Memory-based Cross-talk Canceling CODECs for On-chip Buses

Chunjie Duan*, Kanupriya Gulati†, Sunil P Khatri†

† (kgulati,sunil)@ece.tamu.edu

*duan@merl.com

† Department of Electrical and Computer Engineering

Texas A&M University

College Station, TX 77843

* Mitsubishi Electric Research Laboratory

Cambridge, MA 02139

Outline

- Motivation and Introduction
- Preliminaries and Notation
- Previous Work
- Memory Based Cross-talk Canceling CODECs
  - Overview
  - Mathematical Formulation
- Results
- Conclusions and Future Work

Motivation and Introduction

- Ratio of the cross-coupling capacitance between adjacent on-chip wires on the same metal layer to the total capacitance of any wire is becoming quite large.
- This results in significant delay variation and noise immunity problems, limiting system performance.
- This problem is aggravated for long on-chip buses.
- In this work, we present memory-based crosstalk canceling CODECs for on-chip buses.
  - Our bus overheads are lower than for a memoryless CODEC, and have been quantified in this work.
  - User may trade off the speed gain against the attendant bus size overhead, by using our approach

Preliminaries and Notation

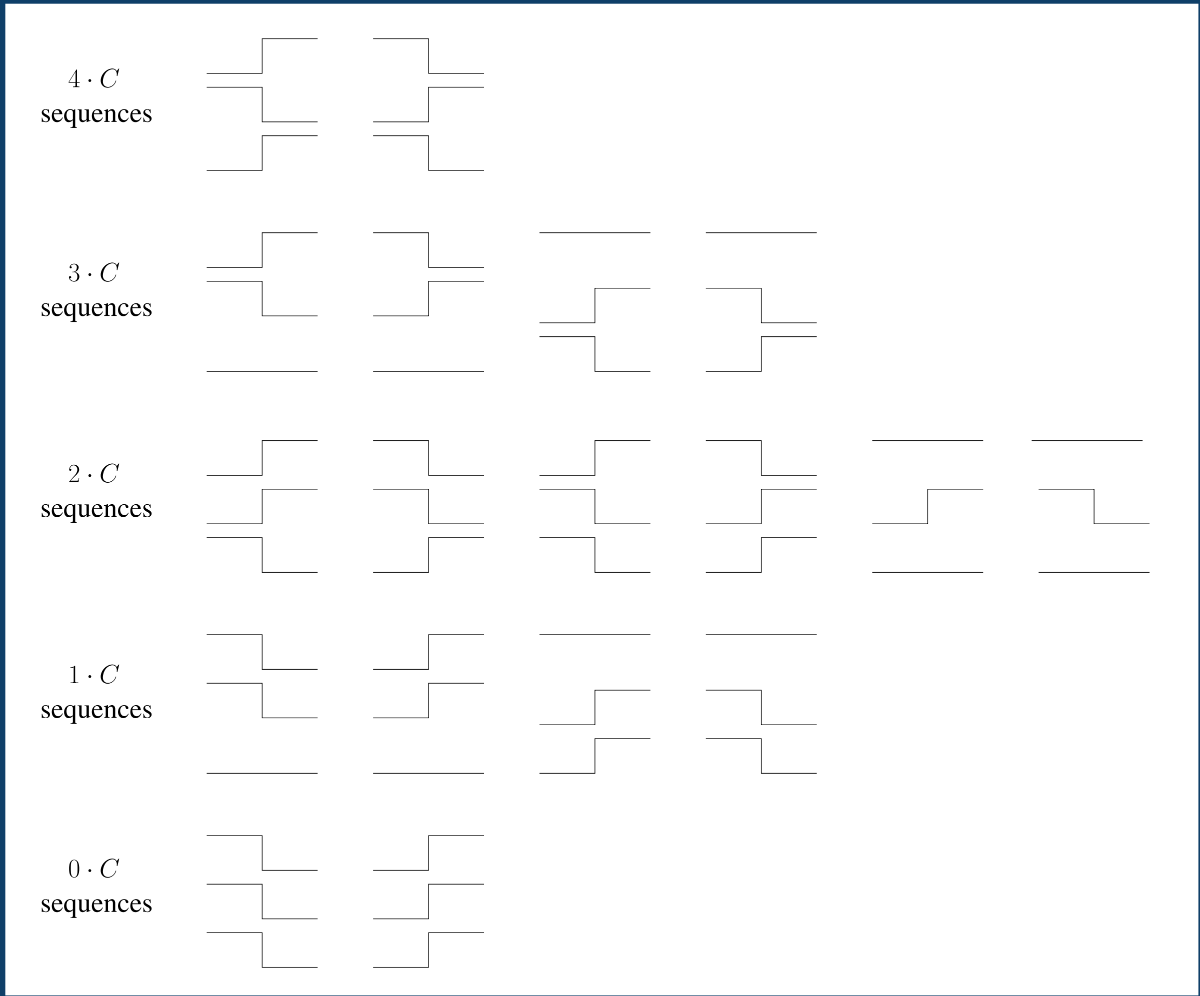
- Consider an n-bit bus, consisting of signals $b_1, b_2, b_3 \cdots b_{n-1}, b_n$.

Definition 1 *: A* **Vector** v *is an assignment to the signals* b_i *as follows:*
$b_i = v_i$, *(where* $1 \leq i \leq n$ *and* $v_i \in \{0, 1\}$*).*

- Consider two successive vectors v^j and v^{j+1}, being transmitted on a bus.
  - For vector v^j, assume $b_i = v_i^j$ ($1 \leq i \leq n$ and $v_i^j \in \{0, 1\}$).
  - For vector v^{j+1}, assume $b_i = v_i^{j+1}$ ($1 \leq i \leq n$ and $v_i^{j+1} \in \{0, 1\}$).
- Consider a vector sequence $v^1, v^2, \cdots, v^j, v^{j+1}, \cdots v^k$ (of k n-bit vectors) applied on a bus.
  - We define five types of crosstalk sequences next.

Preliminaries and Notation ... 2

- For any three physically adjacent bits in the bus, and for any temporally adjacent vectors (a vector pair), if any one of the conditions below occurs, then the bus is classified as such.



Preliminaries and Notation ... 3

Definition 2 *A* $p \cdot C$ **crosstalk canceling CODEC** *(or* $p \cdot C$ *crosstalk free CODEC) transforms an arbitrary* m*-bit vector sequence into a* n*-bit vector sequence* $(m < n)$ *such that the output vector sequence is a* $(p-1) \cdot C$ *sequence.*

Definition 3 *A set* C_n *of* n*-bit vectors is said to be a* $p \cdot C$ **crosstalk free clique** *iff any vector sequence* $v_1 \rightarrow v_2$ *made up of vectors* $v_1, v_2 \in C_n$ *is a* $l \cdot C$ *sequence (where* $l < p$*), and there exists* $v_1^*, v_2^* \in C_n$ *such that* $v_1^* \rightarrow v_2^*$ *is a* $(p-1) \cdot C$ *sequence.*

A **memoryless CODEC** simply encodes an m bit vector with a unique n bit vector. A **memory-based CODEC** encodes an m bit vector with an n bit vector. The encoding depends on the k previous n bit vectors that were transmitted on the bus (for a memory depth k).
Note that in the sequel, *if we say that a CODEC is* $kC - free$*, we mean that it results in cross-talk of magnitude* $(k-1)C$ *or less, for any bus transition*.

Previous Work

- In 2001, [SoCh] suggest that CODECs could be used for buses, to eliminate 4C and 3C sequences.
- In 2001, [DuTiKh] demonstrated memoryless CODECs to eliminate 4C and 3C sequences, using an inductive construction process.
- In 2001, [ViKu] discuss memoryless and memory-based CODECs for crosstalk cancellation. Method is based on *explicit enumeration* of all 2^{2n} vector transitions.
  - In contrast, our approach employs implicit enumeration, and also cancels 2C crosstalk.
- In 2004, [DuKh] describe 2C and 1C cross-talk canceling memoryless CODECs.

Our approach is applicable for cancelling all types of crosstalk, using a unified, implicit formulation. It can actually speed up the bus by exploiting crosstalk among neighboring wires

Memory-based Cross-talk Canceling CODECs

- Let v_r be the vector present on the bus at time t_r.
- Let v_{r+1} be the vector present on the bus at time t_{r+1}.
- If it is guaranteed that for any r, $v_r \rightarrow v_{r+1}$ is a $p \cdot C$ transition, then the sequence is a $p \cdot C$ sequence (sufficient condition).
- A memory-based CODEC will satisfy the $(p+1) \cdot C$ free condition iff for each vector v in the set, there are at least 2^m vectors (including v itself) that are $(p+1) \cdot C$ free with respect to v.
    - It is *not* required that every pair of vectors in the set is a $(p+1) \cdot C$ free pair.
- To decode the data, the receiving decoder needs to know both the current received symbol *and* the previously received symbol. As a consequence, memory elements are needed in both the encoder and decoder.

Summary of our Approach

Our approach to determine the effective bus of width m that can be encoded in a $k \cdot C$ free manner, using a physical bus of width n consists of two steps:

- First, we construct an ROBDD $G_n^{kC-free}$ which encodes all vector transitions on the n-bit bus that are $k \cdot C$ free.
- Then, from $G_n^{kC-free}$, we find the effective bus width m, such that an m bit bus can be encoded in a $k \cdot C$ free manner using $G_n^{kC-free}$.

These steps are described in the sequel.

Efficient Construction of $G_n^{kC-free}$

- We employ an ROBDD based implicit construction of $G_n^{kC-free}$
  - We avoid explicit enumeration of legal kC-free vectors.
  - Implicit computation allows sharing of ROBDD nodes maximally, and in a canonical manner.
- In particular, we inductively compute $\overline{G_n^{kC-free}}$
  - Since the ROBDD of a function and its complement contain the same number of nodes (except for a complement pointer), this enables an efficient construction of $G_n^{kC-free}$
- We next show how this is done.

Efficient Construction of $G_n^{kC-free}$ ... 2

- To construct $G_n^{kC-free}$, we allocate $2n$ ROBDD variables.
  - The first n variables correspond to the vector from which a transition is made (referred to as $v = \{v_1, v_2, \cdots, v_n\}$).
  - The next n variables correspond to the vector to which a transition is made (referred to as $w = \{w_1, w_2, \cdots, w_n\}$).
  - If a vector sequence $v^* \rightarrow w^*$ is legal with respect to $k \cdot C$ crosstalk, then $w^* \rightarrow v^*$ is also legal.
- We construct the ROBDD for $G_n^{kC-free}$ by using ROBDDs for intermediate, partially $k \cdot C$ cross-talk free ROBDDs $\overline{G_i^{kC}}$ $(3 \leq i \leq n)$.
- The construction of the ROBDD of G_n^{kC} proceeds iteratively, starting with the base case of G_3^{kC}.

Efficient Construction of $G_n^{4C-free}$

$$G_3^{4C} = \left[\begin{array}{cccccc|cccccc} \multicolumn{6}{c}{\overbrace{v_1 \quad v_2 \quad v_3 \quad v_4 \quad \cdots \quad v_n}^{v}} & \multicolumn{6}{c}{\overbrace{w_1 \quad w_2 \quad w_3 \quad w_4 \quad \cdots \quad w_n}^{w}} \\ 1 & 0 & 1 & - & \cdots & - & 0 & 1 & 0 & - & \cdots & - \\ 0 & 1 & 0 & - & \cdots & - & 1 & 0 & 1 & - & \cdots & - \end{array} \right]$$

- Note that the ROBDD for $\overline{G_3^{4C}}$ is *only partially free of* $4 \cdot C$ transitions.
- It is immune to $4 \cdot C$ transitions only on the first three bits
- So, how to construct $\overline{G_n^{4C-free}}$ from G_3^{4C}?

for $i = 1$ *to* $n - 3$ **do**

$G_{i+3}^{kC} = G_{i+2}^{kC} + G_3^{kC}((v_{i+1}, v_{i+2}, v_{i+3}) \leftarrow (v_1, v_2, v_3), (w_{i+1}, w_{i+2}, w_{i+3}) \leftarrow (w_1, w_2, w_3))$

end for

$\overline{G_n^{kC-free}} \leftarrow G_n^{kC}$

return $\overline{G_n^{kC-free}}$

Efficient Construction of $G_n^{4C-free}$ ... 2

- Only the final $\overline{G_n^{4C-free}}$ that is constructed using the previous algorithm is utilized for CODEC construction
  - Intermediate ROBDDs for G_i^{4C} $(i < n)$ will possibly have $4 \cdot C$ crosstalk transitions.
- The final $G_n^{kC-free}$ encodes a family of Finite State Machines (FSMs) containing all legal transitions (in an implicit form using ROBDDs).
- Note that the construction of $G_n^{kC-free}$ is similar, details are in the paper.
- From $G_n^{kC-free}$, we can find the effective size m of the bus that can be encoded. This is the second step of our procedure.

Finding Effective kC-free Bus Width from $G_n^{kC-free}$

- If an m-bit ($m < n$) bus can be encoded using the legal transitions in $G_n^{kC-free}$, then there must exist a closed set of vertices $V_c \subseteq B^n$ in the v space of $G_n^{kC-free}(v, w)$ such that:
  - Each source vertex $v_s \in V_c$ has at least 2^m outgoing edges (v_s, w_d) to destination vertices w_d (including the self edge), such that the destination vertex $w_d \in V_c$.
  - The cardinality of V_c is at least 2^m.
- The resulting encoder is memory-based.

Finding Effective kC-free Bus Width from $G_n^{kC-free}$

$test\_encoder(m,\, G_n^{kC-free})$

$find\ out - degree(v_s)\ of\ each\ node\ v_s$, insert $(v_s, out - degree(v_s))$ in V if $out - degree(v_s) \geq 2^m$

$degrees\_changed = 1$

while $degrees\_changed$ **do**

  $degrees\_changed = 0$

  for $each\ v_s \in V$ **do**

    for $each\ w_d$ S.T. $G_n^{kC-free}(v_s, w_d) = 1$ **do**

      if $w_d \notin V$ **then**

        $decrement\ out - degree(v_s)$ in V; $degrees\_changed = 1$

      end if

      if $out - degree(v_s) < 2^m$ **then**

        $V \leftarrow V \setminus v_s; break$

      end if

    end for

  end for

end while

if $|V| \geq 2^m$ **then**

  print(m bit bus **can be encoded** using $G_n^{kC-free}$)

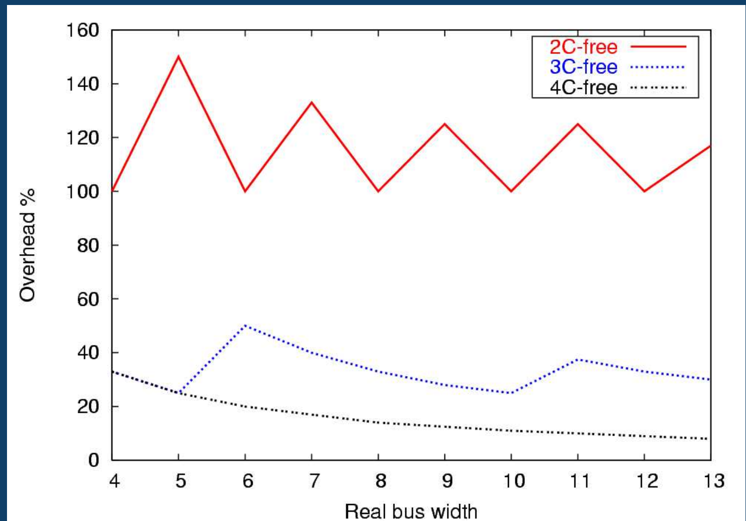
end if

Finding Effective kC-free Bus Width from $G_n^{kC-free}$

- All operations in the algorithm are done using ROBDDs
- We initially call the algorithm with $m = n - 1$
  - If an m bit bus cannot be encoded using $G_n^{kC-free}$, then we decrement m.
  - *We repeat this until we find a value of m such that the m-bit bus can be encoded by $G_n^{kC-free}$.*
- Once we know the effective bus size m, we can construct an FSM for the encoder and decoder. There is significant flexibility in constructing the FSMs.
  - From the vertices in V, we can select a subset V^{FSM} such that $|V^{FSM} = 2^m|$.
  - Once this selection is made, we have further flexibility in assigning the 2^m labels out of each $v \in V^{FSM}$.
- In our current implementation, we make both these selections randomly.

Results

- Implemented in SIS
- Overhead $\frac{n-m}{m}$ shown below.



- Asymptotic overheads for the memory based CODECs are much lower than the memoryless CODEC overheads
  - Overhead for $2 \cdot C$ is 117% compared to 146%, $3 \cdot C$ is 30% compared to 44%, $4 \cdot C$ is 8% compared to 33%
- For wider buses, we recommend that the bus be partitioned into smaller bus segments

Results ... 2

- Standard cell based implementation has delay 280ps in a 0.1μm process.
  - PLA based implementation may reduce this further.
- CODEC area and delay penalty is respectively 15% and 10% larger than memoryless CODEC
- Total area of memory-based CODEC (after accounting for wiring) is 25%, 10% and 20% lower than the memoryless CODECs (for 2C, 3C and 4C free solutions).
- The bus operates faster since delay variation due to cross-talk is eliminated.
  - Since $r = \frac{C_x}{C_{sub}}$ is large, eliminating 4C crosstalk results in a roughly 25% speedup. Eliminating 3C and 2C crosstalk each result in an additional 25% speedup.
- Also, delay overhead can be hidden in pipelined system

Conclusions and Future Work

- In DSM technologies, cross-coupling capacitances for two adjacent wires are high compared to self capacitances.
  - This leads to delay variation and possible loss of signal integrity
- We described memory-based CODECs to eliminate 4C, 3C and 2C cross-talk in buses.
- Formulation is general and handles all types of crosstalk
  - ROBDD based implicit construction of all legal vector transitions.
  - Analysis of the resulting ROBDD yields the effective bus width m for a physical bus width n ($m < n$).
- Bus overhead for $2 \cdot C$ is 117% compared to 146%, $3 \cdot C$ is 30% compared to 44%, $4 \cdot C$ is 8% compared to 33%
- Memory-based CODEC delay about 10% more than memoryless. But delay can be hidden in pipelined systems.