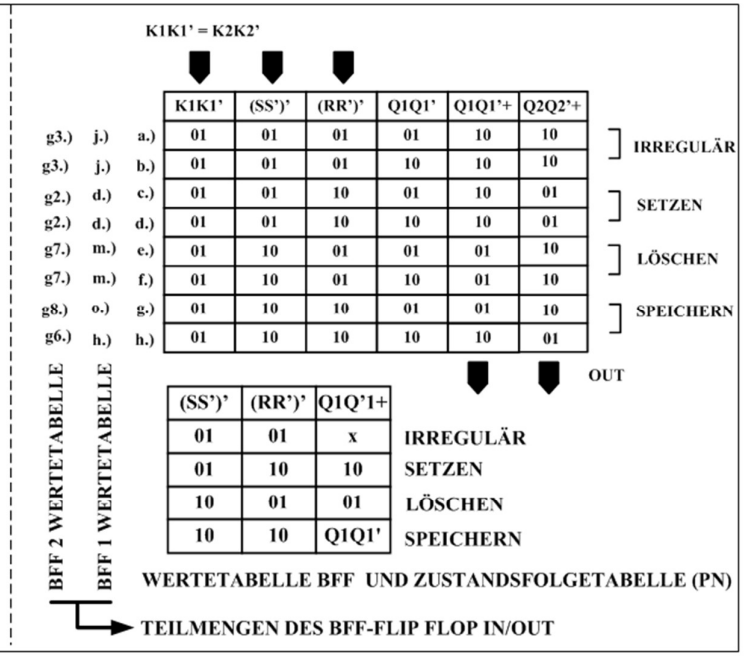
FIGUR 1


AA'
K'
P'
K
P
BB'
SYMBOL
KK' AA' BB' PP'
01 01 01 01
01 01 10 01
01 10 01 01
01 10 10 10
MIT KK' = 01:
PP' = AA' AND BB'
MULTIPLIKATION
OPERATION
KK' AA' AA' PP'
01 01 01 01
01 10 10 10
10 01 01 10
10 10 10 01
MIT AA' = BB':
PP' = KK' XOR AA'
MODULO 2
OPERATION
UMSCHALTBAR ZWISCHEN
MULTIPLIKATION (AND) UND MODULO 2
ADDITION (XOR)
BESTEHT AUS ZWEI CNOT/XOR GATE
DAS NEUE DUAL RAIL AND (SIMULIERT MIT SPICE)

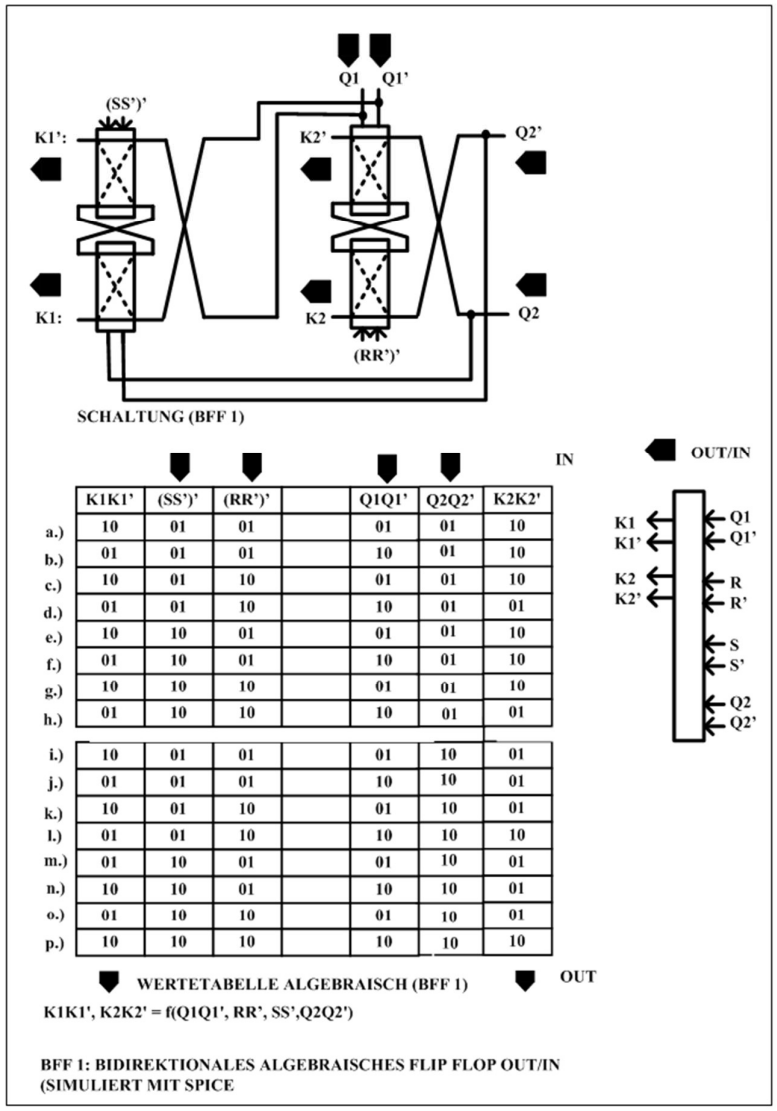

A'
A
C'
C
B'
B
P'
P
R'
R
Q'
Q
C'
AA' BB' CC' PP' QQ' RR'
01 01 01 01 01 01
01 01 10 01 01 10
01 10 01 01 10 01
01 10 10 01 10 10
10 01 01 10 01 01
10 01 10 10 01 10
10 10 01 10 10 10
10 10 10 10 10 01
CC-NOT TOFFOLI:
PP' = AA'; QQ' = BB'; RR' = CC' XOR (AA' AND BB')
BESTEHT AUS ZWEI FEYNMAN GATE
DAS NEUE DUAL RAIL TOFFOLI


C-NOTS NAND: PP' = AA' NAND BB'
AA'
K'
P'
K
P
BB'
NAND
KK' AA' BB' PP'
01 01 01 10
01 01 10 10
01 10 01 10
01 10 10 01
BESTEHT AUS NEUEM DUAL RAIL AND
DAS NEUE DUAL RAIL NAND

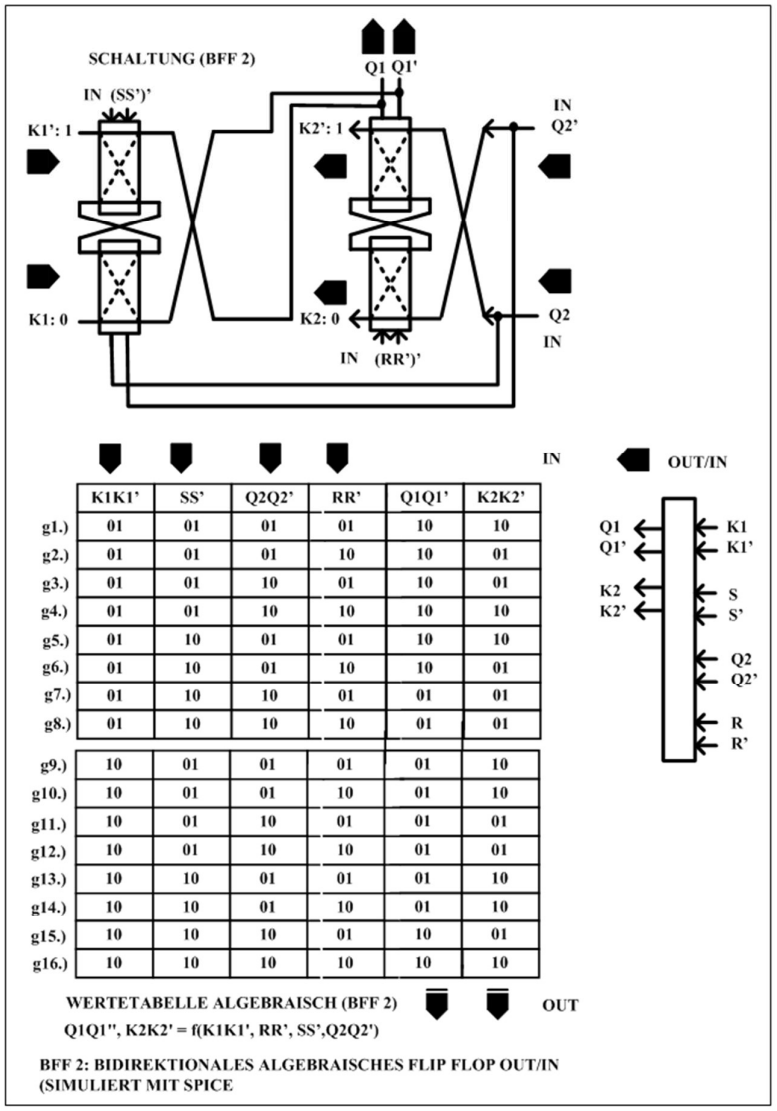

IN/OUT
K1
K1'
K2
K2'
R
R'
S
S'
Q1
Q1'
Q2
Q2'
IN/OUT PRINZIP
Q1+ Q1'+
(SS')'
K1':
K2'
Q2'
K1:
K2
Q2
(RR')'
A.) SCHALTUNG
SETZEN, SET
(01,10)
STATE CHART
(10,10)
QQ1': 01
(SS')', (RR')'
(10,01)
QQ1': 10
(10,10)
SPEICHERN
LÖSCHEN, RESET
SPEICHERN
DAS NEUE BIDIREKTIONALE FLIP FLOP (BFF),
IN/OUT SEQUENTIELL (SIMULIERT MIT SPICE)


K1K1' = K2K2'
K1K1' (SS')' (RR')' Q1Q1' Q1Q1'+ Q2Q2'+
g3.) j.) a.) 01 01 01 01 10 10
g3.) j.) b.) 01 01 01 10 10 10
g2.) d.) c.) 01 01 10 01 10 01
g2.) d.) d.) 01 01 10 10 10 01
g7.) m.) e.) 01 10 01 01 01 10
g7.) m.) f.) 01 10 01 10 01 10
g8.) o.) g.) 01 10 10 01 01 10
g6.) h.) h.) 01 10 10 10 10 01
IRREGULÄR
SETZEN
LÖSCHEN
SPEICHERN
OUT
BFF 2 WERTETABELLE
BFF 1 WERTETABELLE
(SS')' (RR')' Q1Q'1+
01 01 x IRREGULÄR
01 10 10 SETZEN
10 01 01 LÖSCHEN
10 10 Q1Q1' SPEICHERN
WERTETABELLE BFF UND ZUSTANDSFOLGETABELLE (PN)
TEILMENGEN DES BFF-FLIP FLOP IN/OUT

FIGUR 2

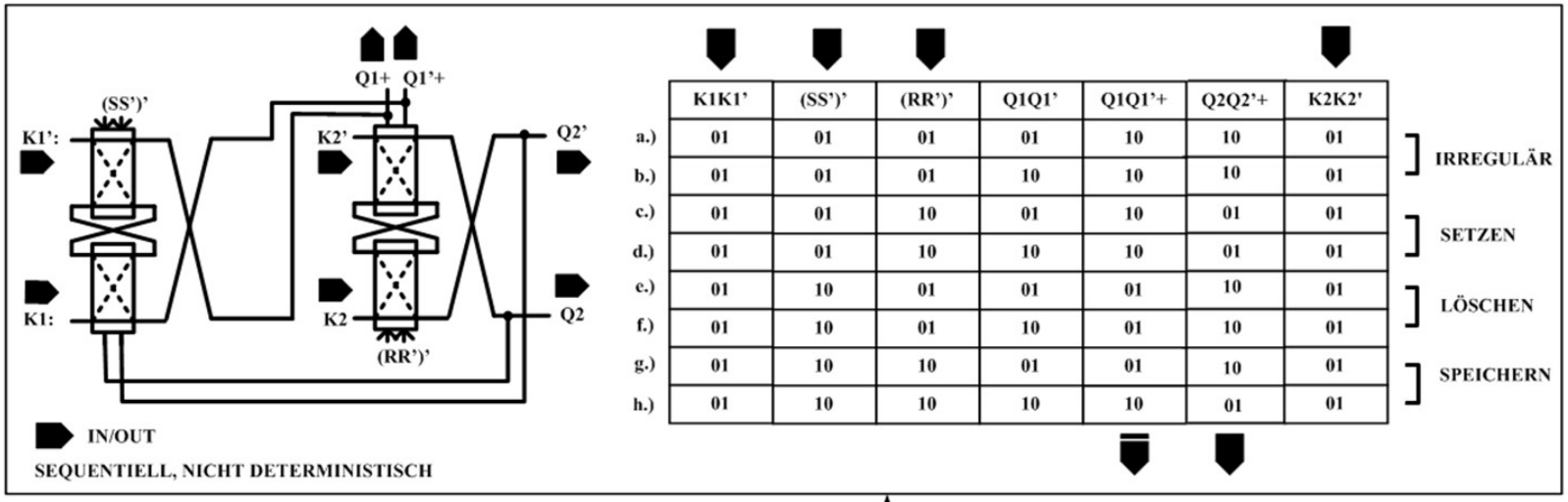


	K1K1'	(SS')'	(RR')'		Q1Q1'	Q2Q2'	K2K2'
a.)	10	01	01		01	01	10
b.)	01	01	01		10	01	10
c.)	10	01	10		01	01	10
d.)	01	01	10		10	01	01
e.)	10	10	01		01	01	10
f.)	01	10	01		10	01	10
g.)	10	10	10		01	01	10
h.)	01	10	10		10	01	01
i.)	10	01	01		01	10	01
j.)	01	01	01		10	10	01
k.)	10	01	10		01	10	01
l.)	01	01	10		10	10	10
m.)	01	10	01		01	10	01
n.)	10	10	01		10	10	01
o.)	01	10	10		01	10	01
p.)	10	10	10		10	10	10



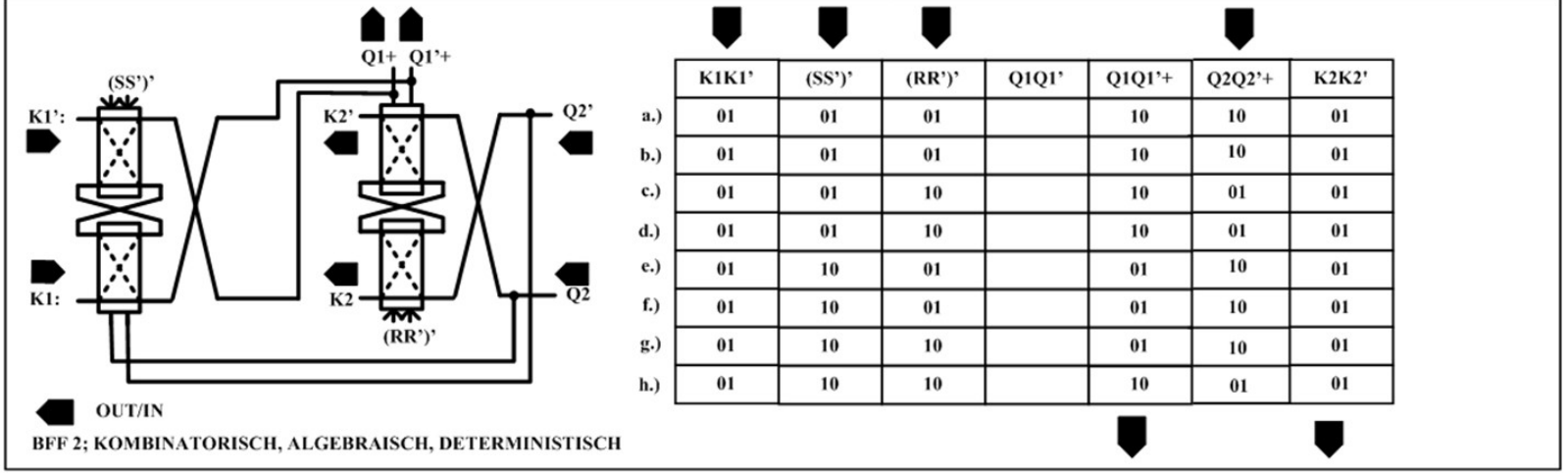
	K1K1'	SS'	Q2Q2'	RR'	Q1Q1'	K2K2'
g1.)	01	01	01	01	10	10
g2.)	01	01	01	10	10	01
g3.)	01	01	10	01	10	01
g4.)	01	01	10	10	10	10
g5.)	01	10	01	01	10	10
g6.)	01	10	01	10	10	01
g7.)	01	10	10	01	01	01
g8.)	01	10	10	10	01	01
g9.)	10	01	01	01	01	10
g10.)	10	01	01	10	01	10
g11.)	10	01	10	01	01	01
g12.)	10	01	10	10	01	01
g13.)	10	10	01	01	01	10
g14.)	10	10	01	10	01	10
g15.)	10	10	10	01	10	01
g16.)	10	10	10	10	10	10

FIGUR 3



	K1K1'	(SS')'	(RR')'	Q1Q1'	Q1Q1'+	Q2Q2'+	K2K2'	
a.)	01	01	01	01	10	10	01	IRREGULÄR
b.)	01	01	01	10	10	10	01	
c.)	01	01	10	01	10	01	01	SETZEN
d.)	01	01	10	10	10	01	01	
e.)	01	10	01	01	01	10	01	LÖSCHEN
f.)	01	10	01	10	01	10	01	
g.)	01	10	10	01	01	10	01	SPEICHERN
h.)	01	10	10	10	10	01	01	

UMSCHALTBAR



	K1K1'	(SS')'	(RR')'	Q1Q1'	Q1Q1'+	Q2Q2'+	K2K2'
a.)	01	01	01		10	10	01
b.)	01	01	01		10	10	01
c.)	01	01	10		10	01	01
d.)	01	01	10		10	01	01
e.)	01	10	01		01	10	01
f.)	01	10	01		01	10	01
g.)	01	10	10		01	10	01
h.)	01	10	10		10	01	01

FIGUR 4

WERTETABELLE ALGEBRAISCH (BFF 1)

IN

	K1K1'	(SS')'	(RR')'		Q1Q1'	Q2Q2'	K2K2'
a.)	10	01	01		01	01	10
b.)	01	01	01		10	01	10
c.)	10	01	10		01	01	10
d.)	01	01	10		10	01	01
e.)	10	10	01		01	01	10
f.)	01	10	01		10	01	10
g.)	10	10	10		01	01	10
h.)	01	10	10		10	01	01
i.)	10	01	01		01	10	01
j.)	01	01	01		10	10	01
k.)	10	01	10		01	10	01
l.)	01	01	10		10	10	10
m.)	01	10	01		01	10	01
n.)	10	10	01		10	10	01
o.)	01	10	10		01	10	01
p.)	10	10	10		10	10	10

OUT

BFF1: EIN/2 AUSGANGSVARIABLE
K1K1', K2K2' = f(Q1Q1', RR', SS' ,Q2Q2')

K2K2' = f (Q1Q1', RR') SS', Q2Q2': CONTROL	K1K1' = f (Q1Q1', RR') SS', Q2Q2': CONTROL
NAND: K2K2' = Q1Q1' NAND RR' SS' = Q2Q2' = 01	NOT Q1Q1': K1K1' = NOT Q1Q1', SS' = Q2Q2' = 01
NAND: K2K2' = Q1Q1' NAND RR' SS' = 10, Q2Q2' = 01	NOT Q1Q1': K1K1' = NOT Q1Q1', SS' = 10, Q2Q2' = 01
AND: K2K2' = Q1Q1' AND RR' SS' = 01, Q2Q2' = 10	NOT Q1Q1': K1K1' = NOT Q1Q1', SS' = 01, Q2Q2' = 10
AND: K2K2' = Q1Q1' AND RR' SS' = 10, Q2Q2' = 10	Q1Q1': K1K1' = Q1Q1', SS' = 10, Q2Q2' = 10

STEUERMÖGLICHKEITEN

4 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE: ANZAHL DER FUNKTIONEN: 2
3 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE, 1 STEUERVARIABLE: ANZAHL DER FUNKTIONEN: 4
2 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE, 2 STEUERVARIABLE: ANZAHL DER FUNKTIONEN: 8
1 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE, 3 STEUERVARIABLE: ANZAHL DER FUNKTIONEN:16

FIGUR 5 STEUERUNG BFF1)

WERTETABELLE ALGEBRAISCH (BFF 2)

IN

	K1K1'	SS'	Q2Q2'	RR'	Q1Q1'	K2K2'
g1.)	01	01	01	01	10	10
g2.)	01	01	01	10	10	01
g3.)	01	01	10	01	10	01
g4.)	01	01	10	10	10	10
g5.)	01	10	01	01	10	10
g6.)	01	10	01	10	10	01
g7.)	01	10	10	01	01	01
g8.)	01	10	10	10	01	01
g9.)	10	01	01	01	01	10
g10.)	10	01	01	10	01	10
g11.)	10	01	10	01	01	01
g12.)	10	01	10	10	01	01
g13.)	10	10	01	01	01	10
g14.)	10	10	01	10	01	10
g15.)	10	10	10	01	10	01
g16.)	10	10	10	10	10	10

OUT

BFF2: EIN/2 AUSGANGSVARIABLE

Q1Q1', K2K2' = f(RR', Q2Q', SS'. K1K1')

Q1Q1' = f (RR', Q2Q2') **K2K2' = f (RR', Q2Q2')**

K1K1', SS': CONTROL **K1K1', SS': CONTROL**

10: **Q1Q1'= 10** **K1K1' = SS' = 01**	**NXOR:** **K2K2' = RR' NXOR Q2Q2'** **K1K1' = SS' = 01**
NOT Q2Q2': **Q1Q1' = NOT Q2Q2'** **K1K1' = 01, SS' = 10**	**NOR:** **K2K2' = RR' NOR Q2Q2'** **K1K1' = 01, SS' = 10**
01: **Q1Q1' = 01** **K1K1' = 10, SS' = 01**	**NOT Q2Q2':** **K2K2' = NOT Q2Q2'** **K1K1' = 10, SS' = 01**
Q2Q2': **Q1Q1' = Q2Q2'** **K1K1' = 10, SS' = 01**	**NOT Q2Q2':** **K2K2' = NOT Q2Q2'** **K1K1' = 10, SS' = 01**

STEUERMÖGLICHKEITEN

4 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE: ANZAHL DER FUNKTIONEN: 2

3 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE, 1 STEUERVARIABLE: ANZAHL DER FUNKTIONEN: 4

2 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE, 2 STEUERVARIABLE: ANZAHL DER FUNKTIONEN: 8

1 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE, 3 STEUERVARIABLE: ANZAHL DER FUNKTIONEN:16

FIGUR 6 (STEUERUNG BFF2)

WERTETABELLE ALGEBRAISCH (BFF 3)

IN ↓ ↓ ↓ ↓

	K1K1'	SS'	Q2Q2'	RR'	Q1Q1'	K2K2'
g1.)	10	01	10	01	01	01
g2.)	10	01	10	01	10	01
g3.)	10	01	10	10	01	01
g4.)	10	01	01	10	10	01
g5.)	01	10	10	01	01	01
g6.)	10	10	10	01	10	01
g7.)	10	10	10	10	01	01
g8.)	10	10	10	10	10	01
g9.)	10	01	01	01	01	10
g10.)	01	01	01	01	10	10
g11.)	10	01	01	10	01	10
g12.)	01	01	10	10	10	10
g13.)	10	10	01	01	01	10
g14.)	10	10	01	01	10	10
g15.)	10	10	01	10	01	10
g16.)	10	10	10	10	10	10

OUT ↓ ↓

BFF3: 2 EIN/2 AUSGANGSVARIABLE
Q2Q2', K1K1' = f(Q1Q1'', RR', SS'. K2K2')

K1K1' = f (Q1Q1', RR')
K2K2', SS': CONTROL

10:
K1K1'= 10
SS' = K2K2' = 01

OR:
K1K1' = Q1Q1' OR RR'
SS' = 10, K2K2' = 01

NOT Q1Q1':
K1K1' = NOT Q1Q1'
SS' = 01, K2K2' = 10

10:
K1K1' = 10
SS' = K2K2' = 10

Q2Q2' = f (Q1Q1', RR'')
K2K2', SS': CONTROL

NAND:
Q2Q2' = Q1Q1' NAND RR''
SS' = K2K2' = 01

10:
Q2Q2' = 10
SS' = 10, K2K2' = 01

AND:
Q2Q2' = Q1Q1' AND RR'
SS' = 01, K2K2' = 10

AND:
Q2Q2' = Q1Q1' AND RR'
SS' = K2K2' = 10

STEUERMÖGLICHKEITEN

4 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE: ANZAHL DER FUNKTIONEN: 2
3 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE, 1 STEUERVARIABLE: ANZAHL DER FUNKTIONEN: 4
2 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE, 2 STEUERVARIABLE: ANZAHL DER FUNKTIONEN: 8
1 EINGANGSVARIABLE, 2 AUSGANGSVARIABLE, 3 STEUERVARIABLE: ANZAHL DER FUNKTIONEN:16

FIGUR 7 (STEUERUNG BFF3)