Pin-Type Based VLSI Partitioning

Gürkan Uygur, Sebastian M. Sattler, Chair of Reliable Circuits and Systems, LZS, Friedrich-Alexander-University Erlangen-Nuremberg, Paul-Gordan-Str. 5, 91052 Erlangen, Germany, Email: {uygur,sattler}@lzs.eei.uni-erlangen.de

Abstract

For deeply modeling and verification of feedback asynchronous and synchronous circuitry it is essential to partition the circuitry into sub-structures. For this, many graph-theoretical approaches for VLSI partitioning are investigated. In this paper, we present a novel partitioning approach based on pin-types. We state the underlying data structure to classify a pin as input, output, state, feed, cycle and iter, respectively. The employed data structure provides the matrix representation in different normal forms. We show the unique partitioning of a given circuit structure considered as a directed graph (network) into its regular and singular parts. Additionally, we upgrade the low level partitioning technique to a high level abstraction and state our formalism for functionally stable and unique parallel composition and decomposition of network under balance (NUB).

Index Terms—network, feedback, balance, partition, parallel, composition, decomposition, functional, stable

1 Introduction

AUTOMOTIVE is one of the most conservative technology with the highest safety demand. And at the same time, the automotive and automobile industry is confronted with the strongly increasing demand for highly complex, distributed, diverse and heterogeneous functionality and connectivity, not only for safety and real time applications but also for multimedia infotainment [1]. For the industry to control such complex automotive systems it becomes a dominating factor to distinguish oneself in international business rivalry. To manage and control the complexity there exit many VLSI partitioning techniques – the principle of divide-and-conqueror – based on graph respectively network theory whereby a node can be a pin in a low level abstraction respectively an arbitrary entity in a high level abstraction, hence partitioning plays a key role in VLSI circuits and systems [2].

Non-directed graph vs. directed graph (digraph): Let $G_i = (N_i, E_i)$ specify a graph with index i, set of nodes N_i and set of edges $E_i \subseteq N_i \times N_i$. In the digital technology, there exist well known graph-theoretical algorithms to divide G_i into equivalence and compatibility classes, respectively [3]. For this classification, the direction of an edge is not necessarily considered. To strictly differ between a non-directed and a directed graph, we say that a non-directed graph $G_i = (N_i, E_i)$ is symmetric, $\forall(a,b) \in (N_i \times N_i)((a,b) \in E_i \Rightarrow (b,a) \in E_i)$, but a directed graph (digraph) $DG_i = (N_i, E_i)$ is not necessarily symmetric. The (serial respectively parallel) composition of two non-directed graphs G_i and G_j is given by connecting (identifying) the so called (non-directed) terminal nodes $n_i \in N_i$ and $n_j \in N_j$: $n_i = n_j$. In case of a directed graph DG_i, a terminal node n_i is more precisely classifiable as an input (source) respectively output (drain) in accordance with the given data flow in DG_i, hence in this paper, we will prefer and use digraphs to model a network of circuits and systems.

Singular vs. regular: As mentioned above, in a digraph DG_i, a node n_i can be classified as an input and output, respectively. If the node n_i is the first node in a path then one can uniquely classify n_i as an input, and if n_i is the last node in a path then one can uniquely classify n_i as an output. However, the classification of n_i becomes non-trivial, e. g., if n_i is in a cycle; in this case, n_i can be a state respectively a feed depending on further criteria. Additionally, a cycle without feed is not controllable, and a cycle without state is not observable. Thus, partitioning a network into its regular and singular parts helps to check consistency and testability in the earlier stage of development.

Testing by pin-type based partitioning: The circuits and systems have to be tested to warrant functional safety and reliable reproducibility of safety critical system components as well as functional stability of multimedia and infotainment components. The tests are executed by specific test systems [4]. One should notice, that a test is done by pin-type based partitioning, namely: to define a functional test, the set of channels (pins) Σ has to be partitioned into pairwise disjoint (sub-)sets of input high σ_1, input low σ_0, output high σ_{H}, output low σ_{L}, output high-resistance σ_{Z} and output don't care σ_{X}. Then, $\sigma = (\sigma_1, \sigma_0, \sigma_{\mathrm{H}}, \sigma_{\mathrm{L}}, \sigma_{\mathrm{Z}}, \sigma_{\mathrm{X}})$ defines a test, and $(\sigma, \Sigma\backslash\sigma) \in 7^{\Sigma}$ partitions Σ into the 6 disjoint (sub-)sets of channels in σ and the (sub-)set of non-specified channels in $\Sigma\backslash\sigma$. Thus, a test is defined by a partial 7-valued logic.

Synchronous vs. asynchronous circuits and systems: Even though the class of synchronous circuit systems is well understood and forms the base of almost all modern microcomputers, there is a renewing and increasing interest in asynchronous design. Asynchronous circuits are designed without the assumption of a (global) master-clock. They can be viewed as a balanced network of digital components composed such that the components are operating in parallel and are synchronized on voltage transitions of the interconnecting wires [5]. Emphasis is shifting from asynchronous-in-the-small to asynchronous very large scale integration (VLSI) circuits and systems. Asynchronous VLSI is progressing from a fashionable academic research topic to a viable solution to a number of digital VLSI design challenges [6]. The inherent reason is

that asynchronous circuits are a promising type of digital circuits [7] with lots of potential benefits of improved system performance, modular design, low power consumption and reduced electro-magnetic emission [8]. Further essential characteristics of asynchronous circuits are, that asynchronous circuits show no problems with clock skew and related subtle issues, and are fundamentally more tolerant of voltage, temperature and manufacturing process variation [9],[10],[11]. It is remarkable, that the international technology roadmap for semiconductors (report on design [12]) predicts that up to 40% of the designs will be driven by 'handshake clocking' (i.e. asynchronous) in 2020 [9].

As the synchronous world is master-clocked, each feedback within a synchronous system can be (after several worst case assumptions) considered as being cut [13]. This allows the synchronous world to be formally considered as a totally specified (edge or state) triggered system. This abstraction then immensely contributes to profit from existing powerful theoretical foundations based on totally defined algebra, e.g., the (totally defined) complementary distributive lattice (Boolean algebra), Boolean ring with regular (totally defined invertible) plus ($\oplus$) and singular (distributive) multiplication ($\cdot$), and the totally defined automata theory provided with digital encodings, respectively. Hence, industrial standardization and development of powerful CAD tools for modeling, verification, synthesis and test are provided inherently for synchronous circuits and systems. However, this totally defined paradisiacal situation is not given in the real life within the asynchronous world. In an asynchronous circuit there is no common and global door mechanism to synchronize the data flow in the hole system. The asynchronous world assumes binary signals, too, but there is no common and discrete time. Any change of signals may cause a concurring respectively competing transition of the system into the next state. To avoid rivalry, the circuits use handshake protocols and diverse arbiters between their components in order to perform the necessary synchronization, communication and sequencing of operations [13]. Hence, the asynchronous circuits and systems intrinsically can be considered as a balanced analog network of digital components. In terms of digitally asynchronous automata theory [14], "balanced" can be interpreted as "functionally stable" [3], so to say the old state information is safely equal to the new feedback state information, the asynchronous circuitry is stably staying in a reflexive state and the corresponding analog network is balanced, respectively. In all remaining cases, the network is unbalanced and unstable, and hence not defined. Thus, for asynchronous circuits and systems, a suitable theoretical framework is needed for formally working with partiality and dealing with concurrency issues in general discrete-state-systems [5] to warrant correctness by construction, which is our aim to contribute to in this work.

Organization of the paper: In Section 2 we start with our pin-type based partitioning of a network (digraph) in low level abstraction. In Section 3 we upgrade the low level partitioning technique to a high level abstraction and state our formalism for functionally stable and unique parallel composition and decomposition of network under balance, and finally the paper closes with a conclusion in Section 4.

2 Low Level Abstraction

Let $DG_i = (N_i, E_i)$ specify a network (digraph) with index i, set of nodes N_i and set of edges $E_i \subseteq N_i \times N_i$. A very simple data structure which represents DG_i is a list (matrix) L_i of lists (words) $w_i \in L_i$ of nodes (pins) $n_i \in w_i$, $L_i = (w_1, w_2, \cdots)$. For each edge $(a, b) \in E_i$, a new word $w_i = a \cdot b$ is appended into the matrix L_i, $\forall (a,b) \in (N_i \times N_i)((a,b) \in E_i \Rightarrow a \cdot b \in L_i)$. All of the remaining (non-related single) nodes n_i are appended as a word of a single node n_i into the matrix L_i, $\forall n_i \in N_i (n_i \in (N_i \backslash \bigcup_{(a,b)\in E_i} \{a, b\}) \Rightarrow n_i \in L_i)$. This construction of L_i represents not only all edges ($E_i \subseteq N_i \times N_i$) of length 2 but also contains all information about DG_i, so we can consider L_i and DG_i as equivalent, $L_i = DG_i$. Notice that the order of words in the matrix doesn't matter.

In the following, we define a more general matrix representation of DG_i: consider an arbitrary matrix L_i with words $w_i \in L_i$ of a (non-negative) length, $|w_i| \geq 0$. An empty word (neutral element) ($w_i = \epsilon$, $|w_i| = 0$) doesn't contain any information about DG_i, so it can be removed from the matrix L_i. A word of length 1 ($w_i = n_i$, $|w_i| = 1$) is trivial and specifies just a node $n_i \in N_i$. Further, a word of length 2 ($w_i = a \cdot b$, $|w_i| = 2$) is trivial and specifies just an edge $(a, b) \in E_i$. A word $w_i = n_1 \cdots n_j \cdot n_{j+1} \cdots$ of length $|w_i| > 2$ specifies the edges $(n_j, n_{j+1}) \in E_i$ with $1 \leq j < |w_i|$.

In the following, we prepare the data structures for the partitioning of DG_i: we build the set of (local) source nodes S_S and the set of (local) drain nodes S_D of DG_i: $S_\mathrm{S} = \{a | (a, b) \in E_i\}$, $S_\mathrm{D} = \{b | (a, b) \in E_i\}$. Then, the set of global input nodes S_I consists of all source nodes which does not appear as a (local) drain node, $S_\mathrm{I} = S_\mathrm{S} \backslash S_\mathrm{D}$, and vice versa, the set of global output nodes S_O consists of all drain nodes which does not appear as a (local) source node, $S_\mathrm{I} = S_\mathrm{D} \backslash S_\mathrm{O}$. We use the circuit-finding algorithm given in [15] to extract all simple cycles in DG_i; the runtime complexity of the algorithm is stated as $O((|N_i| + |E_i|)(c + 1))$ and the state space complexity is stated as $O(|N_i| + |E_i|)$ where c is the number of elementary circuits in DG_i. Each extracted cycle is captured as a word $w_i = n_1 \cdot n_2 \cdots n_j \cdot n_1$ with the first and the last nodes are equal (closed path) where the length of the cycle is $|w_i| - 1$. Then, storing all of the cycles in a list constructs the matrix of the regular part $L_{\mathrm{R},i}$ of DG_i. As next, we generate the matrix of edges $L_{\bar{\mathrm{R}},i}$ whereby each edge occurring in any cycle is excluded: for this, we convert the matrix of the regular part $L_{\mathrm{R},i}$ into its edge normal form and subtract $L_{\mathrm{R},i}$ from the edge matrix L_i of DG_i, $L_{\bar{\mathrm{R}},i} = L_i \backslash L_{\mathrm{R},i} = \{a \cdot b \in L_i | a \cdot b \notin L_{\mathrm{R},i}\}$. Then, $L_{\bar{\mathrm{R}},i}$ stores the singular part of DG_i. Obviously, the union of the regular and singular parts of DG_i is a regular operation: $DG_i = L_i = L_{\mathrm{R},i} \cup L_{\bar{\mathrm{R}},i}$, $L_i \backslash L_{\mathrm{R},i} = L_{\bar{\mathrm{R}},i}$, $L_i \backslash L_{\bar{\mathrm{R}},i} = L_{\mathrm{R},i}$.

ISBN 978-3-8007-3580-8

To extract all state nodes in DG_i, in each cycle $w_i \in L_{\mathrm{R},i}$ we search for a node $n_i \in w_i$ where an edge $n_i \cdot n_{i+1} \in L_i$ exists such that the successor node n_{i+1} is not in this cycle, $n_{i+1} \notin w_i$. Then, the set of all state nodes S_Z is given as $S_\mathrm{Z} = \{n_i \in w_i \in L_{\mathrm{R},i} | n_i \cdot n_{i+1} \in L_i \wedge n_{i+1} \notin w_i\}$. Analogically, to extract all the feed nodes in DG_i, in each cycle $w_i \in L_{\mathrm{R},i}$ we search for a node $n_i \in w_i$ which is not a state node, $n_i \notin S_\mathrm{Z}$, where an edge $n_{i-1} \cdot n_i \in L_i$ exists such that the predecessor node n_{i-1} is not in this cycle, $n_{i-1} \notin w_i$. Then, the set of all feed nodes S_F is given as $S_\mathrm{F} = \{n_i \in w_i \in L_{\mathrm{R},i} | n_i \notin S_\mathrm{Z} \wedge n_{i-1} \cdot n_i \in L_i \wedge n_{i-1} \notin w_i\}$. Further, we classify a node n_i as a cycle iff n_i is a node in a cycle $n_i \in w_i \in L_{\mathrm{R},i}$ and neither a state nor a feed at the same time. Then the set of all cycle nodes S_C is given as $S_\mathrm{C} = \{n_i \in w_i \in L_{\mathrm{R},i} | n_i \notin S_\mathrm{Z} \wedge n_i \notin S_\mathrm{F}\}$. Finally, we classify a node n_i as a iter iff n_i is neither a state nor a feed nor a cycle nor an input and nor an output. Then the set of all iter nodes S_It is given as $S_\mathrm{It} = N_i \backslash (S_\mathrm{Z} \cup S_\mathrm{F} \cup S_\mathrm{C} \cup S_\mathrm{I} \cup S_\mathrm{O})$.

3 High Level Abstraction

We define a trace w as a word $w \in M$ of the algebraic structure $M = (b^\Sigma, \cdot, \emptyset)$ with the carrier set b^Σ ($b \geq 2$), the free associative monoid multiplication (serial composition) $\cdot$ and the neutral element $\emptyset$. b^Σ is the structured set of all ordered partitions of b blocks σ_k (disjoint subsets) over Σ, a partition $\sigma \in b^\Sigma$ is then termed as $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_k, \cdots, \sigma_b)$. It is reasonable to arrange, that the block σ_b is the remaining block after subtracting all other blocks from Σ, $\sigma_b = \Sigma \backslash \bigcup_{k=1}^{b-1}(\sigma_k)$. So, in the special case of $b = 2$ it holds $\sigma = (\sigma_1, \sigma_2) = (\sigma_1, \Sigma \backslash \sigma_1)$. Obviously, in this case, one can ignore the redundant information $\Sigma \backslash \sigma_1$ and just consider $\sigma \in 2^\Sigma$ as a subset of Σ, $\sigma \subseteq \Sigma$.

The advantage of our definition is, that we can very easily explicitly and exklusively specify traces with simultaneous events. As an example, let $\Sigma = \{A, B, C\}$ be the set of single events and $\sigma \in 2^\Sigma$ an endofunction (operation) [16], specified as the set of simultaneously occurring events, acting on the set of states Z. Then, for the states $z, z' \in Z$ and $\sigma = \{A, B\}$, $z\sigma = z'$ is the transition from z to z' after $\{A, B\}$ acts on z. Notice that $\{A, B\} \neq \{A\} \cdot \{B\} \neq \{B\} \cdot \{A\}$, thus, the specified (sequential) transitions $z\{A, B\}$, $z\{A\} \cdot \{B\}$ and $z\{B\} \cdot \{A\}$ do not necessarily lead to the same state. In other words, we consider each $\sigma \in b^\Sigma$ as an hyper event (set of partitioned events, endofunction, operation, input), which (possibly partially) acts simultaneously on the set of states Z of an automaton A. Thus, a $\sigma \in b^\Sigma$ itself induces the corresponding trace on Z which is in general composed of the singular (non-feedback) parts and regular parts (cycles). We use the symbol $*$ to mark a "stack-at" state, where no transition is defined, meaning that any trace becomes deadlocked, see Fig. 1.

Additionally, we profit from the automata theory equipped with multi-set [17] to "control the idempotency property" of an arbitrary set. This is especially interesting for projecting a non-deterministic trace into qualified bases to encode deterministic parallel traces, see Fig. 2.

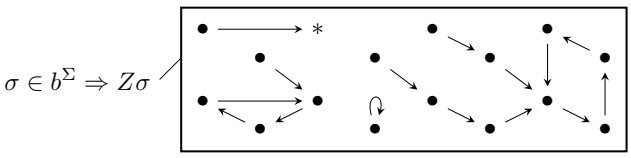
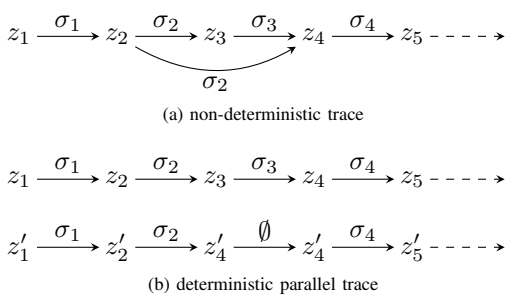


Fig. 1: A hyper event $\sigma \in b^\Sigma$ (possibly partially) acts on the set of states Z of an automaton A and induces the corresponding trace. In this example, the trace is composed of four singular (non-feedback) parts and three regular parts (cycles of the length 4, 3 and 1). σ is not defined on the state $*$, so the trace becomes deadlocked on this state.



(a) non-deterministic trace

(b) deterministic parallel trace

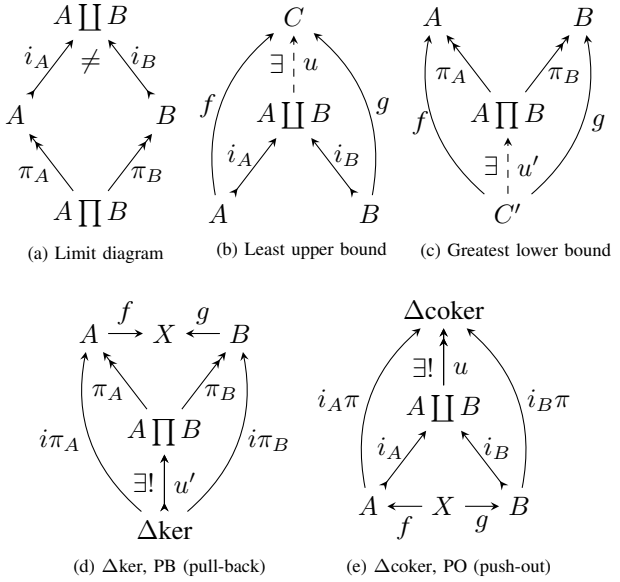
Fig. 2: Illustration of projecting a non-deterministic trace (Fig. 2a) into deterministic parallel traces (Fig. 2b). Notice that $\emptyset$ is the neutral element of the associative monoid multiplication $\cdot$ acting reflexively on the set of states (doing nothing) per definition.

Homo-, Mono- and Epimorphism: [18] shows how the analog circuit-level models, can be used as the basis of formal verification and describes a semi-algorithmic method to extract finite state models from an analog circuit-level model by means of homomorphic (behavior preserving) transformations. In view of electrical and communications engineering a homomorphism can be considered as a data flow channel, such that the channel is capable to preserve the operations on the source and consistently transmit them to the sink. As an example, let A_1 and A_2 are compatible regular matrices modeling some system entities. Then, the determinant $det(A_1 \cdot A_2) = det(A_1) \cdot det(A_2)$ is a homomorphism, which preserves the associative and non-commutative matrix multiplication. In this paper, we also profit form homomorphism (represented with the arrow symbol $\rightarrow$ between two objects A and B, $A \rightarrow B$), and particularly use monomorphism (injective respectively left total and one-to-one homomorphism, represented with the arrow symbol $\rightarrowtail$ between two objects) and epimorphism (surjective respectively right total homomorphism, represented with the arrow symbol $\twoheadrightarrow$ between two objects) [16]. As an example for the co-operation of monomorphism and epimorphism, the direct sum $A \oplus B$ of two bases A and B is equipped with two epimorphisms (projections) $\pi_A : A \oplus B \twoheadrightarrow A$ and $\pi_B : A \oplus B \twoheadrightarrow B$, and two monomorphisms $i_A : A \rightarrowtail A \oplus B$ and $i_B : B \rightarrowtail A \oplus B$. This co-operation is intuitively given in the Euclidean space $\mathbb{E}^2 = A \oplus B = \mathbb{R} \oplus \mathbb{R}$ with $(x, y) \in \mathbb{E}^2$, $(x, y) \overset{\pi_A}{\longmapsto} x$, $(x, y) \overset{\pi_B}{\longmapsto} y$, $x \overset{i_A}{\longmapsto} (x, 0)$ and $y \overset{i_B}{\longmapsto} (0, y)$.

ISBN 978-3-8007-3580-8

3.1 Limits and universal constructions

The limit diagram is a universal theoretical framework in the category theory [16]. It is the foundation in the algebraic world to bring two objects A and B (algebraic structures, entities, subsystems), which are completely not related to each other (there is no morphism between A and B), into the maximal relation by building the product ($\prod$) of A and B, $A \prod B$, and dual to $\prod$, into the minimal relation by building the co-product ($\coprod$) of A and B, $A \coprod B$. Hence, the corresponding diagram is a non-commutative one, $\pi_A \circ i_A \neq \pi_B \circ i_B$ see Fig. 3a[1]. This universal construct (Fig. 3) has two essential universal properties, called as least upper bound (LUB) and greatest lower bound (GLB): $A \coprod B$ is called LUB, since for each such diagram ($\forall$) there exists at least one ($\exists$) homomorphism u ($\dashrightarrow$) that prolongs from $A \coprod B$ into a common entity C, see Fig. 3b. Further, $A \prod B$ is called GLB, since for each such diagram ($\forall$) there exists at least one ($\exists$) homomorphism u' ($\dashrightarrow$) that shortens from C' to a common entity $A \prod B$, see Fig. 3c.



(a) Limit diagram (b) Least upper bound (c) Greatest lower bound

(d) Δker, PB (pull-back) (e) Δcoker, PO (push-out)

Fig. 3: Limits and universal constructions in category theory

Consider the entity X, then, Δker is defined as $\Delta\text{ker} = \{(a,b) \in A \prod B \,|\, af = bg\}$. One can inherently consider Δker as a pull-back (PB) [16]. Δker becomes integrated into the limit diagram using the monomorphism $\Delta\text{ker} \rightarrowtail A \prod B$, see Fig. 3d. And dual to Δker, Δcoker is defined as $\Delta\text{coker} = (A \coprod B)/_{\langle xf, xg\rangle_\sim}$ with $x \in X$. One can inherently consider Δcoker as a push-out (PO). Δcoker becomes integrated into the limit diagram using the epimorphism $A \coprod B \twoheadrightarrow \Delta\text{coker}$, see Fig. 3e.

In this paper, we use the limit construction to bring traces of potentially non-related system entities (circuit structures, gates, automata) into universal relation. This then naturally leads to the parallel composition of entities preserving structural and behavioral properties and warranting conflict-free and maximum possible flow of information specified and controlled by any collection of free bases. In Section 3.2, we will use this technique to construct our associative and invertible Automata Based Composition (ABC) to model the balanced network. We have profit from the work in [19], but in this paper we have significantly upgraded the formalism using more generalized trace monoids, automata definitions, information encoding and decoding and developed the theoretical framework using limits and universal constructions in the category theory.

1. Notice: we read the (serial) composition of morphisms from left to right, i. e., $\pi_A \circ i_A$ says first π_A then i_A.

3.2 Theoretical Framework

At the beginning of Section 3 we defined the trace monoid $M_i = (b^{\Sigma_i}, \cdot, \emptyset)$ for an entity (subsystem) A_i. As we are interested in associative and invertible parallel composition, we will take a co-algebraic approach to define the monomorphism ρ (encoder) and the dual epimorphism ρ^{-1} (decoder) to construct our codec for a trace $(w = \sigma \cdot \sigma' \cdots) \in M$ of hyper events $\sigma, \sigma' \in b^\Sigma$:

Definition 1 (σ-Codec). $\rho : M_{||} \rightarrowtail M_i \prod M_j$ with $M_{||} = (b^{\Sigma_{||}}, \cdot, \emptyset)$ and $\Sigma_{||} = \Sigma_i \cup \Sigma_j$. For generators σ of $M_{||}$ we declare $\rho : \sigma \mapsto (\sigma_i, \sigma_j)$ with $\sigma_i = \sigma \cap \Sigma_i$ and $\sigma_j = \sigma \cap \Sigma_j$. Further, it holds $\rho('\sigma \cdot ''\sigma) = \rho('\sigma) \cdot \rho(''\sigma) = {}'(\sigma_i, \sigma_j) \cdot ''(\sigma_i, \sigma_j) = ({}'\sigma_i \cdot ''\sigma_i \,,\, {}'\sigma_j \cdot ''\sigma_j)$. ρ denotes the σ-Encoder. Dual to ρ, ρ^{-1} is defined as $\rho^{-1} : M_i \coprod M_j \twoheadrightarrow M_{||}$ and denotes the corresponding σ-Decoder. Furthermore, "$\rho(\sigma)$ exists" if and only if $(\sigma_i \cap \Sigma_j = \sigma_j \cap \Sigma_i)$.

In the following, we will integrate ρ and ρ^{-1} into the limit diagram (Fig. 3): Let us consider two trace monoids M_i and M_j as two bases. Then, the product is $M_i \prod M_j$, the coproduct is $M_i \coprod M_j$, and thus, $M_{||}$ becomes the Δker and Δcoker at the same time, see Fig. 4. The required monomorphic channel encoding ρ is given by two epimorphisms (projections) $\rho_i = \rho\pi_i$ with $\rho_i : M_{||} \twoheadrightarrow M_i$ and $\rho_j = \rho\pi_j$ with $\rho_j : M_{||} \twoheadrightarrow M_j$, and the required epimorphic channel decoding ρ^{-1} is given by two monomorphisms (injections) $\rho_i^{-1} = i_i\rho^{-1}$ with $\rho_i^{-1} : M_i \rightarrowtail M_{||}$ and $\rho_j^{-1} = i_j\rho^{-1}$ with $\rho_j^{-1} : M_j \rightarrowtail M_{||}$, see Fig. 4a. Fig. 4b shows the simultaneously parallel and associative composition of two traces w_i and w_j closed under inversion. Fig. 4c shows the hybrid view considering traces in detail. In Fig. 4d, herewith, we introduce our definition of $+$ which enables the linear operation view of the composition procedure.

In the following we state the axioms for ρ^{-1} (we are, of course, only interested in arguments at which $\rho(\sigma)$ exists). The following formulae are to be understood as universally quantified:

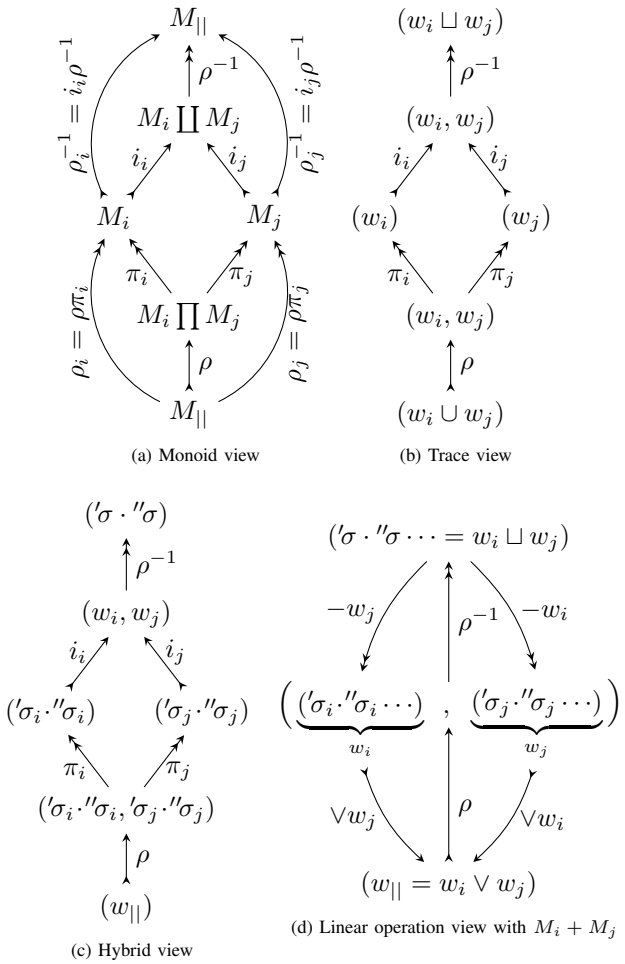
Axiom 1 (Invertibility). $\rho(\rho^{-1}(\sigma_i, \sigma_j)) = (\sigma_i, \sigma_j)$

Axiom 2 (Associativity). $\rho^{-1}(\rho^{-1}(\sigma_i, \sigma_j), \sigma_k) = \rho^{-1}(\sigma_i, \rho^{-1}(\sigma_j, \sigma_k)) =: \rho^{-1}(\sigma_i, \sigma_j, \sigma_k)$ *(herewith introducing an abbreviated notation)*

Axiom 3 (Commutativity). $\rho^{-1}(\sigma_i, \sigma_j) = \rho^{-1}(\sigma_j, \sigma_i)$

Axiom 4 (Neutral El.). $\rho^{-1}(\emptyset, \sigma_i) = \rho^{-1}(\sigma_i, \emptyset) = \sigma_i$

ISBN 978-3-8007-3580-8



(a) Monoid view (b) Trace view

(c) Hybrid view (d) Linear operation view with $M_i + M_j$

Fig. 4: Limit diagram of the channel encoding and decoding, which defines the simultaneously parallel associative and invertible composition of two traces w_i and w_j. Trace w_i is given as $(w_i = '\sigma_i \cdot ''\sigma_i \cdots) \in M_i$ of hyper events $\sigma_i \in b^{\Sigma_i}$ with $M_i = (b^{\Sigma_i}, \cdot, \emptyset)$ and trace w_j is given as $(w_j = '\sigma_j \cdot ''\sigma_j \cdots) \in M_j$ of hyper events $\sigma_j \in b^{\Sigma_j}$ with $M_j = (b^{\Sigma_j}, \cdot, \emptyset)$. The uniquely de-composed trace $w_{||}$ is given as $(w_{||} = '\sigma_{||} \cdot ''\sigma_{||} \cdots) \in M_{||}$ of hyper events $\sigma_{||} \in b^{\Sigma_{||}}$ with $M_{||} = (b^{\Sigma_{||}}, \cdot, \emptyset)$ and $\Sigma_{||} = \Sigma_i \cup \Sigma_j$.

Axiom 5 (Idempotency)**.** $\rho^{-1}(\sigma_i, \sigma_i) = \sigma_i$ *for* $M_i \ni \sigma_i \in M_j$

3.2.1 Automata Based Composition (ABC)

In the following, we will employ the σ-Codec (Def. 1) as the functor to induce the Automata Based Composition.

Definition 2 (Automaton)**.** Let $A_i = (Z_i, b^{\Sigma_i}, z_{i,0}, Z_{Fi})$ be a 4-tuple with index i. Then, A_i specifies a possibly incomplete (partial) deterministic finite automaton with set of states Z_i, structured set of inputs (hyper events) b^{Σ_i}, initial state $z_{i,0} \in Z_i$ and set of final states $Z_{Fi} \subseteq Z_i$. Let $M_i = (b^{\Sigma_i}, \cdot, \emptyset)$ denote the corresponding trace monoid over b^{Σ_i}. Let $\sigma_i \in b^{\Sigma_i}$ act on Z_i by $z_i' = z_i\sigma_i$, and $\emptyset$ acts as the identity on Z_i. By sequential application M_i acts also on Z_i.

Definition 3 (ABC)**.** We define the automaton $A_{||} = A_i||A_j = (Z_{||}, b^{\Sigma_{||}}, z_{||,0}, Z_{F||})$ of two automata A_i and A_j with set of states $Z_{||} = Z_i \times Z_j$, set of hyper events $b^{\Sigma_{||}}$, set of single events $\Sigma_{||} = \Sigma_i \cup \Sigma_j$, initial state $z_{||,0} = (z_{i,0}, z_{j,0})$ and set of final states $Z_{F||} = Z_{Fi} \times Z_{Fj}$. Then, the commutative diagram in Fig. 5 defines the kernel for the ABC algorithm.

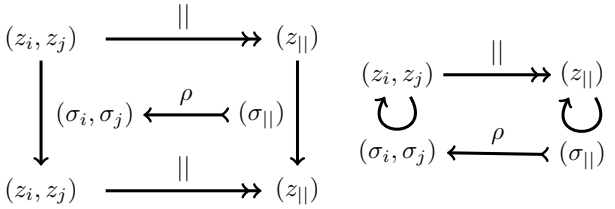


Fig. 5: Commutative diagram (expanded on the left side, abbreviated on the right side), which defines the kernel for the ABC algorithm.

3.2.2 Formalism and language properties

In the following we refer to some basic formalisms and properties of the ABC. The epimorphic property of ρ^{-1} (Axiom 1, Fact) warrants the overall consistency of the ABC. Imagine that n parallel system components are (associatively, see Axiom 2) composed to an overall component system, $A_1||A_2||...||A_n = A_{||}$, and let a scenario (actions, trace, scanning, sampling, sensing) of length m is specified (captured, stored). Then, the following formalism represents this scenario, with $\sigma(k) = \sigma(k-1) \cdot {}^k\sigma$ and $\sigma(-1) = \emptyset$:

$$^{\mathrm{a}}z_{||}(\sigma_{||}(0), \sigma_{||}(1), \cdots, \sigma_{||}(m-1)) = {}^{\mathrm{n}}({}^1z_{||}, {}^2z_{||}, \cdots, {}^mz_{||})$$

$$^{\mathrm{a}}\begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix}^T \begin{pmatrix} \sigma_1(0) & \sigma_1(1) \cdots \sigma_1(m-1) \\ \vdots & \vdots \quad \vdots \quad \vdots \\ \sigma_n(0) & \sigma_n(1) \cdots \sigma_n(m-1) \end{pmatrix} = {}^{\mathrm{n}}\begin{pmatrix} {}^1z_1 & {}^2z_1 \cdots {}^mz_1 \\ \vdots & \vdots \quad \vdots \quad \vdots \\ {}^1z_n & {}^2z_n \cdots {}^mz_n \end{pmatrix}^T$$

The transpose of the right side of the equation preserves the component-wise transpose of the sequential data structure. $z_1, ..., z_n$ are the particular states of the n parallel entities. $\sigma_i(m-1) = {}^0\sigma_i \cdot {}^1\sigma_i \cdot \cdots \cdot {}^{m-1}\sigma_i \in M_i$ is a word (trace) coding the sequence of m actions on A_i whereby each action consists of particular events which are simultaneously occurring within the respective action. ${}^0z_i{}^1z_i{}^2z_i \cdots {}^mz_i$ is the row vector of particular states of length $m+1$ representing the behavioral scenario of A_i on acting $\sigma_i(m-1)$. Correspondingly alike in case of $A_{||}$. Obviously, the commutativity (Axiom 3) induces the isomorphism between $A_i||A_j \cong A_j||A_i$. Further, the neutral element (Axiom 4) induces the identical entity (identity) $1_{||}$ of the ABC up to isomorphism, $1_{||}||A_i \cong A_i||1_{||} \cong A_i$. The idempotency (Axiom 5) even induces the idempotency up to isomorphism, $A_i||A_i \cong A_i$.

The properties of the ABC are also allowing to define a rather topologically and statistically motivated behavioral representation[2]. For a given entity A_i, consider a $(|Z_i| + |\Sigma_i|)$-dimensional linear vector space. Then, the

2. like e.g. in case of the so called barcodes ([20])

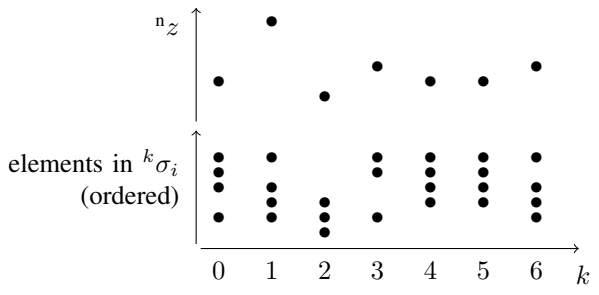


Fig. 6: A Fingerprint Code of the trace $^{0}\sigma_i \cdot {}^{1}\sigma_i \cdot \ldots \cdot {}^{k}\sigma_i$ and the induced behavioral scenario $^{a}z_i\, {}^{1}z_i\, {}^{2}z_i \ldots {}^{k}z_i\, {}^{n}z_i$ in A_i.

diagram in Figure 6 represents a sequence $^{0}\sigma_i \cdot {}^{1}\sigma_i \cdot \ldots \cdot {}^{k}\sigma_i$ and an induced behavioral scenario $^{a}z_i\, {}^{1}z_i\, {}^{2}z_i \ldots {}^{k}z_i\, {}^{n}z_i$ in A_i, which is in fact a trace in the corresponding linear vector space. With regard to the commutative diagram (Fig. 5) in Def. 3 the ABC can be interpreted as the composition and respectively decomposition of linear vector spaces composing and respectively decomposing particular corresponding traces in each linear vector space. The fingerprint code (FPC) can be seen as a suitable histogram representation for all kind of data based management, analysis and verification.

Complexity and Scalability: In the following, we discuss the properties of the formalism regarding to complexity. The universal property together with the associativity of the ABC warrant the ABC kernel to be realized as a non-hierarchic algorithm. This is an essential feature, since the algorithm can decide to start to operate with favorable entities regarding to a cost function, hence the ABC operation becomes easily tunable for specific applications. In addition, the underlying commutativity is a big advantage, too, since the entities can algorithmically get rearranged for the ABC such that the efficiency increases regarding to the specific quality structure of the entities. The idempotency ensures the ABC to discard redundant information. Then again, the isomorphism allows the entities to be designed for preserving sufficient information after de-composition using the multi-set property of the formalism.

4 Conclusion

In this work, we have presented an overall theoretical framework in a low as well as high level abstraction to model, validate and verify an asynchronous feedback network under balance (NUB). As opposed to synchronous networks which are globally master-clocked and hence totally defined, asynchronous networks are partially defined. Thus, functionally stable operating procedure for asynchronous NUB needs to be warranted especially for safety critical systems. It is well known that widely established simulation techniques are not by themselves adequate to ensure the correctness of complex systems. On the other hand, although there exist powerful theoretical foundations for formal modeling and verification, almost all of these theories are based on totally defined algebra, e. g. Boolean algebra, Boolean ring and digital automata theory. Thus, adequate theoretical framework is needed, which is inherently capable to work with partiality. In addition, it has to be closed under inversion. This is an essential requirement for the purposes of test and diagnosis. Furthermore, the theory has to support simultaneously parallel and associative composition and decomposition to being able to authentically model the inherent simultaneous and parallel operating behavior of the asynchronous NUB. Our theoretical framework fulfills these requirements. It is based on limit diagram and universal construction of the theory of categories. We build the data flow channel using monomorphisms and epimorphisms which formally warrant the structure and behavior preserving injective and projective morphisms. It contributes maximum possible flow of correct information specified and controlled by any collection of free bases. Therefore, the σ-Codec provides unique information to observe and detect corrupt information. We show how to test and localize corruptness and to avoid feeding back such inconsistent information into the NUB. This is an interesting feature in view of a correct-by-construction paradigm as the NUB becomes able to do self-check at any time and in all situations. This means, that the NUB can "axiomatically halt itself and then axiomatically continue again" using delay-insensitive asynchronous synchronization elements. Thus, "safely preserving the last correct information" and "safely preventing the propagation of inconsistent data" becomes warranted.

References

[1] M. Ipek, "Eine Testfallspezifikation für das funktionsorientierte Testen von reaktiven eingebetteten Systemen im Automobilen Bereich," Ph.D. dissertation, Universität Kaiserslautern, Mai 2011.

[2] F. Johannes, "Partitioning of vlsi circuits and systems," in *Design Automation Conference Proceedings 1996, 33rd*, 1996, pp. 83–87.

[3] H. J. Zander, *Logischer Entwurf binärer Systeme*, 3rd ed. Verlag Technik Berlin, 1989.

[4] N. Hartmann, "Automation des Tests eingebetteter Systeme am Beispiel der Kraftfahrzeugelektronik," Ph.D. dissertation, Fakultät Elektrotechnik und Informationstechnik der Universität Fredericiana Karlsruhe, Januar.

[5] R. Negulescu, "Process spaces and formal verification of asynchronous circuits," Ph.D. dissertation, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, Aug. 1998.

[6] C. H. K. v. Berkel, M. B. Josephs, and S. M. Nowick, "Scanning the technology: Applications of asynchronous circuits," *Proceedings of the IEEE*, vol. 87, no. 2, pp. 223–233, Feb. 1999.

[7] P. Shirvani, S. Mitra, J. Ebergen, and M. Roncken, "Dudes: a fault abstraction and collapsing framework for asynchronous circuits," in *Advanced Research in Asynchronous Circuits and Systems, 2000. (ASYNC 2000) Proceedings. Sixth International Symposium on*, 2000, pp. 73–82.

[8] F. Gurkaynak, T. Villiger, S. Oetiker, N. Felber, H. Kaeslin, and W. Fichtner, "A functional test methodology for globally-asynchronous locally-synchronous systems," in *Asynchronous Circuits and Systems, 2002. Proceedings. Eighth International Symposium on*, 2002, pp. 181–189.

[9] V. Khomenko, M. Schaefer, W. Vogler, and R. Wollowski, "STG decomposition strategies in combination with unfolding," *Acta Informatica*, vol. 46, pp. 433–474, September 2009.

[10] S. Nowick and D. Dill, "Exact Two-Level Minimization of Hazard-Free Logic with Multiple-Input Changes," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 14, no. 8, pp. 986 –997, aug 1995.

[11] M. Schaefer, *Advanced STG Decomposition*. Books on Demand GmbH, 2008.

[12] *International Technology Roadmap for Semiconductors, Design. (2005)*.

[13] D. Shang, E. University of Newcastle upon Tyne. School of Electrical, and C. Engineering, *Asynchronous Communication Circuits: Design, Test and Synthesis*. University of Newcastle upon Tyne, 2003.

[14] Heinz-Dietrich Wuttke and Karsten Henke, *Schaltsysteme - Eine automatenorientierte Einführung*. Pearson Studium, 2003.

[15] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *SIAM Journal on Computing*, no. 1, pp. 77 – 84, 1975.

[16] J. Adámek, H. Herrlich, and G. E. Strecker, "Abstract and concrete categories: the joy of cats," *Repr. Theory Appl. Categ.*, no. 17, pp. 1–507, 2006, reprint of the 1990 original [Wiley, New York].

[17] Y. Hirshfeld and F. Moller, "Pushdown automata, multiset automata, and petri nets," *Theor. Comput. Sci.*, vol. 256, no. 1-2, pp. 3–21, 2001.

[18] R. P. Kurshan and K. L. McMillan, "Analysis of digital circuits through symbolic reduction," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 10, no. 11, pp. 1356–1371, 1991.

[19] G. Uygur and S. Sattler, "A novel formalism for partially defined asynchronous feedback digital circuits," *Journal of Electronic Testing*, vol. 29, no. 5, pp. 697–714, 2013.

[20] A. Collins, A. Zomorodian, G. Carlsson, and L. J. Guibas, "A barcode shape descriptor for curve point cloud data," *Computers & Graphics*, vol. 28, no. 6, pp. 881 – 894, 2004.