A Unified Coding Framework for Delay-Insensitivity

Frédéric Worm, Patrick Thiran and Paolo Ienne
Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
CH-1015 Lausanne, Switzerland

E-mail: {Frederic.Worm,Patrick.Thiran,Paolo.Ienne}@epfl.ch

Abstract

We study the problem of delay-insensitivity from a coding point of view. We define sequencing rules*, i.e., the rules dictating which symbols can follow which. For example, spacer-based encoding schemes obey a sequencing rule that requires a spacer to be inserted after each symbol conveying information. We define mathematically delay-insensitive sequencing rules and study the ones which maximize bandwidth efficiency.*

1 Introduction

Delay-insensitive codes (also referred to as self-synchronizing codes) enable to check whether transitions occurring on a binary vector have completed or not. As a result, they detect any error originating from a bit transition failure. We abstract bit transition failure by considering communication over the timing error channel [8]. A timing error channel introduces errors only for those bits that transition. Such errors are called timing errors. In comparison with a binary symmetric channel that can output any possible binary vector, a timing error channel outputs deterministically correct bit values for all bits that do not transition. On the contrary, bits that transition are corrupted with a certain probability. Fig. 1 gives a graphic description of a timing error channel with input x_k and output y_k. By definition, delay-insensitive codes detect all timing errors. Therefore, they achieve completely reliable communication over a timing error channel (i.e., no error is undetected).

1.1 Related work

The problem of delay-insensitivity has been extensively studied in a framework that assumes only two sequencing rules. Classic schemes either insert a spacer after each symbol conveying information (e.g., dual rail), or apply differential encoding where data transmitted on the channel consists of the values of signal changes, rather than signal values themselves. In this classic framework, delay-insensitive codes with minimum redundancy are known for decades [1][4][7][6]. These codes are called optimal delay-insensitive codes. The novelty of this work is to consider sequencing rules that, in general, do not fit the framework described above. That is, we consider a larger family of encoding schemes that includes the framework just mentioned. One interest is to ascertain whether such a generalization could lead to improvements on bandwidth efficiency. As a motivational example, consider two encoding schemes. The first one alternates codewords of an optimal delay-insensitive code with a spacer. The second one transmits codewords of a non-optimal (i.e., non minimum redundancy) delay-insensitive code, but can interleave several symbols as spacers. Which of these two delay-insensitive schemes is the more bandwidth efficient? The question can be generalized: do

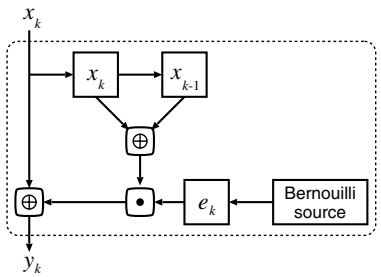

Figure 1. Timing error channel. The operator $\oplus$ denotes the *exclusive-or*, while the operator $\cdot$ denotes the binary *and*. Each time a transition occurs, i.e., each time $x_k \oplus x_{k-1} = 1$, there is a probability that the transition fails, i.e., that $y_k = x_{k-1}$.

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**

delay-insensitive coding schemes exist that achieve a better bandwidth efficiency than optimal codes used with sequencing rules of the classic framework? Indeed, when considering any possible sequencing rule, it is not clear whether optimal delay-insensitive codes, which minimize code redundancy, also maximize bandwidth efficiency.

1.2 Contributions

We study delay-insensitivity within a novel coding framework: sequencing rules. We define bandwidth efficiency, which is a more general metric than code redundancy. We give an upper-bound on the bandwidth efficiency of delay-insensitive sequencing rules and show that differential encoding (refer to Ex. 3) using delay-insensitive codes with minimum redundancy achieves this upper-bound. In spite of deceiving the hope emitted in Sec. 1.1, this is in itself a significant contribution. Furthermore, this paper presents a new and unified mathematical modelling of delay-insensitivity, which includes most well-known and implementable codes, but does not address explicitly hardware complexity issues.

1.3 Outline

Sec. 2 formulates an assumption as to how encoding schemes determine whether transitions are occurring on a piece of data. Sec. 3 introduces sequencing rules and shows that they describe many practical encoding schemes. Sec. 4 defines mathematically delay-insensitivity and characterizes delay-insensitive sequencing rules. Sec. 5 expands on a key notion introduced in Sec. 4: self-synchronizing sets. Sec. 6 elaborates on the question of conveying information with a sequencing rule. Then, Sec. 7 quantifies the bandwidth efficiency of sequencing rules. In Sec. 8, we characterize sequencing rules maximizing bandwidth efficiency. As a conclusion, Sec. 9 summarizes our achievements.

1.4 Notations

The letter N is reserved for the channel width. Unless stated differently, a letter in subscript denotes the time index (for example, x_{10} is the eleventh symbol to be emitted, or the symbol emitted at time index 10). Although time has not been defined formally so far, we adopt a convention where time is discrete and incremented after the emission of a new symbol. The *cardinality* operator is denoted by $|\cdot|$. The *weight* of a binary vector is the number of 1 it contains. We denote the weight of $u \in \{0,1\}^N$ as $\mathrm{w}(u)$. We denote by $\overline{u}$ the logical negation of u. The *exclusive-or* operator is denoted by $\oplus$. Moreover, the operator $\oplus$ applied to a binary vector and a set of binary vectors is to be understood as follows.

Notation 1. *Let u be a binary vector, $u \in \{0,1\}^N$. Let V be a set of binary vectors, $V \subset \{0,1\}^N$. We denote by $u \oplus V$ the set defined by*

$$u \oplus V = \left\{ w \in \{0,1\}^N \mid w = u \oplus v,\ v \in V \right\}.$$

2 Readiness and membership test

This section explains our assumption as to how an encoding scheme determines whether the data received at the channel output effectively contains a new piece of information. We call this feature *readiness*, as underlined in the following definition.

Definition 1. (Data readiness). *A binary data is* ready *if and only if all transitions from the previous piece of data have completed.*

We want to transfer an information sequence that is encoded into a symbol sequence $\{x_k\}_{k\in\mathbb{N}}$ over a timing error channel . We denote by x_k the k^{th} symbol emitted by the encoder, and by y_k the output of the timing error channel for the input x_k. According to Def. 1, a data y_k at the channel output is ready if and only if $y_k = x_k$. Let D_k be the set of symbols x_k that can be emitted at time k. We call the sequence $\{D_k\}_{k\in\mathbb{N}}$ the *sequence of decoding sets*. If y_k is ready, then $y_k \in D_k$. The converse is not necessarily true. However, as in any syndrome decoding scheme, we assume that this test is used to detect whether a given output is ready or not: *The output y_k is declared ready if and only if $y_k \in D_k$.* As a practical implication of this assumption, the membership test has to be implementable with glitch-free logic.

Until now, we have made almost no assumption about encoding schemes we consider. We have only assumed the existence of a sequence of decoding sets. In the next section, we define and study sequencing rules, i.e., encoding schemes whose sequence of decoding sets obeys a particular rule.

3 Sequencing rules

In Sec. 2, we have introduced the sequence of decoding sets. In particular, we have explained that the decoding method only consists in applying a membership test between y_k and D_k. However, we have not precised how the decoding set is obtained from one time index to the next. In the most general case, a decoding set is a function of the whole transfer history $D_k = \mathrm{F}(x_0, x_1, ..., x_{k-1}, k)$. For practical reasons, we

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**

restrict our considerations to encoding schemes with "Markovian" history. More precisely, we focus on schemes having the property that each decoding set is a function only of the last symbol emitted and the time index. We call such schemes sequencing rules.

Definition 2. (Sequencing rule). *A coding scheme is a* sequencing rule *if and only if the sequence of decoding sets is such that* $D_k = F(x_{k-1}, k)$, $\forall k \geq 1$.

We now define sets that are useful to describe sequencing rules.

Definition 3. (Symbol set). *The symbol set* S *of a sequencing rule is the subset of* $\{0,1\}^N$ *containing all symbols that the encoder may output:*

$$S = \left\{ s \in \{0,1\}^N \mid \exists k \in \mathbb{N} \text{ and } \exists a \text{ sequence of symbols } x_0, x_1, ..., x_k \text{ with } x_k = s \right\}.$$

Without loss of generality, we assume that the symbol set is ordered, i.e., we have, assuming an arbitrary ordering, $S = \{s^0, s^1, ..., s^{|S|-1}\}$. *We define the* symbol index *as the index of a symbol in the symbol set. By convention, we write the symbol index in superscript.*

It follows from Def. 2 that, for each time index and for each symbol, we can define a set of symbols that may follow it and will constitute the decoding set of the next time index if the symbol is emitted.

Definition 4. (The set Succ (s,k)). *For a sequencing rule, we define*

$$Succ(s,k) = \{ p \in S \mid \exists \text{ a sequence of symbols such that } x_{k+1} = p \text{ and } x_k = s \}.$$

In a sequencing rule, the set $\text{Succ}(s,k)$ is such that $D_{k+1} = \text{Succ}(s,k)$ if $x_k = s$.

We point out that Def. 2 does not describe explicitly how a sequencing rule conveys information, i.e., how codes can be used in a sequencing rule. We address this question later in Sec. 6 and illustrate now in Fig. 2 a few sequences of symbols that can be generated according to a given sequencing rule. The following examples give the sequencing rules of various encoding schemes.

Example 1. (Standard codes). *Consider an error detecting code* C *(e.g., a CRC). The corresponding sequencing rule is easily obtained. We have* $S = D_0 = C$ *and* $Succ(c,k) = C$, $\forall c \in C$ *and* $\forall k \in \mathbb{N}$.

Example 2. (Alternating-phase codes [8]). *For an alternating-phase code alternating between the codes* C_0 *and* C_1, *we have* $S = C_0 \cup C_1$. *Moreover, we have either* $D_0 = C_0$, *or* $D_0 = C_1$. *Lastly, we have,* $\forall c \in C_0$ *and* $\forall k \in \mathbb{N}$, $Succ(c,k) = C_1$ *and* $\forall c \in C_1$ *and* $\forall k \in \mathbb{N}$, $Succ(c,k) = C_0$.

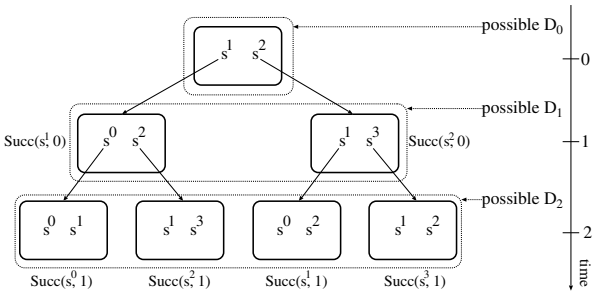

Figure 2. The different symbol sequences that can be generated up to time index 2, for a 2-bit sequencing rule with symbol set $S = \{s^0, s^1, s^2, s^3\}$.

Example 3. (Differential encoding). *Consider a* (N,K) *code* C. *Let* $c(u)$ *be the* N*-bit codeword obtained by encoding the* K*-bit information* u. *Differential encoding is a sequencing rule that outputs* $x_k = x_{k-1} \oplus c(u)$ *to transmit the information* u *in the* k^{th} *symbol* x_k $(k > 0)$. *We have,* $\forall s \in S$ *and* $\forall k \in \mathbb{N}$, $Succ(s,k) = s \oplus C$. D_0 *contains only one arbitrary symbol.* $S = C$ *if* C *is linear and* $D_0 \subset C$. *LEDR [3] can be described with differential encoding.*

The next section defines mathematically delay-insensitivity and characterizes delay-insensitive sequencing rules.

4 Delay-insensitive sequencing rules

The timing error channel enables to give a compact definition of delay-insensitive codes [8]: delay-insensitive codes achieve completely reliable communication over a timing error channel. Based on this definition, we derive a necessary and sufficient condition (namely, Prop. 1) for a sequencing rule to be delay-insensitive. Before doing so, we define two relations.

Definition 5. (Ordering relation). *Let* u *and* v *be two vectors of* $\{0,1\}^N$. *We say that* $u \leq v$ *when the ordering relation holds component-wise, i.e.,* $u \leq v \Leftrightarrow u_i \leq v_i$, $\forall i = 1, ..., N$, *with the index in subscript denoting the bit index.*

Definition 6. (Self-synchronizing set for a symbol). *Let* $s \in \{0,1\}^N$ *be a binary vector. Let* $S \subset \{0,1\}^N$ *be a set of binary vectors. We define* S *as a* self-synchronizing set *for* s *if and only if the two following conditions are met:*

(i) $s \notin S$ *and*
(ii) $\forall p \in S$, $\nexists\, q \in S \backslash \{p\}$, *such that* $q \oplus s \leq p \oplus s$.

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**

We use the notation $s \prec S$ to denote that S is a self-synchronizing set for s.

Def. 6 essentially states that, if one wishes to transmit the symbol s and then an arbitrary symbol p, there must be no other symbol $q \in S$ requiring a subset of the transitions needed to send p. From Def. 6, we derive easily the following property.

Property 1. (Delay-insensitive sequencing rule). *Let S be the symbol set of a sequencing rule. A sequencing rule is delay-insensitive if and only if $\forall s \in S$ and $\forall k \in \mathbb{N}$, $Succ(s,k)$ is a self-synchronizing set for s, i.e., $s \prec Succ(s,k)$.*

Proof. Over a timing error channel, an undetected error may occur if and only if the sequencing rule contains three distinct symbols, $s \in S$, $p, q \in \mathrm{Succ}(s,k)$ such that $q \oplus s \leq p \oplus s$, i.e., if and only if $s \not\prec \mathrm{Succ}(s,k)$. □

As a consequence of Def. 6, the decoding sets of a delay-insensitive sequencing rule satisfy the condition $\forall k \in \mathbb{N}$ and $\forall s \in D_k, s \notin D_{k+1}$, which expresses the intuitive idea that a delay-insensitive encoding has to "change something" to indicate data readiness. Therefore, standard codes discussed in Ex. 1 cannot be delay-insensitive. Because self-synchronizing sets are very convenient for describing delay-insensitivity, we study them in more depth in the next section.

5 Properties of self-synchronizing sets

We first state a few basic properties of self-synchronizing sets without proof, for the sake of brevity. A singleton set S is a self-synchronizing set for a symbol s if and only if $S \neq \{s\}$. In addition, any subset of a self-synchronizing set for a symbol is also a self-synchronizing set for this symbol. We state a last property, showing that the existence and the cardinality of a self-synchronizing set for a vector s does not depend on the choice of s.

Property 2. (Symbol-independence of self-synchronizing sets). *Let $s, r \in \{0,1\}^N$ be two binary vectors. Let $S \subset \{0,1\}^N$ be a self-synchronizing set for s. Define C as $s \oplus S$. We call C a transition set, i.e., a set of transition vectors. Then, $r \oplus C$ is a self-synchronizing set for r with the same cardinality as S.*

Proof. Clearly, $|S| = |C| = |r \oplus C|$. Let us show that $r \oplus C$ is a self-synchronizing set for r. We proceed by contradiction, and assume that there exists two distinct vectors $t, u \in r \oplus C$ such that $r \oplus t \leq r \oplus u$. Since $t, u \in r \oplus C$, we can write $t = r \oplus c_1$ with $c_1 \in C$ and $u = r \oplus c_2$ with $c_2 \in C$. We obtain therefore $c_1 \leq c_2$, which implies that S is not a self-synchronizing set for s. □

Does Prop. 2 suggest that studying self-synchronizing sets (such as S) for a particular symbol is unnecessarily complex since self-synchronizing transition sets (such as C) have similar properties, while being symbol-independent? Actually not, because self- synchronizing sets for a symbol are more general than self-synchronizing transition sets. Indeed, Prop. 2 shows that a self-synchronizing transition set such as C is a self-synchronizing set for the all-zero symbol. Moreover, self-synchronizing sets for a symbol enable to describe a larger family of encoding schemes, e.g., symbol invariant sequencing rules (refer to Sec. 6) which are not defined in terms of transitions.

When considering self-synchronizing sets which are not singletons, a question arises naturally. Given an arbitrary symbol, what is the largest self-synchronizing set for this symbol. Before answering the question, we need preliminary definitions and results, which we introduce in Sec. 5.1. Then, Sec. 5.2 exposes key results on largest self-synchronizing sets for a symbol and for a set of symbols. The consequences of the results presented in Sec. 5.2 will be discussed in Sec. 8.

5.1 Preliminary results

We define first a maximum self-synchronizing set.

Definition 7. (Maximum self-synchronizing set for a symbol). *Let $s \in \{0,1\}^N$. Let $S \subset \{0,1\}^N$ be a self-synchronizing set for s. S is a* maximum *self-synchronizing set for s if and only if there exists no other self-synchronizing set for s, $T \subset \{0,1\}^N$, such that $|T| > |S|$.*

We also define equivalence classes in $\{0,1\}^N$. Let $s \in \{0,1\}^N$ and consider the equivalence relation between N-bit binary vectors: $p \equiv q$ if and only if $\mathrm{w}(s \oplus p) = \mathrm{w}(s \oplus q)$ (the dependence on s is not written explicitly). From this equivalence relation, we derive $N+1$ equivalence classes.

Definition 8. (Equivalence classes in $\{0,1\}^N$). *Let $s \in \{0,1\}^N$. We define $N+1$ equivalence classes in $\{0,1\}^N$: $V_N^i(s) = \{p \in \{0,1\}^N \mid w(s \oplus p) = i\}$, with $0 \leq i \leq N$.*

Def. 8 implies that $V_N^i(s)$ contains exactly all vectors that are at Hamming distance i from s. Lastly, we define a graph structure on $\{0,1\}^N$.

Definition 9. (Graph on $\{0,1\}^N$). *Let $s \in \{0,1\}^N$. We define a graph structure, $\mathbb{G}(s)$, on $\{0,1\}^N$. An arc goes from a node $u \in \{0,1\}^N$ to a node $v \in \{0,1\}^N$ if and only if $u \neq v$ and $s \oplus v \leq s \oplus u$.*

We use the notation $u \rightarrow v$ to mean that an arc of $\mathbb{G}(s)$ connects a node u to a node v. As illustration, Fig. 3 depicts the graph $\mathbb{G}(010)$. We give basic

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**

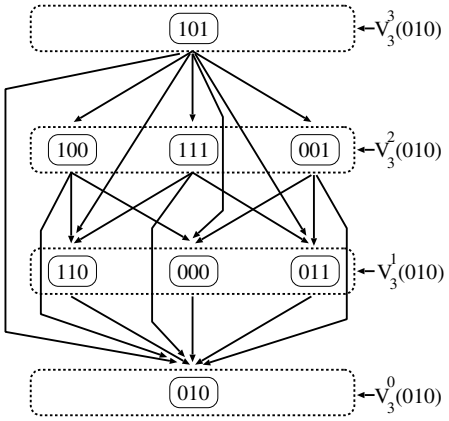

Figure 3. The graph $\mathbb{G}(010)$. The equivalence classes are drawn with a dotted line. The claims of Lemma 1 can be verified on this example.

properties of the graph structure defined in Def. 9.

Lemma 1. *A node that belongs to equivalence class $V_N^i(s)$ has edges*

(i) only towards nodes in equivalence classes $V_N^j(s)$, with $j < i$,
(ii) towards i nodes in equivalence class $V_N^{i-1}(s)$, and
(iii) from $N - i$ nodes in equivalence class $V_N^{i+1}(s)$.

The proof of Lemma 1 is simple and omitted for brevity. For a set of binary vectors Q, define $i_{\max}(Q) = \max_{0\leq j\leq N}\left\{j \mid Q \cap V_N^j(s) \neq \emptyset\right\}$. In other words, $i_{\max}(Q)$ is the maximum index of an equivalence class that contains at least one element of Q. We show the following claim.

Lemma 2. *Let S be a self-synchronizing set for a vector s. Assume that $i_{max}(S) > \left\lfloor \frac{N}{2} \right\rfloor$. Then, there exists a self-synchronizing set for s, S', such that (i) $i_{max}(S') = i_{max}(S) - 1$, and (ii) $|S'| = |S|$.*

Proof. Let $R = S \cap V_N^{i_{\max}(S)}(s)$. Knowing R, we define the following set of vectors:

$$A = \left\{a \in V_N^{i_{\max}(S)-1}(s) \mid \exists r \in R \text{ such that } r \to a\right\}.$$

By definition of A and because S is a self-synchronizing set for s, $S \cap A = \emptyset$. We observe that $(S\backslash R) \cup A$ is a self-synchronizing set for s. We prove this claim by contradiction, and assume that $(S\backslash R) \cup A$ is not a self-synchronizing set for s. This implies that $\exists a, p \in (S\backslash R) \cup A$ such that $a \to p$. Because $S\backslash R$ is a self-synchronizing set for s, we have necessarily $a \in A$. We have therefore a situation where $r \to a \to p$ with $r \in R$. This implies in turn $r \to p$, which is impossible since S is a self-synchronizing set.

We obtain S' from S by performing $|R|$ iterations defined as follows. Assume that R is arbitrarily ordered: $R = \left\{r_1, ..., r_{|R|}\right\}$. Let $S_0 = S$. Define, for $i = 0, ..., |R|-1$, $S_{i+1} = (S_i \backslash \{r_i\}) \cup \{a_i\}$, where $a_i \in A$ such that $r_i \to a_i$. We have to show that it is effectively possible to perform $|R|$ iterations, i.e., that there exists at least $|R|$ vectors $a_i \in A$ such that $r_i \to a_i$ for $i = 0, ..., |R| - 1$. This is feasible if and only if $|A| \geq |R|$. We show that the latter inequality holds by reasoning on the number of arcs that connect vectors of R to vectors of A. According to (ii) of Lemma 1, this number of arcs is exactly $|R| \cdot i_{\max}(S)$. However, each vector in A is pointed by at most $N - (i_{\max}(S) - 1)$ vectors of R (due to (iii) of Lemma 1). Therefore, this very same number of arcs is upper-bounded by $|A| \cdot (N - (i_{\max}(S) - 1))$, so that we have the relation $|R| \cdot i_{\max}(S) \leq |A| \cdot (N - (i_{\max}(S) - 1))$. We write thus

$$\frac{|R|}{|A|} \leq \frac{N - (i_{\max}(S) - 1)}{i_{\max}(S)} \leq 1,$$

where the last inequality holds if $i_{\max}(S) \geq \frac{N+1}{2} \geq \left\lfloor \frac{N}{2} \right\rfloor$, which is true by assumption.

Finally, we show that $S' = S_{|R|}$ fulfils items (i) and (ii) of the claim. By construction, item (ii) is verified as $|S| = |S_i|$ for $i = 1, ..., |R|$. Moreover, the construction ensures that $i_{\max}\left(S_{|R|}\right) = i_{\max}(S) - 1$. At last, since $S_{|R|} \subset (S\backslash R) \cup A$ and $(S\backslash R) \cup A$ is a self-synchronizing set for s, $S_{|R|}$ is also a self-synchronizing set for s. As a result, $S_{|R|}$ satisfies (i) and (ii). □

We state a last claim, very similar to Lemma 2. For a set of binary vectors Q, define $i_{\min}(Q) = \min_{0\leq j\leq N}\left\{j \mid Q \cap V_N^j(s) \neq \emptyset\right\}$.

Lemma 3. *Let S be a self-synchronizing set for a vector s, and assume that $i_{min}(S) < \left\lfloor \frac{N}{2} \right\rfloor$. Then, there exists a self-synchronizing set for s, S', such that (i) $i_{min}(S') = i_{min}(S) + 1$, and (ii) $|S'| = |S|$.*

A proof similar to the one of Lemma 2 is possible; we omit it for brevity.

5.2 Maximum self-synchronizing sets

The following theorem constructs a maximum self-synchronizing set.

Theorem 1. (Maximum self-synchronizing set for a symbol). *Let $s \in \{0,1\}^N$. Define W_N as*

$$W_N = \left\{q \in \{0,1\}^N \mid w(q) = \left\lfloor \frac{N}{2} \right\rfloor\right\}.$$

$s \oplus W_N$ is a maximum self-synchronizing set for s.

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**

Proof. We have the set equality $V_N^{\lfloor \frac{N}{2} \rfloor}(s) = s \oplus W_N$. Indeed, $s \oplus W_N$ contains all vectors that are at Hamming distance $\lfloor \frac{N}{2} \rfloor$ of s, which is exactly the definition of $V_N^{\lfloor \frac{N}{2} \rfloor}(s)$. Due to (i) of Lemma 1, we obtain that $s \oplus W_N$ is a self-synchronizing set for s. Let S be another self-synchronizing set for s. We have to show that $|S| \leq |s \oplus W_N|$. Thanks to Lemma 2 and 3, we know that, from S, we can obtain another self-synchronizing set for s, S', such that (a) $|S'| = |S|$, and (b) $i_{\max}(S') = i_{\min}(S') = \lfloor \frac{N}{2} \rfloor$. However, (b) implies that $S' \subset V_N^{\lfloor \frac{N}{2} \rfloor}(s)$. As a result, $S' \subset s \oplus W_N$, which in turn implies $|S'| \leq |s \oplus W_N|$. Finally, combining with (a), we get the desired result $|S| \leq |s \oplus W_N|$. □

As expected from Prop. 2, the cardinality of a maximum self-synchronizing set only depends on the size of the vector space, N, and not on the choice of a particular s. Moreover, we remark that differential encoding using the code W_N is a sequencing rule maximizing the number of symbols that can be emitted (refer to Ex. 3). We give another result about maximum self-synchronizing sets.

Theorem 2. (Maximum self-synchronizing set for a set). *Let $s \in \{0,1\}^N$ be a binary vector. Let S be a maximum self-synchronizing set for s. Let $Q = \left\{ q \in \{0,1\}^N \mid S \text{ is a self-synchronizing set for } q \right\}$. Then, $Q = \{s, \overline{s}\}$.*

Proof. For brevity, we only sketch the proof of this theorem. It relies on the fact that a maximum self-synchronizing set for a vector has a known specific form. Indeed, it can be shown that if S is a maximum self-synchronizing set for s, either $S = s \oplus W_N$ or $S = s \oplus X_N$, where $X_N = \left\{ q \in \{0,1\}^N \mid \mathrm{w}(q) = \lceil \frac{N}{2} \rceil \right\}$. We show the claim for an even N. As N is assumed even, we have $W_N = X_N$. Because a maximum self-synchronizing set has a unique form, we also have, $\forall q \in Q$, $S = s \oplus W_N = q \oplus W_N$. Equivalently, we write

$$s \oplus q \oplus W_N = W_N. \tag{1}$$

We show that Eq. (1) has no other solutions than $q = s$ and $q = \overline{s}$. Let $t \in \{0,1\}^N$ such that $t \oplus W_N = W_N$. We assume first that $0 < \mathrm{w}(t) < \frac{N}{2}$. Then, there exists a vector $u \in W_N$, such that $t \leq u$. As a result, $\mathrm{w}(t \oplus u) < \frac{N}{2}$, which implies $t \oplus u \notin W_N$. Now, assume that $\mathrm{w}(t) = \frac{N}{2}$, i.e., $t \in W_N$. Let $u = \overline{t} \in W_N$. Since $\mathrm{w}(t \oplus u) = N$, $t \oplus u \notin W_N$. Finally, assume that $\frac{N}{2} < \mathrm{w}(t) < N$. Then, there exists a vector $u \in W_N$, such that $u \leq t$. As a result, $\mathrm{w}(t \oplus u) < \frac{N}{2}$, which implies $t \oplus u \notin W_N$. In conclusion, we have shown that, if there exists $t \in \{0,1\}^N$ such that $t \oplus W_N = W_N$, we have necessarily $\mathrm{w}(t) = N$ or $\mathrm{w}(t) = 0$, i.e., $Q = \{s, \overline{s}\}$. If N is odd, a similar procedure is possible. □

We give a last result that characterizes maximum *systematic* self-synchronizing sets. Systematicity of a set is defined as follows.

Definition 10. (Systematic set of binary vectors). *Let $S \subset \{0,1\}^N$ and let K, $0 < K \leq N$, be an integer. S is K-systematic if and only if $\forall u \in \{0,1\}^K$, there exists a unique $s \in S$ such that s is the concatenation of u and v, for a certain vector $v \in \{0,1\}^{N-K}$.*

For example, a systematic (N, K) code constitutes a K-systematic set of 2^K N-bit codewords, each one being obtained by concatenating redundant bits to information bits.

Theorem 3. (Maximum systematic self-synchronizing set). *Let $s \in \{0,1\}^N$. Let $S \subset \{0,1\}^N$ be a self-synchronizing set for s. Then, S is K-systematic implies $N \geq K + \log_2(K+1)$.*

Proof. We assume that S is K-systematic and consider N as an unknown constant. We are looking for an integer R such that concatenating an R-bit vector to every vector in $\{0,1\}^K$ yields a self-synchronizing set for s. K, R and N therefore satisfy $K + R = N$ and we have to show that $R \geq \log_2(K+1)$. We denote by $r(x) \in \{0,1\}^R$ the R-bit vector concatenated to the K-bit vector $x \in \{0,1\}^K$.

According to Def. 8, we define $K+1$ equivalence classes $V_K^i(s_K)$ for $i = 0, ..., K$, where s_K is the K-bit vector formed by the K most significant bits of the N-bit vector s. We also define $R+1$ equivalence classes $V_R^i(s_R)$ for $i = 0, ..., R$, where s_R is the R-bit vector formed by the R least significant bits of the N-bit vector s. We denote by $(u \mid v)$ the concatenation of vector v to vector u. Because $\{0,1\}^R = \bigcup_{i=0}^{R} V_R^i(s_R)$, there exists orderings of vectors of $\{0,1\}^R$, $\{0,1\}^R = \{r_0, ..., r_{2^R-1}\}$ such that $s_R \oplus r_i \not\leq s_R \oplus r_j$ if $i < j$. For example, take $r_0 \in V_R^R(s_R)$, $r_i \in V_R^{R-1}(s_R), i = 1, ..., \begin{pmatrix} R \\ R-1 \end{pmatrix}$, etc. We claim that the following construction (which describes the Berger code [1]) forms a self-synchronizing set for s . For $i = 0, ..., K$ and $\forall u \in V_K^i(s_K)$, $r(u) = r_i$. Indeed, assume that there exists $u, v \in \{0,1\}^K$, $u \neq v$, such that $s_K \oplus u \leq s_K \oplus v$. This implies $\mathrm{w}(s_K \oplus u) < \mathrm{w}(s_K \oplus v)$. By construction, there exists two integers i and j such that $r(v) = r_j$, $r(u) = r_i$, and $i < j$. As $i < j$, we have $r_i \not\leq r_j$, which ensures $s \oplus (u \mid r(u)) \not\leq s \oplus (v \mid r(v))$. Therefore, the construction gives a self-synchronizing set for s with $2^R - 1 = K$, i.e., $R = log_2(K+1)$.

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**

It remains to show that $R = log_2(K+1)$ is the smallest integer needed to obtain a self-synchronizing set for s. Take $K+1$ distinct vectors $u_i \in V_K^i(s_K)$, for $i = 0, ..., K$, such that $u_0 \leq u_1 \leq ... \leq u_K$. For S to be a self-synchronizing set for s, we must have $r(u_i) \neq r(u_j)$, $\forall i \neq j$, $0 \leq i, j \leq K$. As a result, the mapping $r(\cdot)$ defines at least $K+1$ different values, i.e., we have necessarily $2^R \geq K+1$. □

According to Thrm. 3, if a delay-insensitive sequencing rule can output any possible K-bit vector (i.e., 2^K different vectors), then the size of the symbols emitted is at least $N = K + \log_2(K+1)$.

6 Codes and sequencing rules

So far, we have defined sequencing rules as the rules dictating which symbols can follow which. We now study the question of how a sequencing rule conveys information. We discuss two specific types of sequencing rules, namely *time* and *symbol* invariant sequencing rules, that are most relevant from an implementation point of view.

Definition 11. (Time invariant sequencing rule). *A sequencing rule is* time invariant *if and only if the set of symbols that may follow a given symbol is not a function of the time index, i.e.,* $\forall s \in S$, *and* $\forall k \in \mathbb{N}$, $Succ(s,k) = Succ(s)$.

Definition 12. (Symbol invariant sequencing rule). *A sequencing rule is* symbol invariant *if and only if the set of symbols that may follow a given symbol is not a function of the symbol index, i.e.,* $\forall k \in \mathbb{N}$, *and* $\forall x_k \in S$, $Succ(x_k,k) = Succ(k)$.

The encoder structures of time and symbol invariant sequencing rules are depicted in Fig. 4. We point out that encoding schemes described by time invariant sequencing rules have already been proposed for other applications, such as bus energy reduction [5].

We discuss in more depth time and symbol invariant sequencing rules in Sections 6.1 and 6.2. For this purpose, we define a *code* as a mapping between binary information vectors and other binary vectors called codewords. Moreover, we assume the existence of a *coding set*, which is an ordered set of distinct codes.

Definition 13. (Coding set). *A coding set is an ordered set of codes* $\{C_0, C_1, ..., C_{Q-1}\}$ *such that (i)* $C_i \subset \{0,1\}^N$, $\forall i = 0, ..., Q-1$, *and (ii)* $C_i \neq C_j$, $\forall i \neq j$.

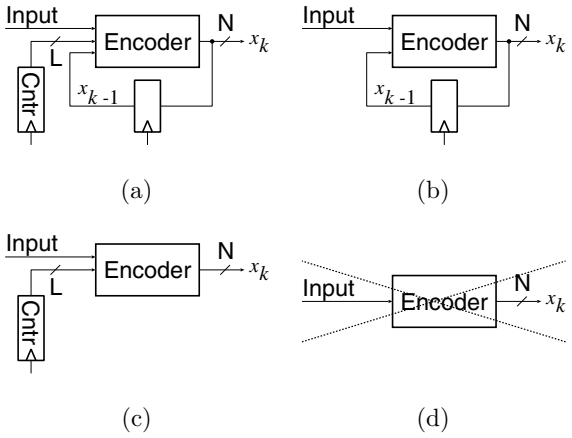

Figure 4. Encoder structure of (a) a general, (b) a time-invariant, and (c) a symbol-invariant sequencing rule. Case (d) cannot represent a delay-insensitive encoder.

6.1 Time invariant sequencing rules

It follows from Def. 11 that a time invariant sequencing rule is entirely specified by a finite collection of sets, namely $\text{Succ}(s^j)$, for $j = 0, ..., |S|-1$. We have, of course, $S = \bigcup_{j=0}^{|S|-1} \text{Succ}(s^j)$. We consider a time invariant sequencing rule described by $|S|$ sets $\text{Succ}(s^j)$, $j = 0, ..., |S|-1$. For the sequencing rule to convey information, we have to define the relation between a coding set and each of the sets $\text{Succ}(s^j)$, $j = 0, ..., |S|-1$, as performed in Prop. 3

Property 3. (Time invariant sequencing rule). *A sequencing rule is time invariant if and only if there exists a coding set* $\{C_0, C_1, ..., C_{Q-1}\}$, *and a mapping* M *from* $\{0, ..., |S|-1\}$ *to* $\{0, ..., Q-1\}$ *such that,* $\forall j = 0, ..., |S|-1$, $Succ(s^j) = s^j \oplus C_{M(j)}$.

Proof. We start by assuming that there exists a coding set $\{C_0, C_1, ..., C_{Q-1}\}$ and a mapping $M : \{0, ..., |S|-1\} \mapsto \{0, ..., Q-1\}$ such that $\text{Succ}(s^j) = s^j \oplus C_{M(j)}$. This means that the set $\text{Succ}(s^j, k)$ is only function of the symbol index j. Therefore, the sequencing rule is time invariant. Now, we prove the reverse implication, and consider a time invariant sequencing rule. We show how to build a coding set and a mapping M as claimed. We proceed iteratively over the symbol set. We initialize the coding set to an empty set, $C = \emptyset$. Then, we do the following iterations: $\forall j = 0, ..., |S|-1$, we define $C_j = s^j \oplus \text{Succ}(s^j)$. If $C_j \notin C$, then we add C_j to C and set $M(j) = j$. Otherwise, there exists an integer $i < j$ such that $C_i = C_j$. In this case, we set $M(j) = i$. After a finite number of iterations, we have

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**

constructed a coding set, namely C, and a mapping M as claimed. □

We give an important example of time invariant sequencing rule.

Example 4. (Differential encoding). *Consider a singleton coding set: $C_0 \subset \{0,1\}^N$. The symbol set is taken (a priori) as $S = \{0,1\}^N$. The mapping M is defined as follows:*

$$M : \{0, 1, ..., 2^N - 1\} \mapsto \{0\}$$
$$M(j) = 0,\ \forall j = 0, 1, ..., N.$$

As a result, we have $Succ(s) = s \oplus C_0$, $\forall s \in S$. We point out that LEDR is a particular case of differential encoding, where the symbol set is given by $S = \{0,1\}^2$ and $C_0 = \{(0,1),(1,0)\}$.

Based on Prop. 1, we characterize a delay-insensitive time invariant (*DITI*) sequencing rule. Because any time invariant sequencing rule can be expressed as defined in Prop. 3, we give a condition on the coding set.

Property 4. (DITI sequencing rule). *Consider a time invariant sequencing rule described by a coding set and a mapping M, as defined in Prop. 3. Let $\vec{0} \in \{0,1\}^N$ be the all-zero vector. The sequencing rule is delay-insensitive if and only if $\vec{0} \prec C_i$, $\forall i = 0, ..., Q-1$.*

Proof. For any $s^j \in S$, each symbol $s \in \text{Succ}\left(s^j\right)$ can be written as $s = s^j \oplus p$, with $p \in C_{M(j)}$. Applying Prop. 1 with this notation gives the result. □

We derive from Prop. 4 that finding delay-insensitive of time invariant sequencing rules is equivalent to finding *unordered* codes, i.e., codes detecting all unidirectional errors [2]. The next section gives properties and examples of symbol invariant sequencing rules.

6.2 Symbol invariant sequencing rules

Proceeding as in Sec. 6.1, we first characterize the coding set of a symbol invariant sequencing rule.

Property 5. (Symbol invariant sequencing rule). *A sequencing rule is symbol invariant if and only if there exists a coding set $\{C_0, C_1, ..., C_{Q-1}\}$ and a mapping M from $\mathbb{N}$ to $\{0, ..., Q-1\}$ such that, $\forall k \in \mathbb{N}$, $Succ(x_k) = C_{M(k)}$ and the initial decoding set $D_0 \in \{C_0, C_1, ..., C_{Q-1}\}$.*

Proof. We start by assuming that there exists a coding set $\{C_0, C_1, ..., C_{Q-1}\}$, and a mapping $M : \mathbb{N} \mapsto \{0, ..., Q-1\}$ such that $\text{Succ}(x_k) = C_{M(k)}$ and $D_0 \in \{C_0, C_1, ..., C_{Q-1}\}$. Obviously, this means that the set $\text{Succ}(x_k, k)$ is only function of the time index k. Therefore, the sequencing rule is symbol invariant. Now, we prove the reverse implication, and consider a symbol invariant sequencing rule. We show how to build a coding set and a mapping M as claimed. We give an iterative construction. We initialize the coding set to D_0, i.e., $C = \{D_0\}$. Then, we do the following iterations: $\forall k \geq 0$, define $C_{k+1} = \left\{p \in \{0,1\}^N \mid p \in \text{Succ}(q,k), q \in C_k\right\}$. If $C_{k+1} \notin C$, then we add C_{k+1} to C and set $M(k+1) = k+1$. Otherwise, there exists an integer $j \leq k$ such that $C_j = C_{k+1}$. In this case, we set $M(k+1) = j$. By iterating infinitely, we construct a coding set, namely C, and a mapping M as claimed. □

We illustrate Prop. 5 with an example.

Example 5. (Alternating-phase codes [8]). *Consider a coding set $\{C_0, C_1\}$. Let $S = C_0 \cup C_1$ be the symbol set. Let M be a mapping defined by*

$$M : \mathbb{N} \mapsto \{0,1\}$$
$$M(k) = k \mod 2.$$

Then, we obtain

$$Succ(x_k) = \begin{cases} C_1 & \text{if } k \mod 2 = 1, \\ C_0 & \text{if } k \mod 2 = 0. \end{cases}$$

Dual rail is a particular case where $C_0 = \{(0,0)\}$ and $C_1 = \{(0,1),(1,0)\}$. Notice also that, in Ex. 2, we have described alternating-phase codes as a time invariant sequencing rule. Therefore, some particular encoding schemes can be described by a time invariant or by a symbol invariant sequencing rule. However, there is no general method to express a symbol invariant sequencing rule as a time invariant sequencing rule, and vice-versa.

We now characterize the coding set of a delay-insensitive symbol invariant (*DISI*) sequencing rule.

Property 6. (DISI sequencing rule). *Consider a symbol invariant sequencing rule defined by its coding set $\{C_0, C_1, ..., C_{Q-1}\}$ and the mapping M of Prop. 5. The sequencing rule is delay-insensitive if and only if, $\forall i = 0, 1, ..., Q-1$, $\forall c \in C_i$, and $\forall j$ such that $\exists k \in \mathbb{N}$ with $M(k-1) = i$ and $M(k) = j$, we have $c \prec C_j$.*

Proof. By direct application of Prop. 1, a symbol invariant sequencing rule is delay-insensitive if and only if $c \prec C_{M(k)}$, $\forall k \in \mathbb{N}$ and $\forall c \in C_{M(k-1)}$. □

The fundamental difference between time and symbol invariant sequencing rules stems from the convention (namely, the mapping M) they define. Symbol invariant sequencing rules define a convention based on

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**

the time index k. Indeed, the encoder and decoder have to keep track of how many symbols have been emitted. In practice, this would be done by the counter (Cntr) drawn in Fig. 4(a) and (c). On the contrary, time invariant sequencing rules define a convention bearing on the symbol space: the encoder and decoder have to keep track of the last emitted symbol, as shown in Fig. 4(b).

The next section quantifies the bandwidth efficiency of sequencing rules.

7 Bandwidth efficiency

Bandwidth efficiency is a direct generalization of the rate of a code. The bandwidth efficiency of a sequencing rule informs on what percentage of the data transferred over the channel effectively conveys information. Therefore, at similar reliability, the higher the bandwidth efficiency, the better the coding scheme. Vice-versa, at similar bandwidth efficiency, the better the reliability, the better the coding scheme. We would like to quantify bandwidth efficiency of a sequencing rule and proceed as follows. Assume that K information bits $u_0, ..., u_{K-1}$ are encoded into T N-bit symbols, $T \in \mathbb{N}$. In all generality, it may be that $T \geq K$ or $T < K$, and K may not be an integer. For the transfer of these K information bits, a total of $\prod_{i=0}^{T-1} 2^N = 2^{TN}$ different sequences of bit vectors could have been generated. However, among them, there are only $\prod_{i=0}^{T-1} |D_i|$ different sequence of symbols that carry information. As a result, the bandwidth efficiency for the transfer of information bits $u_0, ..., u_{K-1}$ is given by

$$\begin{aligned}\eta(u_0, ..., u_{K-1}) &= \frac{\log_2\left(\prod_{i=0}^{T-1} |D_i|\right)}{\log_2(2^{TN})} \\ &= \frac{\log_2\left(\prod_{i=0}^{T-1} |D_i|\right)}{T \cdot N}, \end{aligned} \quad (2)$$

where T is the number of symbols needed to encode the information bit sequence $u_0, ..., u_{K-1}$. By averaging over all possible sequences of K information bits, we obtain the following definition of bandwidth efficiency.

Definition 14. (Bandwidth efficiency). *The bandwidth efficiency of a sequencing rule is*

$$\eta(K) = \sum_{u_0,...,u_{K-1} \in \{0,1\}^K} [\eta(u_0, ..., u_{K-1}) \cdot P(u_0, ..., u_{K-1})],$$

where $\eta(u_0, ..., u_{K-1})$ has been defined in Eq. (2) and $P(u_0, ..., u_{K-1})$ is the probability of the information bit sequence $u_0, ..., u_{K-1}$.

Let us now go through a few examples.

Example 6. (A time invariant sequencing rule). *Consider a 2-bit wide time invariant sequencing rule defined by $Succ((0,0)) = Succ((1,1)) = \{(0,1),(1,0)\}$, $Succ((1,0)) = \{(0,0),(1,1)\}$, $Succ((0,1)) = \{(1,0)\}$, and $D_0 = \{(0,0),(1,1)\}$. Assume that all information bits are equally likely, and that we transfer $K = 3$ information bits u_0, u_1, u_2. If the second emitted symbol (conveying information u_1) is $(1,0)$, then we have $T = K = 3$ and $\prod_{i=0}^{T-1} |D_i| = 2^K = 2^3$. On the contrary, if the second emitted symbol is $(0,1)$, we have this time $T = 4$, and $\prod_{i=0}^{T-1} |D_i| = 2^K = 2^3$. Both cases being equally likely, we obtain*

$$\eta(3) = \frac{3}{3 \cdot N} \cdot \frac{1}{2} + \frac{3}{4 \cdot N} \cdot \frac{1}{2} = \frac{1}{4} + \frac{3}{16} = \frac{7}{16}.$$

As a last example, we compute the bandwidth efficiency of differential encoding with the Sperner code [4].

Example 7. (Differential encoding with Sperner code). *Consider a N-bit time invariant sequencing rule defined by the relation $Succ(s) = s \oplus W_N$, $\forall s \in S$ (W_N has been defined in Thrm. 1). Assume that every information bit is equally likely and consider the transfer of a sequence of K information bits. Let T be the number of symbols emitted to convey a sequence of K information bits $u_0, ..., u_{K-1}$ (actually, $T = \frac{K}{\log_2(|W_N|)}$; however the exact value of T is not needed). We have*

$$\prod_{i=0}^{T-1} |D_i| = \prod_{i=0}^{T-1} |W_N| = |W_N|^T.$$

As a result, we obtain, $\forall K \geq 1$,

$$\eta(K) = \frac{\log_2 |W_N|^T}{T \cdot N} = \frac{\log_2 |W_N|}{N}.$$

In the next section, we upper-bound the bandwidth efficiency of delay-insensitive sequencing rules. A delay-insensitive sequencing rule is called *optimal* if it achieves this upper-bound.

8 Optimal delay-insensitive sequencing rules

We summarize here key results that characterize optimal delay-insensitive sequencing rules.

Property 7. *Any N-bit delay-insensitive sequencing rule has a bandwidth efficiency less than or equal to $\frac{\log_2 |W_N|}{N}$, where W_N has been defined in Thrm. 1. Moreover, any N-bit, $N > 2$ symbol invariant sequencing rule has a bandwidth efficiency less than $\frac{\log_2 |W_N|}{N}$.*

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**

Proof. At any time index, a delay-insensitive sequence rule cannot transmit more symbols than contained in a maximum self-synchronizing set. We know from Thrm. 1 that the size of a maximum self-synchronizing set for a symbol is $|W_N|$, independently of the symbol chosen. Henceforth, any delay-insensitive sequencing cannot transmit more than $|W_N|$ different symbols at each time index. This is exactly what the sequencing rule of Ex. 7 does. As a result, it has the largest possible bandwidth efficiency.

Now, we sketch the proof showing that, if $N > 2$, any symbol invariant sequencing rule cannot achieve a bandwidth efficiency of $\frac{\log_2 |W_N|}{N}$. Let $\{C_0, ..., C_{Q-1}\}$ be the coding set of a symbol invariant sequencing rule that is assumed to achieve the upper-bound on bandwidth efficiency we just stated. As it achieves this upper-bound, the symbol invariant sequencing rule transmits each time one symbol among $|W_N|$. Therefore, there exists necessarily two indices i and j such that (a) $|C_i| = |C_j| = |W_N|$ and (b) C_j is a maximum self-synchronizing set, $\forall c \in C_i$. According to Thrm. 2, C_i only contains one symbol and its logical negation, i.e., $|C_i| = 2$. This proves that (a) and (b) can only be satisfied when $N = 2$. □

Property 8. *Any delay-insensitive sequencing rule using a systematic encoding for K bits has a bandwidth efficiency less than or equal to $\frac{K}{N}$, where $N \geq K + \log_2(K+1)$.*

Proof. A systematic delay-insensitive sequencing rule cannot transmit more symbols than contained in a maximum systematic self-synchronizing set. According to Thrm. 3, the cardinality of such a set is 2^K if $N \geq K + \log_2(K+1)$. □

Prop. 7 means that sending symbols that are each time at Hamming distance $\lfloor \frac{N}{2} \rfloor$ from the preceding one is an optimal coding strategy for delay-insensitivity. Moreover, this cannot be performed by a symbol invariant sequencing rule, unless $N = 2$. If a systematic code is required, differential encoding using the Berger code is optimal, as suggested by Prop. 8 and Thrm. 3. Even though both coding constructions are simple, practical implementations only exist for $N = 2$ (i.e., LEDR), due to the limitations imposed by glitch-free membership testing.

We now express compactly the problem of delay-insensitivity for time and symbol invariant sequencing rules. As already stated in Prop. 4, a delay-insensitive time invariant sequencing rule uses bandwidth optimally with an unordered code of minimum redundancy. It is worth pointing out that the coding set of an optimal time invariant sequencing rule needs not contain more than one code. On the contrary, characterizing

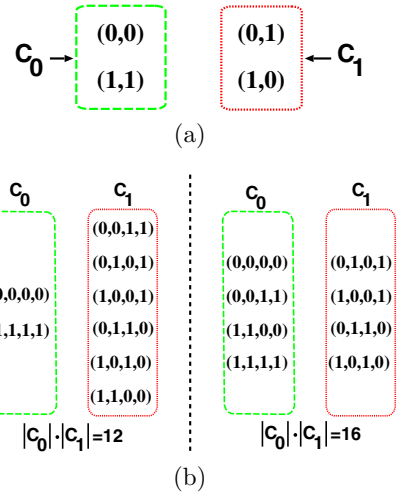

Figure 5. For $N = 2$ and $N = 4$, LEDR ((a) and right side of (b)) solves Prob. 1. In (a) and the left side of (b), $|C_0| = 2$ because C_1 is a maximum self-synchronizing set for each symbol of C_0, as claimed in Thrm. 2.

optimal delay-insensitive symbol invariant (DISI) sequencing rules is a tough problem. Referring to Prop. 6 and to Fig. 4, we know that the coding set of a DISI sequencing rule has to contain at least two codes. Optimal DISI sequencing rules with a coding set containing two codes solve the following problem.

Problem 1. (Optimal DISI sequencing rules using two codes). *Find two codes $C_0, C_1 \subset \{0,1\}^N$ such that (i) we have $c \prec C_1$ and $d \prec C_0$, $\forall c \in C_0, \forall d \in C_1$, and (ii) $|C_0| \cdot |C_1|$ is maximized.*

We have verified that LEDR solves Prob. 1 for $N = 2, 4$ and 6. Fig. 5 sketches the solutions for $N = 2$ and $N = 4$. In the particular case $N = 2$, the solution (i.e., LEDR) is still based on unordered codes of minimum redundancy. However, the solution of Prob. 1 does not use unordered codes of minimum redundancy when $N > 2$. Indeed, minimum redundancy only maximizes the cardinality of one code, while restricting the other to two codewords (this is a consequence of Thrm. 2). What and how many codes form the coding set of optimal DISI sequencing rules when $N > 2$ is an open question. Table 1 compares the optimality of a few codes when used in time and symbol invariant sequencing rules. Finally, Fig. 6 plots the ratio of the bandwidth efficiency of various codes to the bandwidth efficiency of the Sperner code, assuming a sequencing rule with spacers or differential encoding. Interestingly, dual rail and LEDR always have a bandwidth efficiency that is at least 55 % of optimal.

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**

codes	optimal in DITI sequencing rule	optimal in DISI sequencing rule
1-of-N	if $N = 2$	if $N = 2$
Sperner	yes	if $N = 2$
Berger	if $N = 2$	if $N = 2$
LEDR	if $N = 2$	if $N = 2, 4, 6$ (only?)

Table 1. Optimality of a few well-known delay-insensitive codes when used in time and symbol invariant (resp. DITI and DISI) sequencing rules .

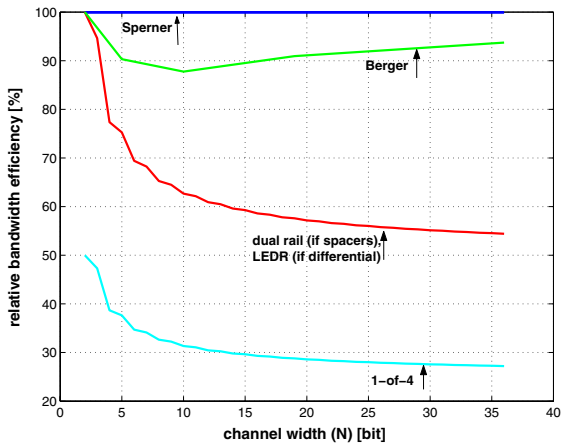

Figure 6. Relative bandwidth efficiency of various codes with respect to the Sperner code, for a spacer-based sequencing rule or differential encoding.

9 Conclusions

We have reviewed the question of delay-insensitivity from a coding point of view. Though, we have not addressed hardware complexity issues. We have defined sequencing rules, i.e., the rules that determine which sequences of symbols can be generated. We have upper-bounded the bandwidth efficiency of delay-insensitive sequencing rules and shown that differential encoding with optimal delay-insensitive codes, such as the Sperner or Berger codes, achieves this upper-bound. It was already known that such optimal codes maximize bandwidth efficiency for the particular cases of spacer-based and differential encoding schemes, where maximum bandwidth efficiency is obtained with codes of minimum redundancy. Our contribution is to give an achievable upper-bound (Prop. 7) on bandwidth efficiency of any delay-insensitive sequencing rule. Then, we have discussed optimal delay-insensitive time and symbol invariant sequencing rules. We have shown that a time invariant delay-insensitive sequencing rule such as differential encoding achieves this upper-bound. However, symbol invariant delay-insensitive sequencing rules do not achieve the bound when $N > 2$, and maximize bandwidth efficiency using unordered codes which are not minimum redundancy.

As future work, we will study Prob. 1 and sequencing rules that exhibit strong resilience to both additive and timing errors.

References

[1] J. M. Berger. A note on error detecting codes for asymmetric channels. *Information and Control*, 4(1):68–71, Mar. 1961.

[2] B. Bose and T. Rao. Theory of unidirectional error correcting/detecting codes. *IEEE Transactions on Computers*, C-31(6):521–30, June 1982.

[3] M. E. Dean, T. E. Williams, and D. L. Dill. Efficient self-timing with level-encoded two-phase dual-rail (LEDR). In *Proceedings of the 1991 University of California/Santa Cruz Conference on Advanced Research in VLSI*, pages 55–70. MIT Press, Mar. 1991.

[4] C. V. Freiman. Optimal error detecting codes for asymmetric binary channels. *Information and Control*, 5(1):64–71, Mar. 1962.

[5] P. P. Sotiriadis and A. P. Chandrakasan. Bus energy reduction by transition pattern coding using a detailed deep submicrometer bus model. *IEEE Transactions on Circuits and Systems I—Fundamental Theory and Applications*, CAS1-50(10):1280–95, Oct. 2003.

[6] V. I. Varshavsky, editor. *Self-Timed Control of Concurrent Processes*. Kluwer Academic, Dordrecht, The Netherlands, 1990.

[7] T. Verhoeff. Delay-insensitive codes—an overview. *Distributed Computing*, 3(1):1–8, Jan. 1988.

[8] F. Worm, P. Ienne, and P. Thiran. Soft self-synchronising codes for self-calibrating communication. In *Proceedings of the International Conference on Computer Aided Design*, San Jose, Calif., Nov. 2004.

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**