NEW SCALED TM DUAL-RAIL COMPUTER ARCHITECTURE
Memory (flip-flop) becomes the processor – the end of the von Neumann bottleneck
Software/algorithms become algebraic
Deterministic AI, minimal energy consumption, and safety-critical real time

Author: Peter Hettich
Patent Application: WO2025061229
Version: 2.0
Date: 18 -12-2025

Summary
The exponential increase in computational demand, driven by artificial intelligence, autonomous systems, industrial automation, telecommunications, and high-performance computing, has exposed the fundamental physical and architectural limits of classical computer systems. The separation of memory and compute units, as formalized in the von Neumann architecture, leads to excessive data movement, high latencies, thermal instability, and poor energy efficiency.

This white paper introduces **TM Dual-Rail Computing (Thue-Morse Dual-Rail Computing)**—a novel, scaled Boolean architecture based on a new fundamental logic element: **the bidirectional algebraic Dual-Rail Flip-Flop (BFF)**. The architecture unifies memory, control, and operation/computation within a single physical structure, thereby enabling minimal data movement, deterministic timing behavior, ultra-low energy consumption, and high computational density. Software and algorithms become algebraic. A typical application scenario for the TM Dual-Rail architecture is the safety-critical control loop, as found for example in autonomous systems, industrial automation, energy infrastructure, or telecommunications. This paper presents the theoretical foundations, transistor-level implementation, system architecture, performance characteristics, validated simulation results, and application domains including AI, automotive, industrial automation, telecommunications, medical technology, energy, and critical infrastructures.

1. Introduction
The End of Classical Scaling

For more than 70 years, performance gains in computing were achieved through process scaling (Moore's Law) and frequency scaling (Dennard scaling). Both mechanisms have now fundamentally slowed down:
• Transistor scaling no longer leads to proportional performance gains
• Frequency scaling is thermally limited
• Memory bandwidth grows significantly more slowly than compute performance
• Interconnect energy consumption dominates chip energy budgets

At the same time, AI workloads grow disproportionately with respect to data movement, memory accesses, and parameter counts. The result is a structural crisis in computational efficiency.

2. The von Neumann Bottleneck
Architecture as the Limiting Factor

In all classical architectures, the following applies:
• Memory is physically separated from the processor
• Every operation requires:

1. Loading from memory
2. Transport via interconnects
3. Computation in the ALU
4. Writing back to memory

This leads to:
• 70–90% of total energy consumption caused by data movement
• 10–30 clock cycles of latency per memory operation
• Dominance of interconnects in chip area and heat dissipation
• Scaling limits independent of transistor count

Modern GPUs, TPUs, and NPUs therefore remain fundamentally memory-bound systems.

3. Why AI, Industry & Infrastructure Break This Model

The following domains are structurally incompatible with classical architectures:
• Autonomous systems (real-time constraints)
• Medical imaging (deterministic safety-critical timing paths)
• Telecommunications & 5G/6G RAN (ultra-low latency at the edge)
• Industrial automation (24/7 deterministic control loops)
• Energy infrastructure (thermal and EMC constraints)

These application domains require:
• Deterministic timing behavior
• Low heat generation
• Highest reliability
• Extreme energy efficiency
• Local (on-device) computation

3.1 Typical Use Case – Deterministic AI in Safety-Critical Control Loops

A typical application of the TM Dual-Rail architecture is the safety-critical control loop, as found for example in autonomous systems, industrial automation, energy infrastructure, or telecommunications. Such systems can be formally described as finite-state machine models consisting of states, state transitions, and feedback. Classical architectures implement these automata by separating sequential storage (flip-flops) from combinational logic, which inevitably results in repeated load, process, and write-back cycles.

The TM Dual-Rail architecture replaces this separation with a homogeneous control-loop structure. The combinational blocks of an automaton model are implemented using algebraic Dual-Rail Flip-Flops (BFFs), which unify storage, state transition, and logical operation within a single physical structure. As a result, the control loop becomes a locally closed, deterministic state space.

The BFF can be operated bidirectionally: in IN/OUT mode, it functions as a sequential storage automaton; in OUT/IN mode, it operates as an algebraic function. Switching between these two modes is performed purely electrically by swapping input and output variables and can be completed in a single step. This allows the same control loop to react immediately to input signals as a reactor and to generate state changes as an actuator.

In this way, AI-based decision functions are integrated directly into the control loop. The decision is no longer a separate computational step, but rather a deterministic algebraic state transition with a guaranteed latency of only a few clock cycles and minimal energy consumption due to purely local switching. The complete algebraic description of the state spaces further enables formal verification and certifiability for safety-critical applications.

For system integration, the Dual-Rail control loop can be embedded as an encapsulated safety island within existing single-rail architectures. This achieves a significant increase in determinism, energy efficiency, and functional safety without fundamentally altering the existing system architecture.

4.TM Dual-Rail Computer: Architectural Overview

TM Dual-Rail Computing completely eliminates the separation between memory and logic. It represents the first scaled realization level of TM Computing and is based on the Boolean CNOT/XOR primitive. This primitive generates a new physical Dual-Rail Shannon bit (01, 10) as a concatenation of the two single-rail bits 0 and 1.

The new dual-rail system thus forms the formally simplest, yet structurally complex, two-particle system capable of coordinated, algebraically controllable state transitions.

The scaling and unification of combinational algebraic switching structures with sequential switching structures (flip-flops) lead to a new technological foundation in which storage, logic, and state machines are physically unified for the first time—constituting a fundamental unification of computer science.

Basic Principle:
Memory—the flip-flop—becomes the new processor (store, operate, control).

At the core of the architecture is the:
BFF – Bidirectional Algebraic Dual-Rail Flip-Flop

A single physical device supports **two operating modes**:
- **Sequential storage operations**
- **Combinational algebraic logic operations**

Switching between operating modes is performed purely electrically by simply swapping input and output variables. The performance advantage of the BFF primarily arises from eliminating classical data transfers between memory and compute units. In conventional technology, a state update (set/reset/hold) must at minimum pass through the sequence “read → combinational decision → write back.” In the BFF, storage, control, and operation occur within the same physical element. As a result, transfer and distribution delays (interconnects, fan-out, MUX gating), which often dominate cycle time in conventional data paths, are eliminated.

The rows in the truth tables for set, reset, and hold (storage) remain unchanged in both operating modes. In combinational algebraic mode, multiple variants are available by swapping

input/output variables (e.g., BFF1, BFF2, BFF3). Physically, the BFF is constructed from 16 transmission gates (32 transistors). Functionally, it provides four input variables and two output variables. Depending on the specific combination of input, output, and control variables, a high degree of functional diversity is achieved.

The functional capabilities of all algebraic BFFs can be summarized as follows:
• 4 input variables, 2 output variables: number of functions: 2
• 3 input variables, 2 output variables, 1 control variable: number of functions: 4
• 2 input variables, 2 output variables, 2 control variables: number of functions: 8
• 1 input variable, 2 output variables, 3 control variables: number of functions: 16

The electronic fundamental element of the BFF, as well as of the underlying CNOT/XOR, is the transmission gate—a proven electronic switching element used for decades. The operation of this component is technology-independent and can in principle be implemented in other physical realizations as well—such as physical, hydraulic, pneumatic, biological, or atomic systems—provided that a switch (IDENT) and its complement (NOT) can be realized.

4.1 Architectural Comparison

The BFF is compared with classical computer architectures (CPU, GPU, TPU, PIM). All classical architectures are characterized by a separation between memory and the processor (compute unit).

Explanation:

CPU (Central Processing Unit)
A universal processor for general-purpose programs; flexible, but energy- and memory-bound.

GPU (Graphics Processing Unit)
A massively parallel processor for executing many identical operations simultaneously (e.g., AI, graphics), but strongly memory-dependent.

TPU (Tensor Processing Unit)
A specialized processor for AI matrix operations; very fast, but limited by HBM memory accesses and only partially programmable.

HBM (High Bandwidth Memory)
An extremely fast, stacked high-performance memory located directly next to GPUs/TPUs, offering very high bandwidth but with high power consumption.

PIM (Processing In Memory)
Computation directly in memory reduces data movement, but is usually fixed-function and not freely programmable. In classical PIM, memory and compute units remain separate, merely placed closer together.

BFF – Bidirectional Algebraic Dual-Rail Flip-Flop
In the TM Dual-Rail BFF, storage (IN/OUT) and computation (OUT/IN) are unified within a single physical device.

The decisive difference of the BFF / TM Dual-Rail:

Architecture	Memory Cell	Compute Unit	Physically Identical?
CPU	Separate	Separate	No
GPU	Separate	Separate	No
TPU	Separate	Separate	No
PIM	On same chip	Adjacent	No
BFF / TM Dual-Rail	**Yes**	**Yes**	**Yes – identical**

Only in the BFF does the same physical structure serve as both memory and compute unit.

4.2 Fundamental Architectural Principles (Physical Perspective)

Architecture	Memory	Compute Unit	Physical Separation?	Fundamental Problem
CPU	External (cache/DRAM)	ALU/FPU	Yes	Data movement dominates
GPU	External (HBM)	Thousands of ALUs	Yes	Memory bandwidth limited
TPU	External (HBM)	Matrix units	Yes	HBM bottleneck
Classical PIM	In DRAM	Mini-ALUs	Partial	Fixed-function
BFF / TM Dual-Rail	**Identical to compute unit**	**Identical**	**No**	**None**

Only in the BFF are memory and compute physically the same component.

Abbreviations:

• Cache (Cache Memory), DRAM (Dynamic Random Access Memory)
Intermediate storage (cache) for extremely fast access within the processor; DRAM is the main working memory for programs and data.

• ALU (Arithmetic Logic Unit) / FPU (Floating Point Unit)
The ALU performs integer operations (addition, comparison, logic), while the FPU performs floating-point operations (e.g., for AI and physics).

• DRAM (Dynamic Random Access Memory)
Volatile main memory in which programs and data reside during operation (slower than cache, faster than SSD).

• HBM (High Bandwidth Memory)
An extremely fast, stacked high-performance memory located directly next to GPUs/TPUs, offering very high bandwidth but high power consumption.

4.3 Data Movement (Central Bottleneck of Modern Systems)

Architecture	Data Movement per Operation	Buses Required?	Cache Required?
CPU	Very high	Yes	Yes
GPU	Extremely high	Yes	Yes
TPU	Extremely high	Yes	Yes
Classical PIM	Reduced	Partial	Usually not
BFF / TM Dual-Rail	**Near 1**	**No**	**No**

4.4 Programmability & Computational Model

Architecture	General-Purpose Programmable?	State Machines?	Control Logic?	AI + Control on One System?
CPU	Yes	Yes	Yes	Yes
GPU	Yes	Poor	Poor	Separate systems
TPU	Strongly limited	No	No	No
Classical PIM	No	No	No	No
BFF / TM Dual-Rail	**Yes, fully algebraic**	**Yes, directly algebraic**	**Yes, directly in state**	**Yes, fully integrated**

The BFF integrates within a single device:
- Operator
- Memory
- State machine
- Control logic
- Large parts of the algorithms/software

4.5 Latency Comparison (Physically Induced)

Architecture	Memory Latency	Logic Latency	Total Path
CPU	10–30 cycles	1–3 cycles	20–100 cycles
GPU	200–800 cycles (HBM)	1–4 cycles	Very high
TPU	200–800 cycles (HBM)	1 cycle	Very high
Classical PIM	2–10 cycles	1–2 cycles	Medium
BFF / TM Dual-Rail	**1–2 cycles**	**1 cycle**	**Minimal**

For control loops, real-time AI, and safety-critical systems, this represents a quantitative leap.

Logic latency denotes the time (or number of clock cycles) required by a logical computing element (e.g., AND, NOT, ALU, gate) to transform an input signal into a stable output signal.

4.5.1 Formal Derivation of the Speed Advantage

In classical computer architectures, a state update of a control loop or finite-state machine model consists of several logically separated phases: reading the current state from memory, combinational evaluation of the transition function, and writing back the new state. In modern processor architectures, the time spent on data movement between memory, cache hierarchies, buses, and compute units often dominates the overall path, while the actual logical evaluation contributes only a comparatively small portion of the latency.

This relationship can be described formally by decomposing the total time required for a state update into a logic component and a transfer component. Let p denote the fraction of data movement in the total latency of classical architectures, and r the reduction factor of this transfer component achieved through the physical unification of memory and logic. The resulting speedup S can be approximated as:

$$S \approx 1 / ((1 - p) + p / r)$$

Since memory-bound architectures typically exhibit values of p in the range of 0.5 to more than 0.8, even moderate reduction factors result in speedups of several times. In the TM Dual-Rail BFF, the classical load/store cycle is completely eliminated, as storage, state transition, and logical operation occur within the same physical element. The state transition therefore takes place locally within one to two clock cycles, which systemically explains the quantitative leap observed in the latency comparison.

4.6 Energy per Operation

Architecture	Energy / Operation	Primary Loss Source
CPU	10–100 pJ	Buses + cache
GPU	50–300 pJ	HBM + interconnect
TPU	30–150 pJ	HBM
Classical PIM	5–20 pJ	Internal DRAM transport
BFF / TM Dual-Rail	**1–2 pJ**	**Local switching only**

This represents a fundamental physical advantage of the BFF architecture.

Physical Significance – Calculation of "Energy per Operation"

The future BFF consists of only 16 high-performance transmission gates, i.e., 32 transistors. Storage and computation therefore take place locally within exactly the same physical device. There are:

• No buses
• No cache hierarchies
• No external memory accesses
• No long interconnects
• No global data traffic

The switching energy arises exclusively from the local reconfiguration of these 32 transistors. This explains the expected ultra-low energy per operation of only 1–2 pJ.

Why Other Architectures Require So Much More Energy

CPU: 10–100 pJ
• Data must constantly be transported between:
– Cache
– DRAM
– ALU/FPU
over long interconnects
• Primary loss source: buses + cache
• Physically: large parasitic capacitances → high charge/discharge currents

GPU: 50–300 pJ
• Massive parallelism
• Constant access to HBM
• Very wide interconnects
• Primary loss source: HBM + interconnect
• Physically: high bandwidth = extremely high switching energy

TPU: 30–150 pJ
• Matrix compute units are efficient
• However: weights and activations are constantly fetched from HBM
• Primary loss source: HBM
• Physically: memory access dominates the energy budget

Classical PIM: 5–20 pJ
• Computation closer to memory
• However: compute unit still separated from the memory cell
• Internal DRAM transport remains necessary
• Primary loss source: internal DRAM transport

Why the BFF Operates at Only 1–2 pJ

For the BFF / TM Dual-Rail:
• Memory = compute unit
• Operation = local switching of 32 transistors
• No data transport over distances
• No buses
• No cache
• No external memory

• No arbitration
• No memory access in the classical sense

Bus arbitration is the decision mechanism that determines which of multiple simultaneously requesting devices is allowed to use a shared data bus first.

In classical CPU architectures, even a simple flip-flop set operation requires many clock cycles (20–100 cycles) for addressing, data loading, instruction execution, processing, and write-back. The dominant energy consumption arises from buses and cache/DRAM accesses. In the BFF, the same operation is performed as a direct local state transition within a single cycle, with minimal switching energy.

All energy is consumed solely for the local state transformation in the BFF.
This is the fundamental physical origin of the 10× to 100× efficiency advantage.

4.6.1 Formal Derivation of Energy per Operation

At a supply voltage in the range of **V = 0.6–0.8 V**, an energy consumption of **1–2 pJ per operation** corresponds to an effective switching capacitance of **$C_{e}ff \approx 1.6–5.6$ pF**. When mapped onto a BFF consisting of **16 transmission gates (32 transistors)**, this results in an equivalent effective capacitance of approximately **100–350 fF per transmission gate**, or **50–175 fF per transistor**.

This order of magnitude is consistent with modern CMOS low-capacitance logic and reflects exclusively local gate, diffusion, and wiring capacitances, without involvement of global interconnects or memory accesses.

4.7 Determinism & Safety

Architecture	Predictable Timing?	Cache Effects?	Bus Arbitration?	Safety Certifiability
CPU	No	Yes	Yes	Costly
GPU	No	Yes	Yes	Very difficult
TPU	No	Yes	Yes	Hardly possible
Classical PIM	Limited	Usually not	Partial	Difficult
BFF / TM Dual-Rail	**Yes, fully deterministic**	**No**	**No**	**Yes, optimal for ASIL/SIL**

The BFF enables classification according to **SIL (Safety Integrity Level)**.

4.8 Software Dependence

Architecture	Classical Software Required?	Compiler Complexity	Runtime Errors Possible?
CPU	Yes	Yes	Yes
GPU	Yes	Extreme	Yes
TPU	Specialized	Yes	Yes
Classical PIM	Hardly	Hardly	Still
BFF / TM Dual-Rail	**Strongly reduced**	**Algebraic**	**Massively reduced**

With the BFF, logic “migrates” from software into hardware state spaces.
Machine language—and thus algorithms and software—can be transformed into algebraic structures.

4.9 Strategic Positioning

• **CPU:** Flexible, but inefficient
• **GPU:** Massively parallel, but memory-bound
• **TPU:** AI-specialized, but limited by HBM
• **PIM:** Near-memory, but not freely programmable
• **BFF / TM Dual-Rail:** The first architecture to physically unify memory, computation, state, and control within a single element (a unification of computer science).

5. Dual-Rail Encoding & Algebraic Logic

The theoretical foundation is formed by the Boolean TM scaling of bits, variables, and functions.

Dual-Rail Symbols:
• Logical 0 → (01)
• Logical 1 → (10)

Properties:
• No invalid logic state
• Symmetric representation
• Reversible logic possible
• Extremely low power dissipation

The logic is based on the physical Shannon dual-rail bit and the new scaled TM algebra and supports:
• NOR
• NAND
• AND
• NXOR
• NOT
• Identity

6. The BFF: Functional Operating Modes

Mode 1 – IN/OUT (Sequential)
• Set
• Reset
• Hold
• Stateful storage operation

Mode 2 – OUT/IN (Algebraic)
• Output = algebraic transformation
• Logic functions selected via control variables

This algebraically unifies state machines and combinational logic.

7. Transistor-Level Implementation

Component	Transmission Gates	Transistors
Dual-Rail AND	8	16
BFF (sequential)	16	32
BFF (algebraic)	16	32

Properties:
• CMOS-compatible
• No exotic materials required

8. Reversibility, Involution & Thermodynamics

The algebraic BFF supports:
• Reversible state transitions
• Involutive transformations
• Low entropy production
• Nearly adiabatic switching behavior

Reversible computing requires a bijective mapping with an equal number of input and output variables, for which a unique inverse function exists. Involutive logic forms a stricter subclass, in which the mapping is identical to its own inverse, such that applying the same operation again exactly restores the original state (same operation forward and backward).

Formal Difference:

Property	Reversible	Involutive
Equal number of inputs/outputs	Yes	Yes
Uniquely invertible	Yes	Yes
Separate inverse function needed	Yes	No

Property	Reversible	Involutive
Same operation forward & back	No	Yes
f(f(x)) = x	Not required	Always
Physical efficiency	High	Maximal

Involutive logic is a stricter special case of reversibility.

This results in:
• Ultra-low heat generation
• Reduced thermal noise
• High long-term operational stability
• Robustness against radiation and soft errors

9. System Architecture

A TM Dual-Rail system consists of:
• Large arrays of BFF cells
• Local algebraic interconnection structures instead of ALUs
• No global buses
• No cache hierarchy
• No separation between instructions and data

The overall system becomes:
A continuous algebraic compute fabric with embedded state.

10. Performance Characteristics

Latency

Operation	Classical	Dual-Rail
Memory access	10–30 cycles	1–2 cycles
Logic operation	1–4 cycles	1 cycle
Load/Store	20–100 cycles	Eliminated

Energy
• Classical: 10–100 pJ per operation
• TM Dual-Rail: approx. 2–3 pJ per operation

Density
• 5×–20× higher effective compute density per mm^2

11. Determinism & Safety

TM Dual-Rail provides:
• Fully deterministic, traceable timing behavior
• No cache unpredictability
• No bus arbitration
• No race conditions due to mixed memory hierarchies

Ideally suited for:
• ASIL-D automotive systems (Automotive Safety Integrity Level)
• SIL-4 industrial applications
• Medical safety systems
• Nuclear engineering and aerospace

12. Validation via SPICE Simulation

Externally validated were:
• Dual-Rail AND with CNOT/XOR implementation
• IN/OUT sequential BFF
• OUT/IN algebraic BFF 1 and algebraic BFF 2
• State transitions
• Stable state storage

Results:
• Electrical correctness confirmed
• Algebraic truth tables verified
• No metastability observed
• The BFF functions set, reset, and store are each uniquely defined by corresponding rows in the flip-flop truth table. These rows are identical in both IN/OUT and OUT/IN operation. The functional difference between the two operating modes arises exclusively from the respective assignment and variation of input and output variables.
• Alternative control schemes for the BFF are possible.

13. Impact on AI & Machine Learning

Transformation of Machine Language into Algebra:
Machine language—and thus algorithms and software—can be systematically transformed into algebraic structures and physically mapped directly into the dual-rail compute fabric.

Elimination of Memory-Bound Training Processes:
By unifying memory and computation within the BFF, classical memory-bound training processes—currently limited by data movement—are eliminated.

Highly Localized Inference:
AI computations (inference) are performed directly at the point of data generation—within the sensor, the device, or the edge system—without transfer to centralized data centers or the cloud. This drastically reduces latency, energy consumption, and data privacy risks.

Elimination of HBM Dependency:
The architecture makes high-performance AI independent of external high-bandwidth memory (HBM), which today represents the dominant energy and cost factor in modern AI accelerators.

On-Device LLMs:
Large Language Models (LLMs) can be executed directly on end devices and edge systems—without cloud connectivity, with deterministic timing behavior and high energy efficiency.

Economic Scaling of AI:
Through higher Boolean scaling, the architecture enables—for the first time—an economically and energetically scalable implementation of artificial intelligence (AI).

14. Impact on Automotive

- ADAS (Advanced Driver Assistance Systems) with reaction times of 1–2 clock cycles
- Centralized vehicle HPC with significantly reduced power consumption
- AI models running directly on ECUs (Electronic Control Units)
- Deterministic safety pipelines
- Increased electric vehicle range due to lower computational energy demand

15. Telecommunications, Edge & 6G

- Ultra-low-latency RAN (Radio Access Network) scheduling
- Edge AI without thermal overload
- MEC (Mobile Edge Computing) acceleration – decentralized computation directly at the edge of the mobile network, close to end devices, instead of remote cloud data centers
- Inline network security at packet level
- Deterministic URLLC (Ultra-Reliable Low-Latency Communication) support

16. Industrial Automation & Process Control

- Replacement of classical PLC (Programmable Logic Controller) systems
- Real-time robotics
- Predictive maintenance at sensor level
- Deterministic DCS compute performance (DCS: Distributed Control System)
- Edge control without cloud dependency

17. Medical Technology, Biotechnology & Pharma

- Portable diagnostics
- Real-time imaging
- Accelerated sequencing
- Deterministic surgical robotics
- On-device AI compliant with HIPAA (Health Insurance Portability and Accountability Act) and GDPR

18. Energy, Oil & Gas

- Turbine and grid control
- Pipeline monitoring

• Inline chemical analytics
• Explosion-proof ultra-low-heat computing systems
• 24/7 long-term controllers with extremely high reliability

19. Manufacturing & Licensing Model

The TM Dual-Rail technology addresses two fundamental use cases:

a) Hardware Application & Hardware Optimization
Migration and optimization of customer-specific chips based on the TM Dual-Rail architecture.

b) Software, Algorithm & Computational Model Transformation
Migration of existing algorithms and software into algebraic dual-rail structures for massive efficiency gains.

Business and Licensing Model:
• **Architecture licensing** – licensing of the complete TM Dual-Rail architecture for direct integration into customer-specific chips (ASICs / SoCs)
• **IP cores (functional blocks)** – pre-fabricated, licensable building blocks
• **Industry-specific variants**

20. Conclusion & Outlook

Visionary Perspective – The Beginning of a New Computing Era

TM Dual-Rail Computing marks a fundamental paradigm shift in computer physics—not as an incremental improvement, but as the first truly new computing architecture in more than 70 years. Where memory and computation were previously physically separated, they now merge into a single algebraic, physical fundamental building block.

This ends a core principle that has limited all digital systems since the dawn of computing: the von Neumann bottleneck.

This architecture represents nothing less than a redefinition of what “computation” physically is:
• Unification of sequential state logic (flip-flops) and algebraic computation in a single structure
• Reduction of the entire machine language to pure algebra
• Transformation of algorithms and software into physically computable algebraic state spaces – software and algorithms become algebraic
• Migration of existing single-rail hardware architectures into the new dual-rail technology
• A new scaling path for compute performance beyond classical clock and transistor limits
• Orders-of-magnitude energy savings
• Deterministic real-time systems as the standard rather than the exception
• Entirely new AI deployment paradigms—from sensors to scale infrastructure
• Transparent, physically explainable, and scalable new computational capabilities

TM Dual-Rail is therefore not merely a new architecture—
it is the transition from software-driven computing to physical-algebraic computing.

It marks the beginning of a new era of digital systems:
faster, deterministic, transparent, far more energy-efficient—and, for the first time again, physically scalable.

21. Drawings

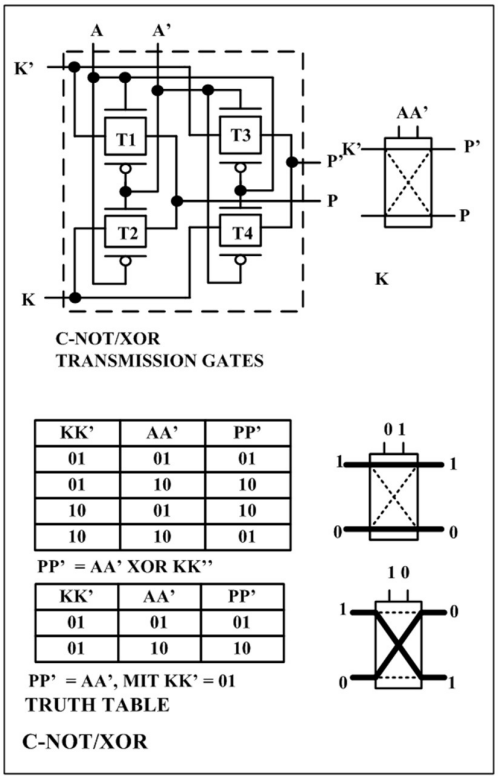
The essential technical statements of the above descriptions can be represented in drawings.

Figure 1 shows the scaled Boolean CNOT/XOR.

The dual-rail CNOT/XOR: **PP' = AA' XOR KK'** is constructed from **four transmission gates**. The input variables are **KK'** and **AA'**, and the output variable is **PP'**. When **AA' = 01**, the signals **KK'** are passed through to the output **PP'**. When **AA' = 10**, the signal paths are swapped.

With **KK' = 01 = constant**, the CNOT/XOR becomes a dual-rail switch **PP' = AA'**.

This functionality can be represented by a simple symbol. A complementary signal either connects or swaps two lines. This makes the CNOT/XOR a Boolean switch of dimension 2, dual-rail.



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

PP' = AA' XOR KK''

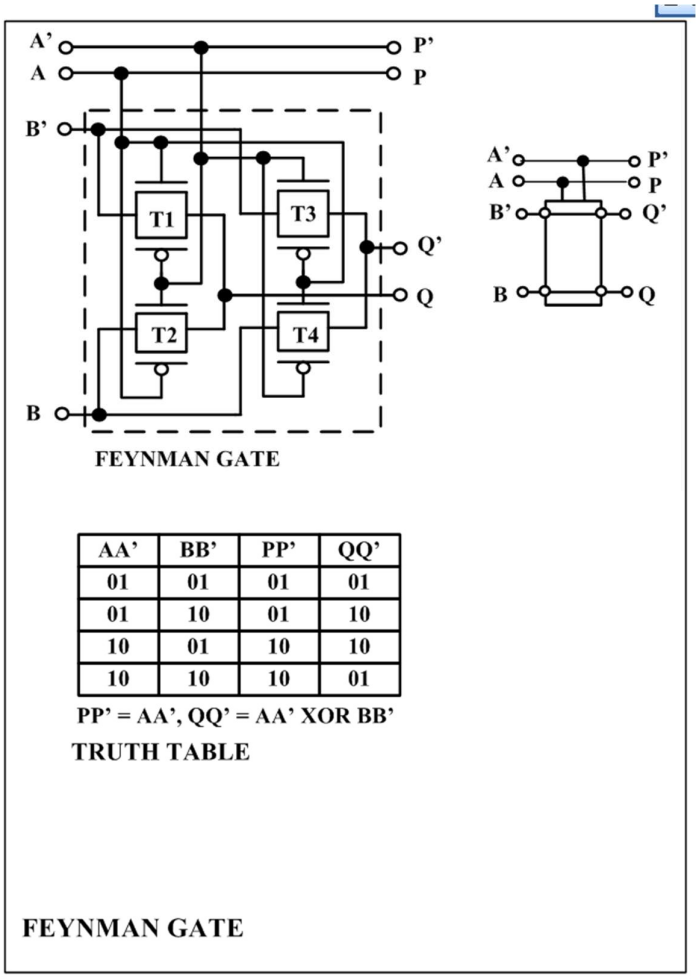
KK'	AA'	PP'
01	01	01
01	10	10

PP' = AA', MIT KK' = 01
TRUTH TABLE

C-NOT/XOR

The **dual-rail Feynman gate**: **PP' = AA', QQ' = AA' XOR BB'** is a variation of the dual-rail CNOT/XOR. The input variables are **AA'** and **BB'**, and the output variables are **PP'** and **QQ'**. The input variable **AA'** is forwarded directly to the output. The number of input and output variables is equal. In the representation, this corresponds to a function operating from left (L) (inputs) to right (R) (outputs). In this operating mode, the Feynman gate (L→R) is reversible.

The Feynman gate can also be operated from right to left (R→L): **AA' = PP', BB' = AA' XOR QQ'**. In this case, the input variables are **PP'** and **QQ'**, and the output variables are **AA'** and **BB'**. This makes the dual-rail Feynman gate **involutive**. Involutive logic represents a stricter subclass of reversible logic in which the mapping is identical to its own inverse, so that by applying the same operation again, the original state is exactly restored (the same operation forward and backward).



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

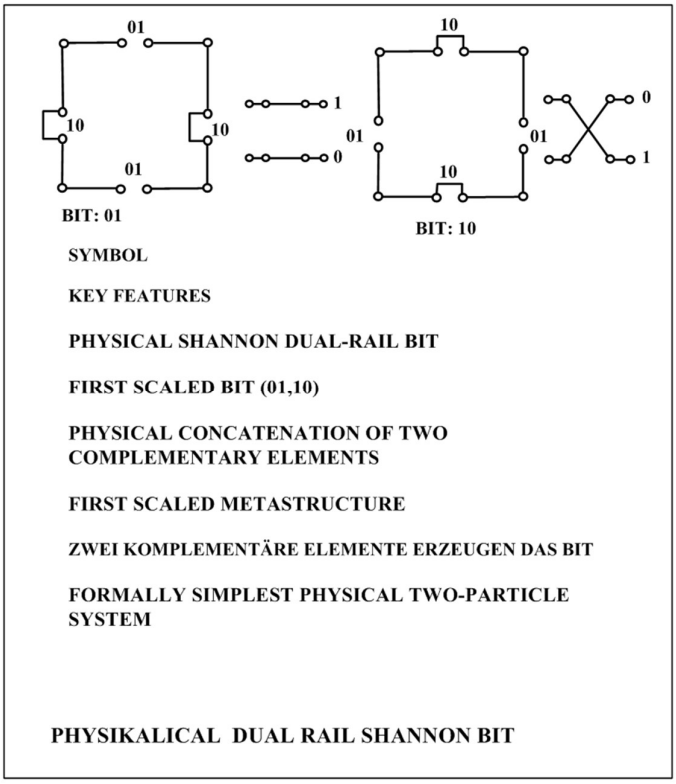
PP' = AA', QQ' = AA' XOR BB'

TRUTH TABLE

FEYNMAN GATE

The CNOT/XOR, when interpreted as a switch, can be transformed into a mechanical version, thereby defining the physical **dual-rail Shannon bit**. This is the first scaled bit **(01, 10)**; it is a concatenation (aggregation) of the single-bit complementary pairs **(0,1)** and **(1,0)**, forming a metastructure.

The dual-rail system thus constitutes the formally simplest, two-particle system capable of coordinated, algebraically controllable 2 state transitions.

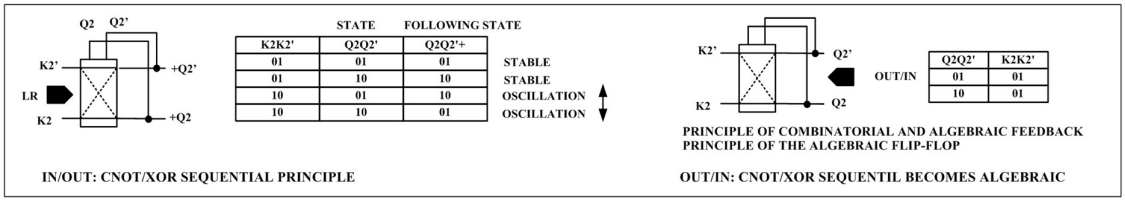


IN/OUT: CNOT/XOR Feedback / Sequential Operation

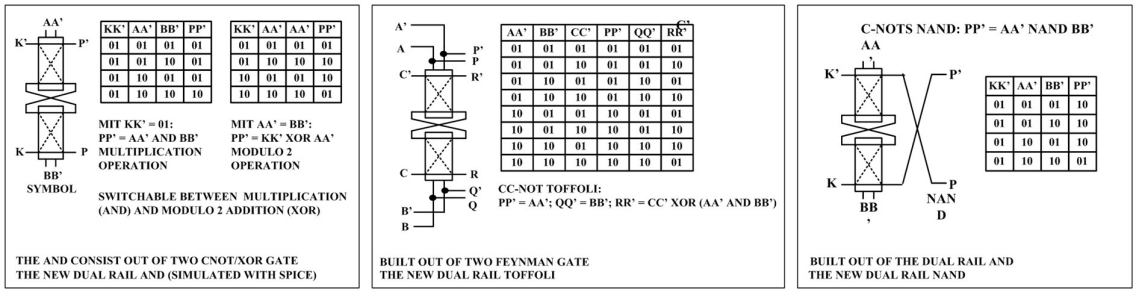
As a feedback function $\mathbf{Q_2Q_2'^{+} = Q_2Q_2'\ XOR\ K_2K_2'}$, the CNOT/XOR most clearly illustrates the algebraic form of feedback. For $\mathbf{K_2K_2' = 01}$, this function is stable; for $\mathbf{K_2K_2' = 10}$, it oscillates. It is therefore a **sequential structure**.

When operated in the reverse direction **OUT/IN**:
$\mathbf{K_2K_2' = Q_2Q_2'^{+}\ XOR\ Q_2Q_2'}$,

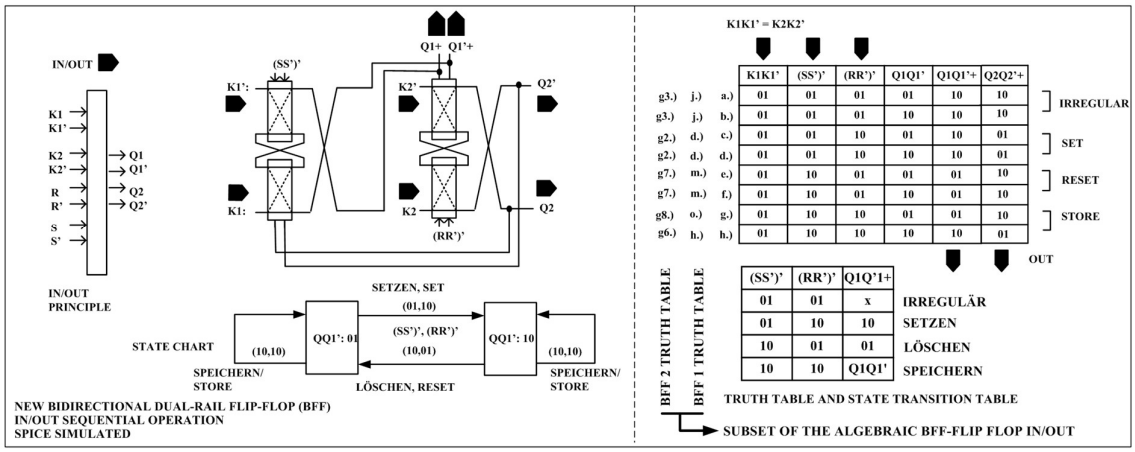
the result is $\mathbf{K_2K_2' = 01}$, which constitutes an **algebraic function**.



These principles form the basis of the new **bidirectional flip-flop (BFF)**, operating in **IN/OUT sequential mode**, which has also been simulated using SPICE (**Figure 2**).



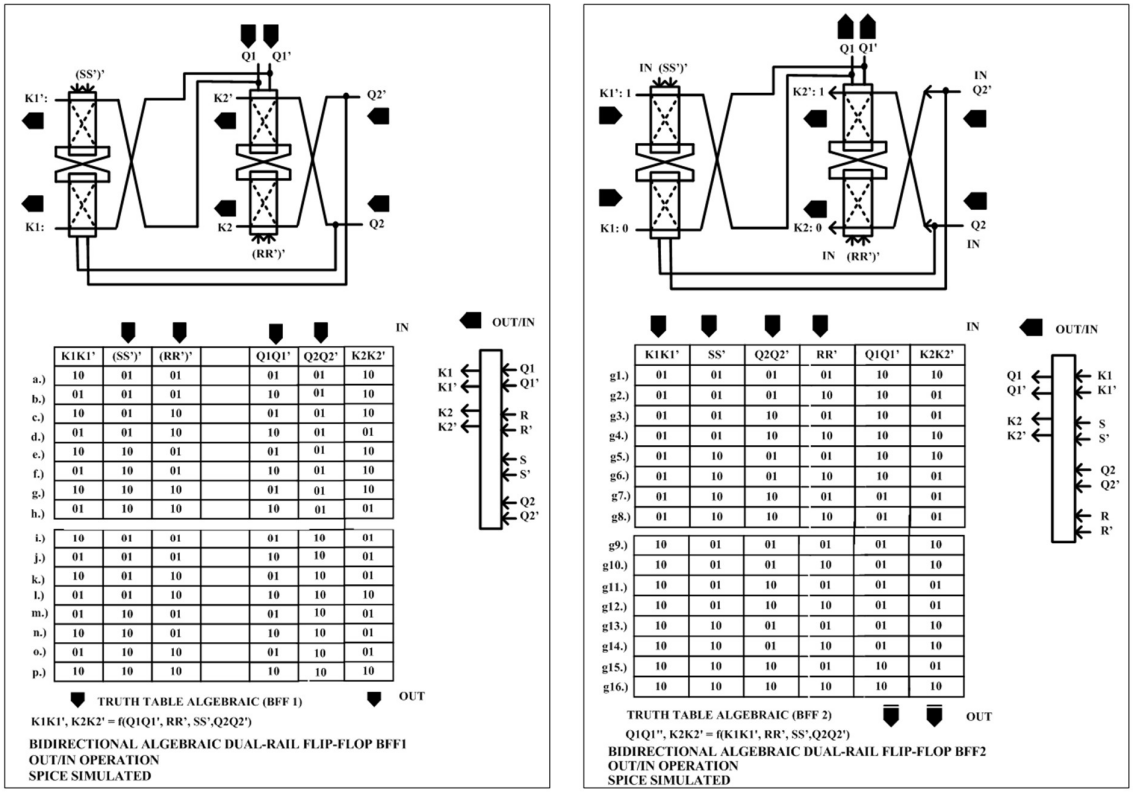
The flip-flop consists of two feedback-coupled new dual-rail NAND gates, which themselves are constructed from the new dual-rail AND gate. The reversible dual-rail Toffoli gate can be built from two Feynman gates. The SPICE simulation confirms the fundamental functionality of the flip-flop (set, reset, store), as represented in the state chart and the truth table.



			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

(SS')'	(RR')'	Q1Q'1+	
01	01	x	IRREGULÄR
01	10	10	SETZEN
10	01	01	LÖSCHEN
10	10	Q1Q1'	SPEICHERN

Rows **a.) to h.)** of the truth table can be mapped onto the truth tables of **BFF1** and **BFF2**, as demonstrated by a direct comparison of the respective rows. They form a subset of the algebraic flip-flops **BFF1** and **BFF2** in **OUT/IN operation**, as shown in **Figure 3**.



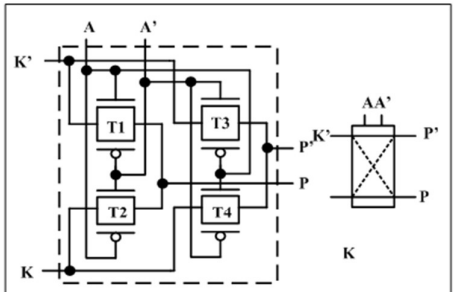
	K1K1'	(SS')'	(RR')'		Q1Q1'	Q2Q2'	K2K2'
a.)	10	01	01		01	01	10
b.)	01	01	01		10	01	10
c.)	10	01	10		01	01	10
d.)	01	01	10		10	01	01
e.)	10	10	01		01	01	10
f.)	01	10	01		10	01	10
g.)	10	10	10		01	01	10
h.)	01	10	10		10	01	01
i.)	10	01	01		01	10	01
j.)	01	01	01		10	10	01
k.)	10	01	10		01	10	01
l.)	01	01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

Switching between the operating modes of the **sequential flip-flop** and the **algebraic BFF2** is achieved by **swapping the input and output variables** (Figure 4). The rows in the truth tables corresponding to **set, reset, and hold (storage)** remain unchanged in both modes.

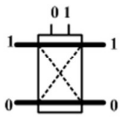
By swapping the input/output variables, the different **algebraic flip-flops (BFF1, BFF2, BFF3)** are generated. The **24 algebraic functions** with **2 input and 2 output variables** are obtained using **2 control variables**, as shown in **Figures 5, 6, and 7**.



C-NOT/XOR
TRANSMISSION GATES

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

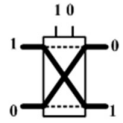
PP' = AA' XOR KK''



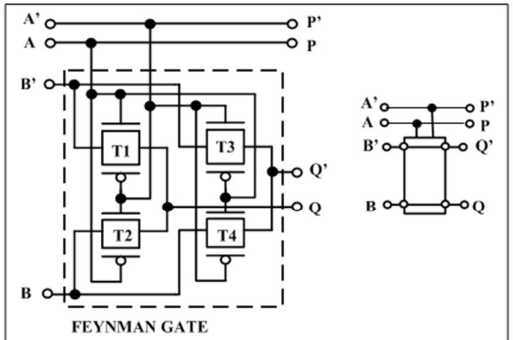
KK'	AA'	PP'
01	01	01
01	10	10

PP' = AA', MIT KK' = 01

TRUTH TABLE



C-NOT/XOR



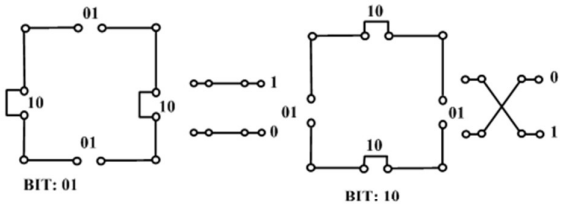
FEYNMAN GATE

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'

TRUTH TABLE

FEYNMAN GATE



SYMBOL

KEY FEATURES

PHYSICAL SHANNON DUAL-RAIL BIT

FIRST SCALED BIT (01,10)

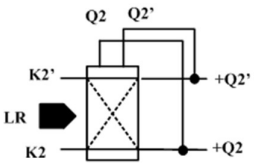
PHYSICAL CONCATENATION OF TWO COMPLEMENTARY ELEMENTS

FIRST SCALED METASTRUCTURE

ZWEI KOMPLEMENTÄRE ELEMENTE ERZEUGEN DAS BIT

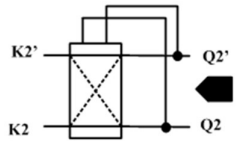
FORMALLY SIMPLEST PHYSICAL TWO-PARTICLE SYSTEM

PHYSIKALICAL DUAL RAIL SHANNON BIT



	STATE	FOLLOWING STATE	
K2K2'	Q2Q2'	Q2Q2'+	
01	01	01	STABLE
01	10	10	STABLE
10	01	10	OSCILLATION
10	10	01	OSCILLATION

IN/OUT: CNOT/XOR SEQUENTIAL PRINCIPLE



OUT/IN

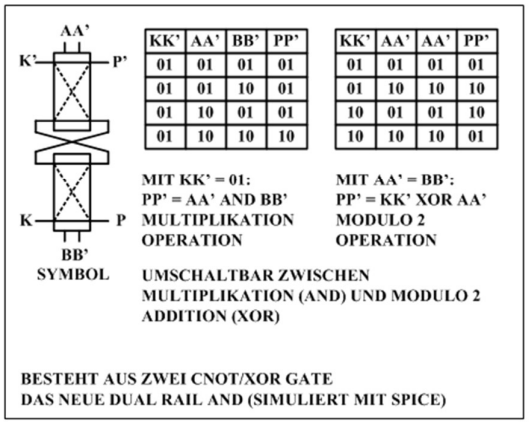
Q2Q2'	K2K2'
01	01
10	01

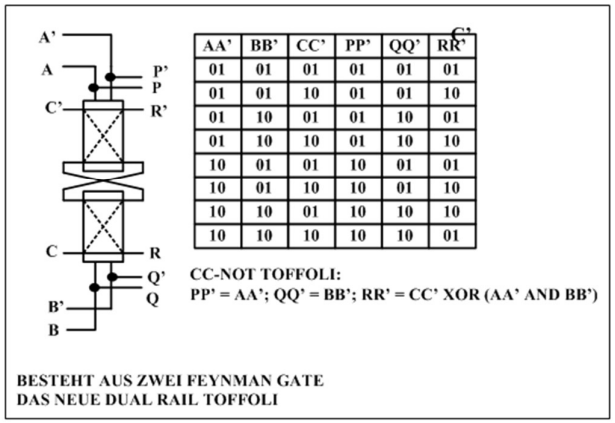
PRINCIPLE OF COMBINATORIAL AND ALGEBRAIC FEEDBACK
PRINCIPLE OF THE ALGEBRAIC FLIP-FLOP

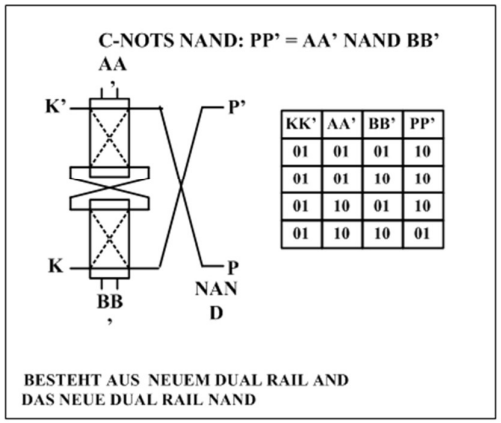
OUT/IN: CNOT/XOR SEQUENTIL BECOMES ALGEBRAIC

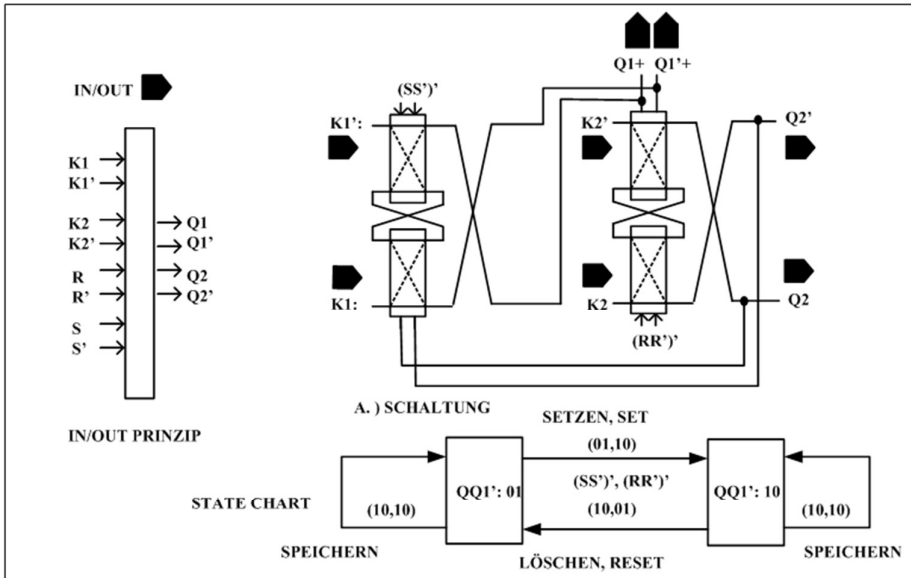
SKALED BOOLEAN CNOT/XOR, DATE: 07-12-2025

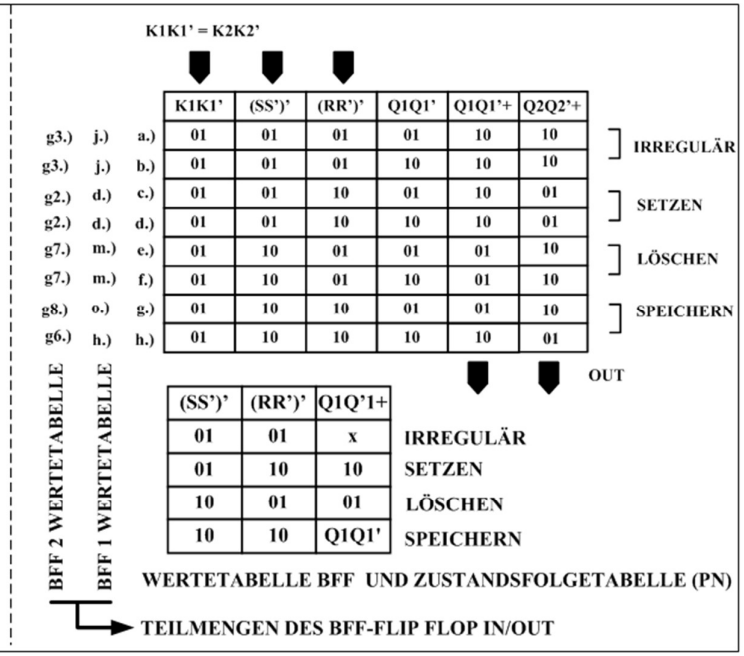
FIGUR 1


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)

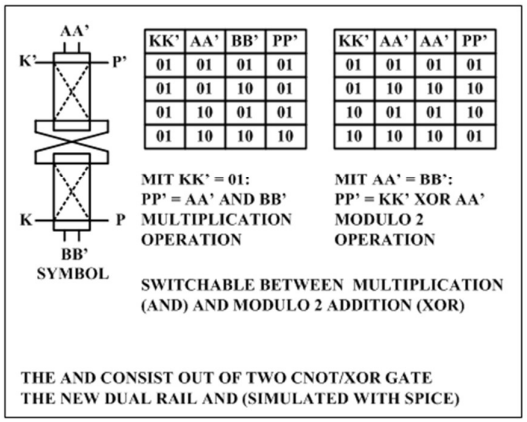

A'
A
C'
C
B'
B
P'
P
R'
R
Q'
Q
C'
AA' BB' CC' PP' QQ' RR'
01 01 01 01 01 01
01 01 10 01 01 10
01 10 01 01 10 01
01 10 10 01 10 10
10 01 01 10 01 01
10 01 10 10 01 10
10 10 01 10 10 10
10 10 10 10 10 01
CC-NOT TOFFOLI:
PP' = AA'; QQ' = BB'; RR' = CC' XOR (AA' AND BB')
BESTEHT AUS ZWEI FEYNMAN GATE
DAS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

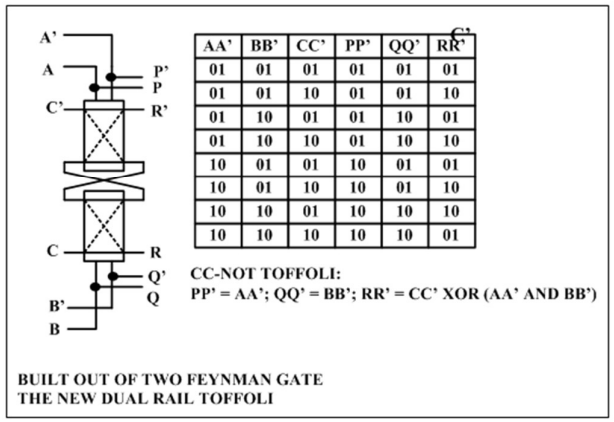

C-NOTS NAND: PP' = AA' NAND BB'
AA'
K'
P'
K
P
BB'
NAND
KK' AA' BB' PP'
01 01 01 10
01 01 10 10
01 10 01 10
01 10 10 01
BESTEHT AUS NEUEM DUAL RAIL AND
DAS NEUE DUAL RAIL NAND

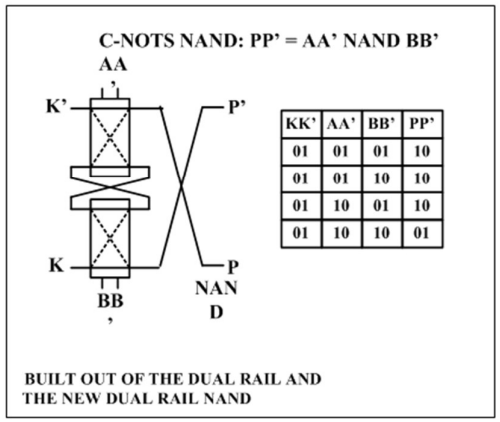

IN/OUT
K1
K1'
K2
K2'
R
R'
S
S'
Q1
Q1'
Q2
Q2'
IN/OUT PRINZIP
Q1+ Q1'+
(SS')'
K1':
K2'
Q2'
K1:
K2
Q2
(RR')'
A.) SCHALTUNG
SETZEN, SET
(01,10)
STATE CHART
(10,10)
QQ1': 01
(SS')', (RR')'
(10,01)
QQ1': 10
(10,10)
SPEICHERN
LÖSCHEN, RESET
SPEICHERN
DAS NEUE BIDIREKTIONALE FLIP FLOP (BFF),
IN/OUT SEQUENTIELL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

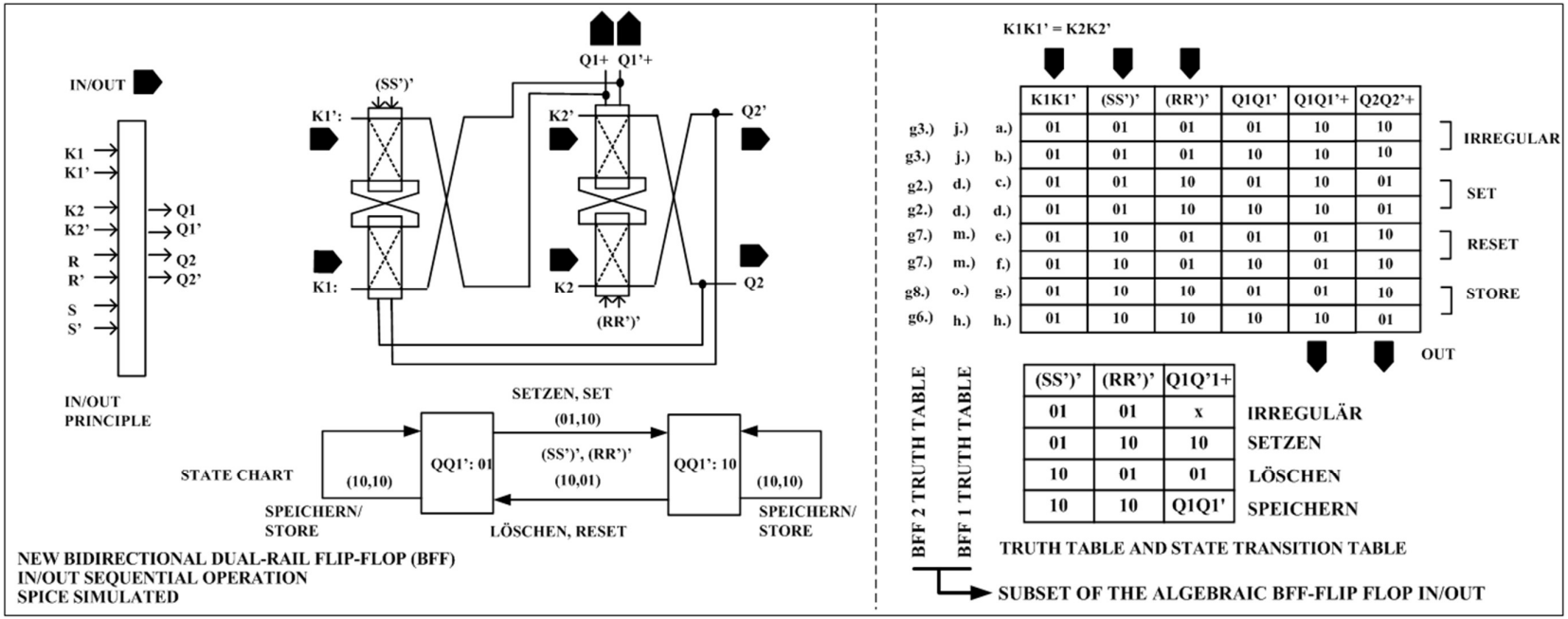

K1K1' = K2K2'
K1K1' (SS')' (RR')' Q1Q1' Q1Q1'+ Q2Q2'+
g3.) j.) a.) 01 01 01 01 10 10
g3.) j.) b.) 01 01 01 10 10 10
g2.) d.) c.) 01 01 10 01 10 01
g2.) d.) d.) 01 01 10 10 10 01
g7.) m.) e.) 01 10 01 01 01 10
g7.) m.) f.) 01 10 01 10 01 10
g8.) o.) g.) 01 10 10 01 01 10
g6.) h.) h.) 01 10 10 10 10 01
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

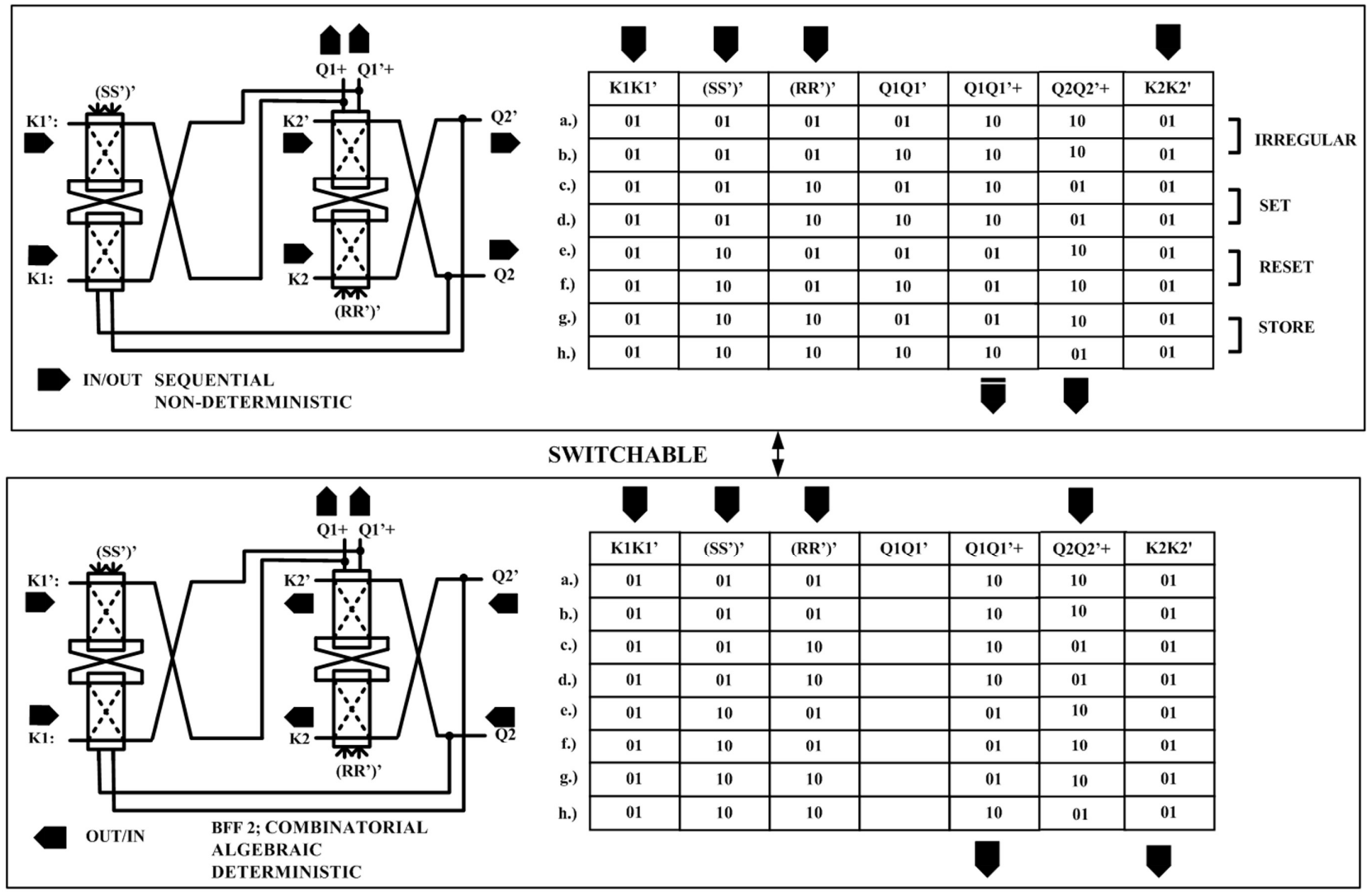

AA'
K'
P'
K
P
BB'
SYMBOL
KK' AA' BB' PP'
01 01 01 01
01 01 10 01
01 10 01 01
01 10 10 10
MIT KK' = 01:
PP' = AA' AND BB'
MULTIPLICATION
OPERATION
KK' AA' AA' PP'
01 01 01 01
01 10 10 10
10 01 01 10
10 10 10 01
MIT AA' = BB':
PP' = KK' XOR AA'
MODULO 2
OPERATION
SWITCHABLE BETWEEN MULTIPLICATION
(AND) AND MODULO 2 ADDITION (XOR)
THE AND CONSIST OUT OF TWO CNOT/XOR GATE
THE NEW DUAL RAIL AND (SIMULATED WITH SPICE)


A'
A
C'
C
B'
B
P'
P
R'
R
Q'
Q
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')
BUILT OUT OF TWO FEYNMAN GATE
THE NEW DUAL RAIL TOFFOLI


C-NOTS NAND: PP' = AA' NAND BB'
AA'
K'
P'
K
BB'
P
NAND
KK' AA' BB' PP'
01 01 01 10
01 01 10 10
01 10 01 10
01 10 10 01
BUILT OUT OF THE DUAL RAIL AND
THE NEW DUAL RAIL NAND


IN/OUT
K1
K1'
K2
K2'
R
R'
S
S'
Q1
Q1'
Q2
Q2'
IN/OUT
PRINCIPLE
(SS')'
Q1+ Q1'+
K1':
K1:
K2'
K2
Q2'
Q2
(RR')'
SETZEN, SET
(01,10)
STATE CHART
QQ1': 01
QQ1': 10
(10,10)
(SS')', (RR')'
(10,01)
(10,10)
SPEICHERN/
STORE
LÖSCHEN, RESET
SPEICHERN/
STORE
NEW BIDIRECTIONAL DUAL-RAIL FLIP-FLOP (BFF)
IN/OUT SEQUENTIAL OPERATION
SPICE SIMULATED
K1K1' = K2K2'
K1K1' (SS')' (RR')' Q1Q1' Q1Q1'+ Q2Q2'+
g3.) j.) a.) 01 01 01 01 10 10 IRREGULAR
g3.) j.) b.) 01 01 01 10 10 10
g2.) d.) c.) 01 01 10 01 10 01 SET
g2.) d.) d.) 01 01 10 10 10 01
g7.) m.) e.) 01 10 01 01 01 10 RESET
g7.) m.) f.) 01 10 01 10 01 10
g8.) o.) g.) 01 10 10 01 01 10 STORE
g6.) h.) h.) 01 10 10 10 10 01
OUT
BFF 2 TRUTH TABLE
BFF 1 TRUTH TABLE
(SS')' (RR')' Q1Q'1+
01 01 x IRREGULÄR
01 10 10 SETZEN
10 01 01 LÖSCHEN
10 10 Q1Q1' SPEICHERN
TRUTH TABLE AND STATE TRANSITION TABLE
SUBSET OF THE ALGEBRAIC BFF-FLIP FLOP IN/OUT

FIGUR 2



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

	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

TRUTH TABLE ALGEBRAIC (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 ↓ ↓

CONTROL:

NUMBER OF FUNCTIONS
4 INPUT VARIABLES 2 OUTPUT VARIABLES 2 FUNCTIONS
3 INPUT VARIABLES 2 OUTPUT VARIABLES 4 FUNCTIONS
2 INPUT VARIABLES 2 OUTPUT VARIABLES 8 FUNCTIONS
1 INPUT VARIABLE 2 OUTPUT VARIABLES 16 FUNCTIONS

FIGUR 5 CONTROL OF BFF1)

BFF1: IN/OUT VARIABLES

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

TRUTH TABLE ALGEBRAIC (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

CONTROL

NUMBER OF FUNCTIONS
4 INPUT VARIABLES 2 OUTPUT VARIABLES 2 FUNCTIONS
3 INPUT VARIABLES 2 OUTPUT VARIABLES 4 FUNCTIONS
2 INPUT VARIABLES 2 OUTPUT VARIABLES 8 FUNCTIONS
1 INPUT VARIABLE 2 OUTPUT VARIABLES 16 FUNCTIONS

BFF2: IN/OUT VARIABLES

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

Q1Q1' = f (RR', Q2Q2') K1K1', SS': CONTROL	K2K2' = f (RR', Q2Q2') 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

FIGUR 6 (CONTROL OF BFF2)

TRUTH TABLE ALGEBRAIC (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

CONTROL

NUMBER OF FUNCTIONS
4 INPUT VARIABLES 2 OUTPUT VARIABLES 2 FUNCTIONS
3 INPUT VARIABLES 2 OUTPUT VARIABLES 4 FUNCTIONS
2 INPUT VARIABLES 2 OUTPUT VARIABLES 8 FUNCTIONS
1 INPUT VARIABLE 2 OUTPUT VARIABLES 16 FUNCTIONS

BFF3: IN/OUT VARIABLE

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

FIGUR 7 (CONTROL BFF3)