# DATE 2006

**Session 5B: Timing and Noise Analysis**

# Bus Stuttering : An Encoding Technique To Reduce Inductive Noise In Off-Chip Data Transmission


**Authors:** **Brock J. LaMeres,** **Agilent Technologies**
**Sunil P. Khatri,** **Texas A&M University**

# Agenda

- **Problem Motivation**
- **Our Solution**
- **Experimental Results**

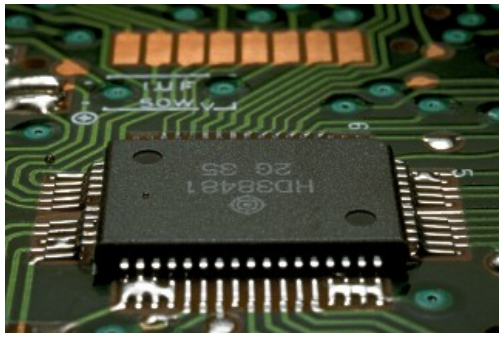
# Why is IC Packaging Important?

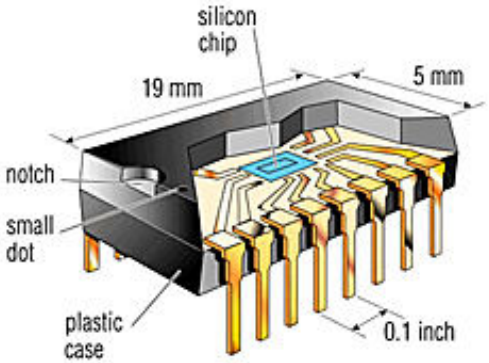
- **All Electronic Circuitry Resides in a Package**

  **- The package serves many purposes:**

  **1) Protection of devices**
  **2) Density Translation**
  **3) Thermal Dissipation**
  **4) Manufacturing Standardization**

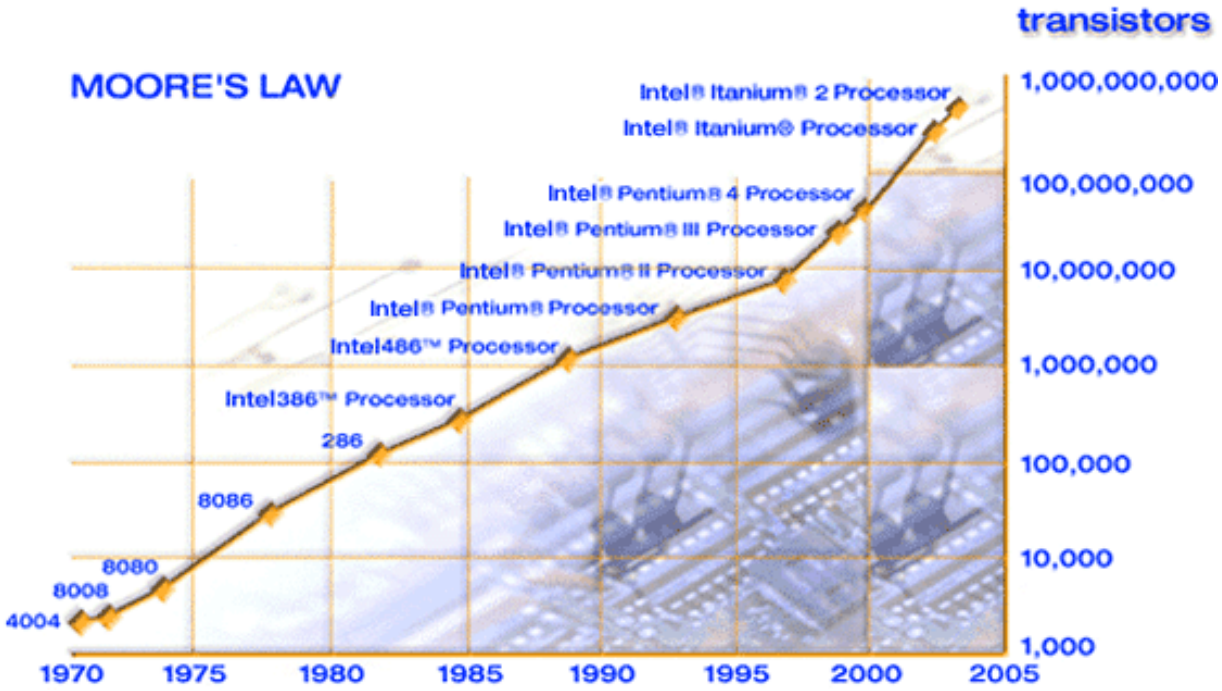
- **Packaging Limits System Performance**





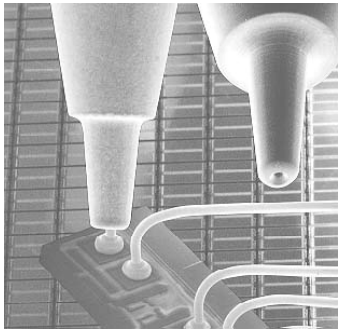
# Why is packaging limiting performance?

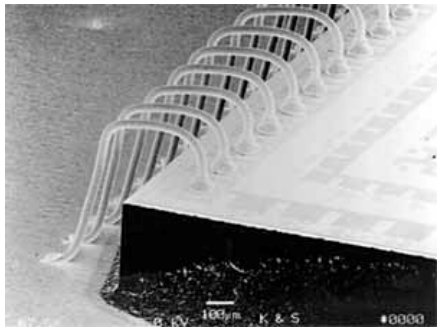
- **IC Design/Fabrication is Outpacing Package Technology**
  - **We're seeing exponential increase in IC transistor performance**
  - **>1.3 Billion transistors on 1 die [Fall IDF-05]**

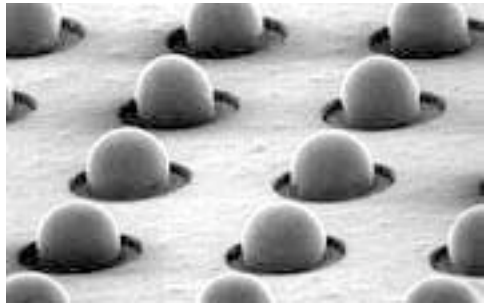


# Why is packaging limiting performance?

- **Packages Have Been Designed for Mechanical Performance**
  - **Electrical performance was not primary consideration**
  - **IC's limited electrical performance**
  - **Package performance was not the bottleneck**

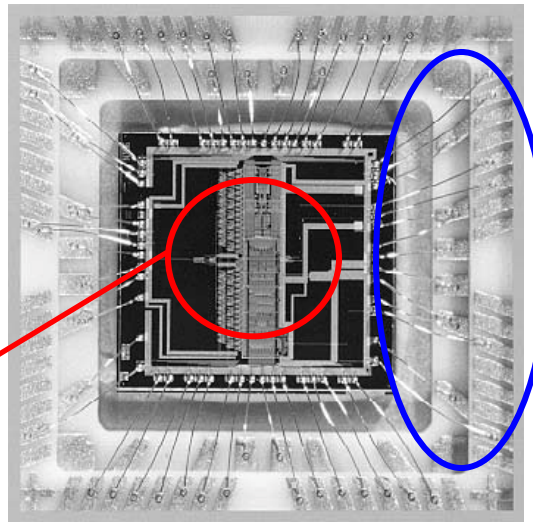






# Why is packaging limiting performance?

- **VLSI Performance Exceeds Package Performance**
  - **Packages optimized for mechanical reliability, but still used due to cost**
  - **IC performance far exceeds package performance**



**On-Chip**

- **$f_{IC}$ > 4GHz**
- **large signal counts**
- **exponential scaling**

**Package**

- **$f_{pkg}$ < 2GHz**
- **limited signal counts**
- **linear scaling**

# Why is packaging limiting performance?

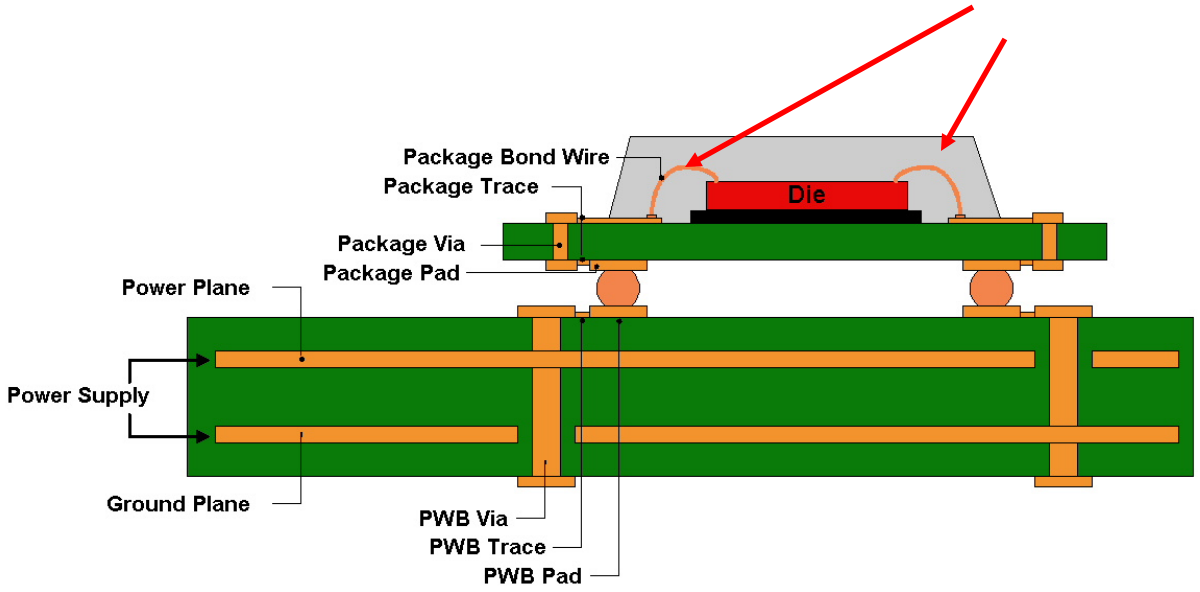
- **Package Interconnect Contains Parasitic Inductance**

- Long interconnect paths

- Large return loops

- $L = \frac{\Phi}{Area}$

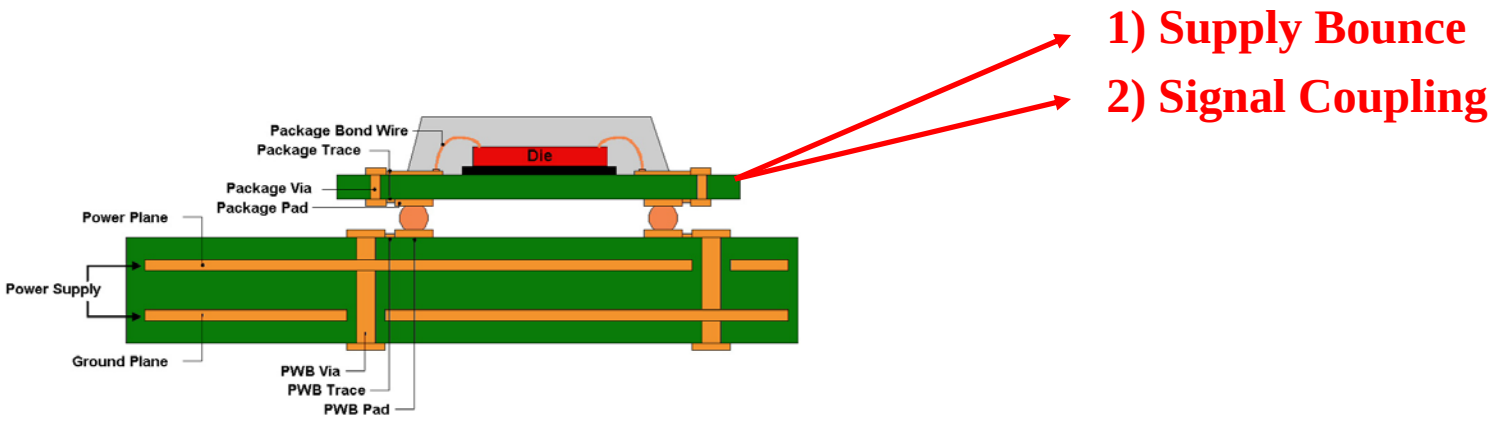
**Wire Bond Inductance (~2.8nH)**



# Why is packaging limiting performance?

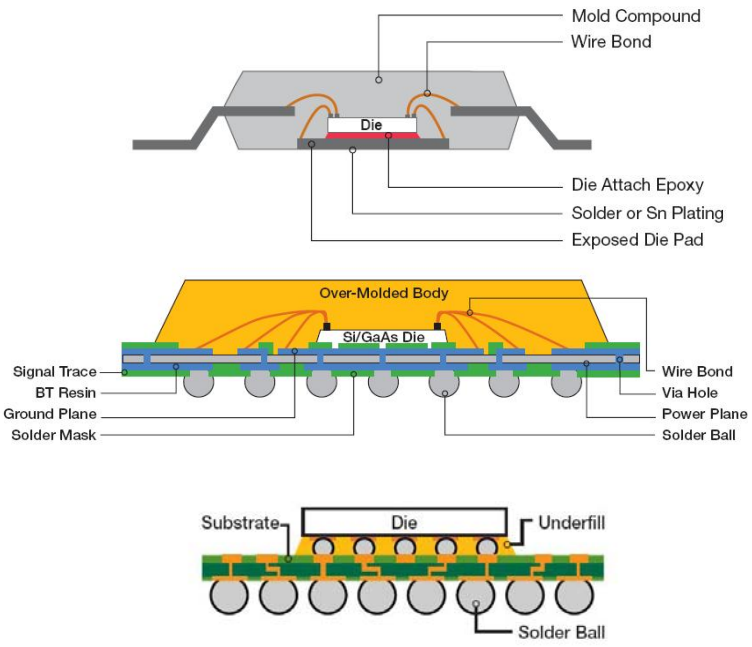
- **Package Parasitics Limit Performance**

  - **Excess inductance causes package noise**
  - **Noise limits how fast the package can transmit date**



# Why is packaging limiting performance?

- **Aggressive Package Design Helps, but is expensive…**
  - **95% of VLSI design-starts are wire bonded**
  - **Goal: Extend the life of current packages**




**QFP – Wire Bond : 4.5nH → $0.22 / pin**

**BGA – Wire Bond : 3.7nH → $0.34 / pin ***

**BGA – Flip-Chip : 1.2nH → $0.63 / pin**

# Our Solution

## "Encode Off-Chip Data to Avoid Inductive Cross-talk"

- **Avoid the following cases:**

1) **Excessive switching in the same direction** **= reduce ground/power bounce**

2) **Excessive X-talk on a signal when switching** **= reduce edge degradation**

3) **Excessive X-talk on signal when static** **= reduce glitching**

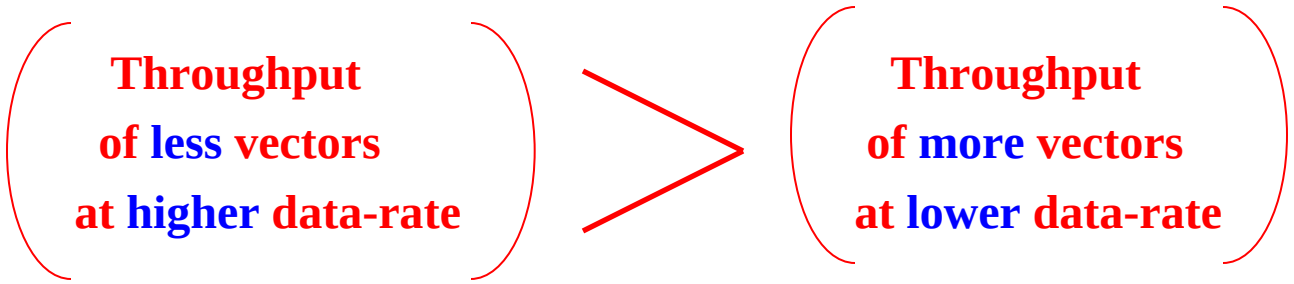
# Our Solution

- **This results in:**

**1) A subset of vectors is transmitted that avoids inductive X-talk.**

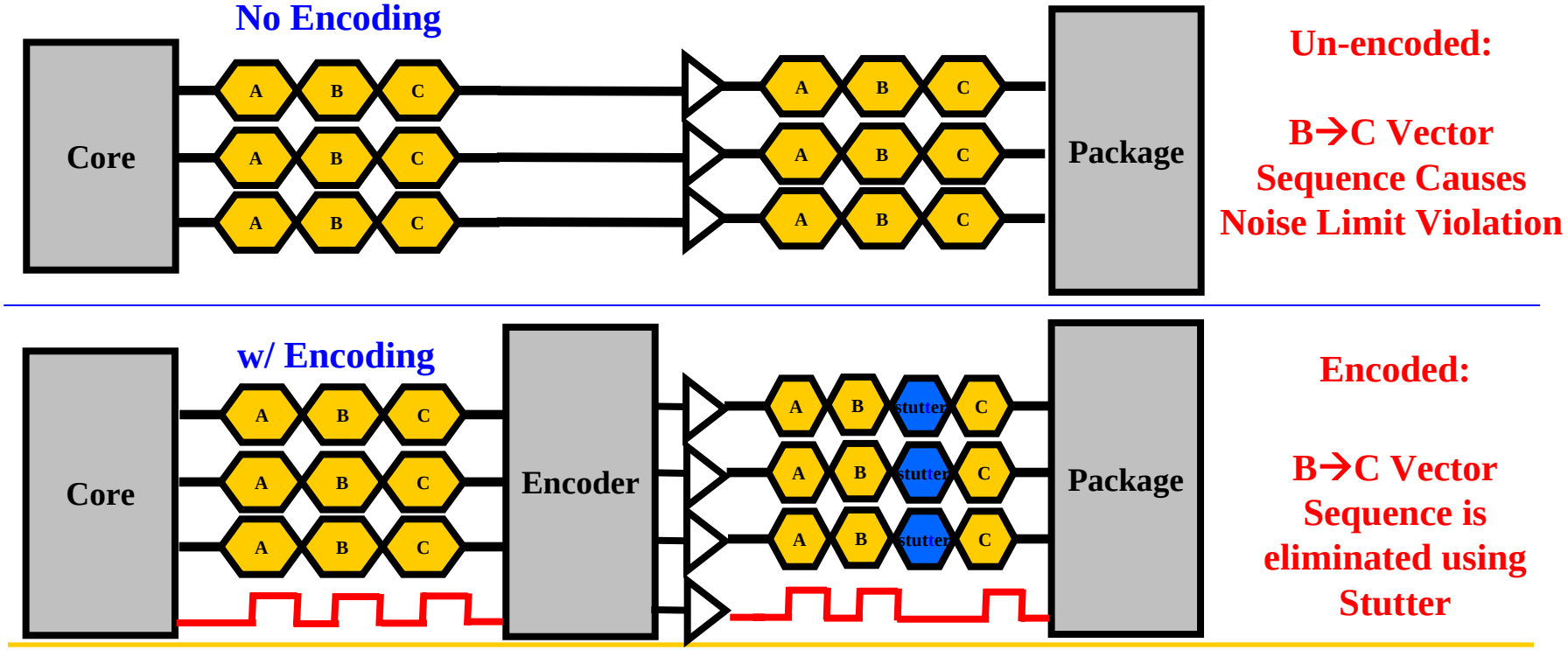
**2) The off-chip bus can now be ran at a higher data rate.**

**3) The subset of vectors running faster can achieve a higher throughput over the original set of vectors running slower (including overhead).**



# Bus Stuttering CODEC

- **Intermediate States are Inserted Between Noise Causing Transitions**
  - ***Stutter*** **states limit the number of simultaneously switching signals**
  - **The source synchronous clock is gated during stutter state**



# Bus Stuttering CODEC – Noise Sources

- **Simultaneous Switching Noise**

**Supply Bounce**

- **Induced Self Voltage**

$$V_{self} = L \cdot \sum_{i}^{n} \left( \frac{di_i}{dt} \right)$$

**Glitching**

- **Coupling onto *Non-Switching* Signals**

**Edge Degradation**

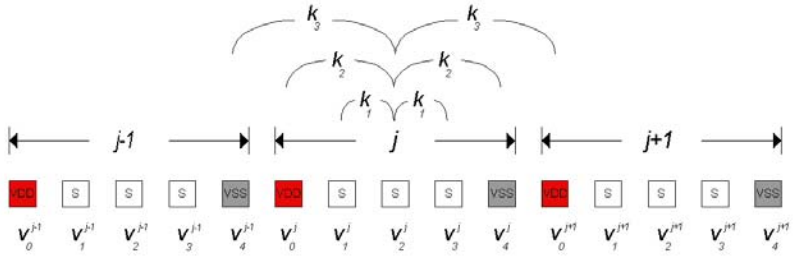
- **Coupling onto *Switching* Signals**
- **Data Dependent Delay**

$$V_{couple} = \sum_{1}^{k} M_{1k} \cdot \left( \frac{di_k}{dt} \right)$$

# Bus Stuttering CODEC - Constraints

- **For Each Possible Noise Source on the Bus, a Constraint is written:**

1) $v_0^j = VDD \rightarrow -P_{bnc} \leq (L/2)\cdot(\#\text{ of } v_i^j \text{ pins} = 1) \leq Pbnc$
2) $v_1^j = 1 \rightarrow k1\cdot(v_2^j) + k2\cdot(v_3^j) \geq P_1$
3) $v_1^j = -1 \rightarrow k1\cdot(v_2^j) + k2\cdot(v_3^j) \leq P_{-1}$
4) $v_1^j = 0 \rightarrow -P_0 \leq k1\cdot(v_2^j) + k2\cdot(v_3^j) \leq P_0$



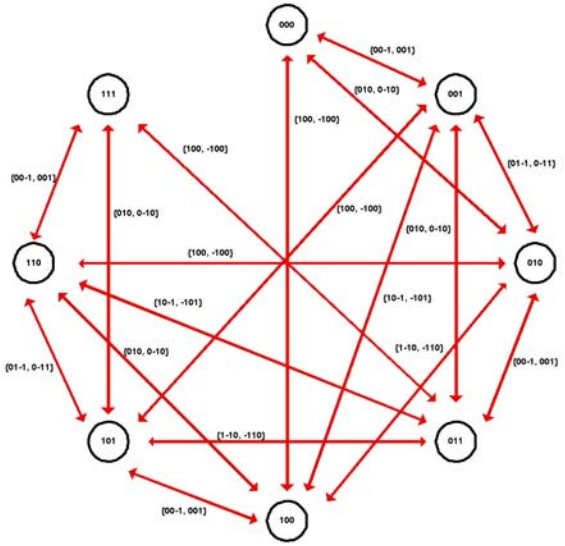

- **Each Constraint is Evaluated to Find Illegal Transitions:**

$v_1^j = 1$ = rising
$v_1^j = 0$ = static
$v_1^j = -1$ = falling

| $v_1^j$ | $v_2^j$ | $v_3^j$ | |
|---|---|---|---|
| 1 | 0 | 1 | |
| 1 | -1 | 0 | |
| 1 | -1 | -1 | |
| ~~1~~ | ~~-1~~ | ~~1~~ | violates user-defined "*glitch*" parameter |
| 1 | 1 | 0 | |
| 1 | 1 | -1 | |
| ~~1~~ | ~~1~~ | ~~1~~ | violates user-defined "*supply*" bounce parameter |

# Bus Stuttering CODEC - Algorithm

- **Constraints are Evaluated and a Legal Directed Graph is Created**




***G***

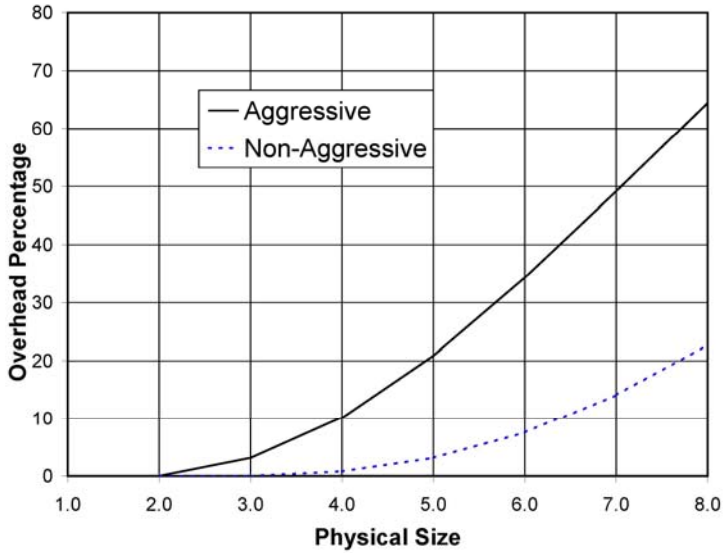
- **Directed Graph is Used to Map Transitions Between any Two Vectors**
  - **A transition path (which may include stutters) exists between any two vectors if:**
    - **There exists at least two outgoing edges for each vector $v_s \in G$ (including self-edge)**
    - **There exists at least two incoming edges for each vector $v_d \in G$ (including self-edge)**

# Bus Stuttering CODEC - Construction

- **Multiple Stutter States can be used**
  - **between 0 and $2^{(W_{bus}-1)}$ stutters can be inserted between any two vectors**
  - **experimental results show that for segments up to 8 bits, more than 3 stutters is rare**

- **Overhead**
  - **Overhead increases as segments sizes increase**
  - **Still useful since segments greater than 8 bits are rarely used (SPG=8:1:1)**

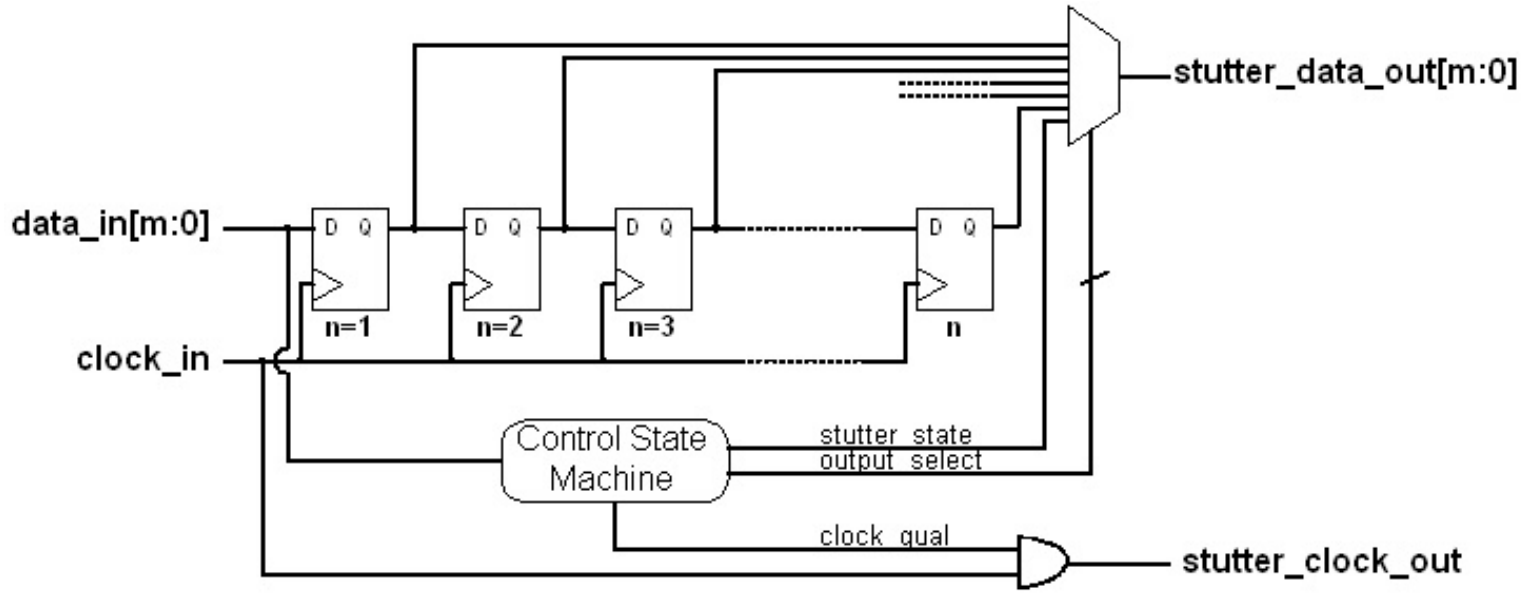
$$Overhead = \frac{\sum_{k=1}^{2^{(W_{bus}-1)}} (\#\_Trans\_\mathrm{Re}quiring\_k\_stutters)\cdot k}{2^{(2\cdot W_{bus})}}$$



# Bus Stuttering CODEC – Physical Results

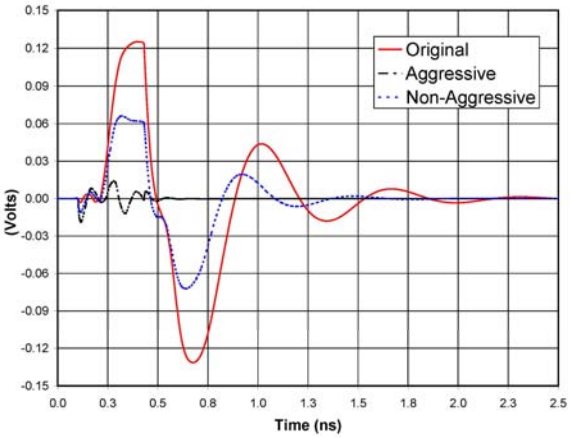
- **Circuit Implementation**

**- 32 pipeline stages used**
**- pipeline reset after 32 idle states (similar to SRIO, HT, and PCI Express)**
**- protocol inherently handles pipeline overflow**

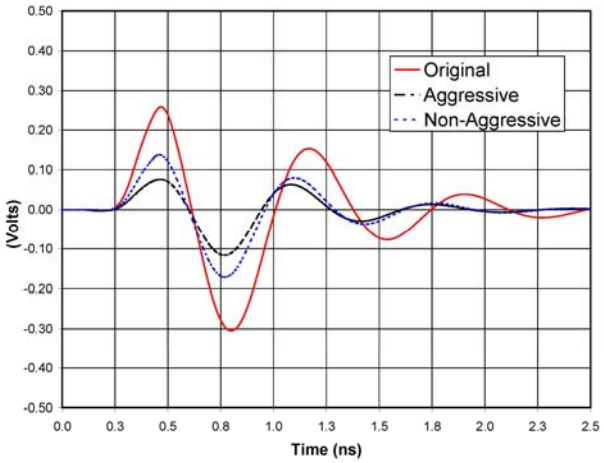


# Bus Stuttering CODEC – Physical Results

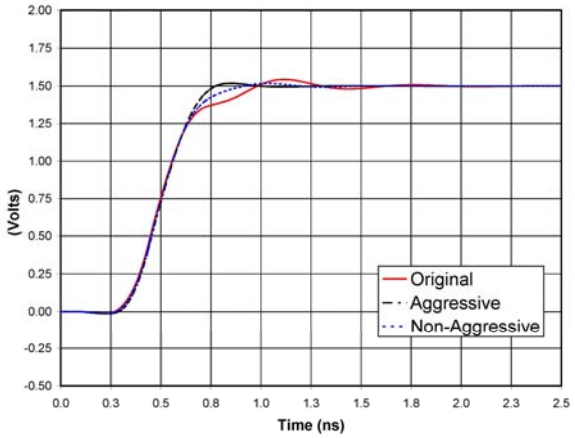
- **SPICE Simulations**
  - **3 bit segment (SPG=3:1:1)**
  - **fixed di/dt**
  - **Maximum noise reduced by limiting simultaneously switching signals**




**Ground Bounce**




**Glitching**



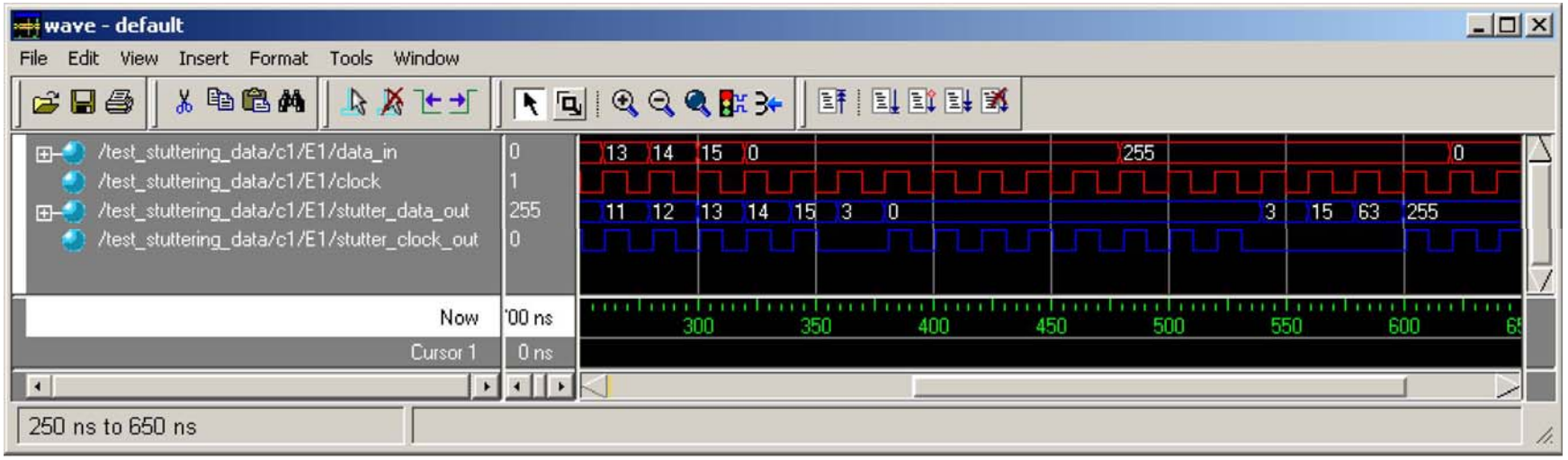

**Edge Degradation**

# Bus Stuttering CODEC – Physical Results

- **TSMC 0.13um Synthesis Results**

**- RTL design, synthesized and mapped**
**- Segment sizes 2 → 8 implemented**
**- Logic, delay, and area evaluated**

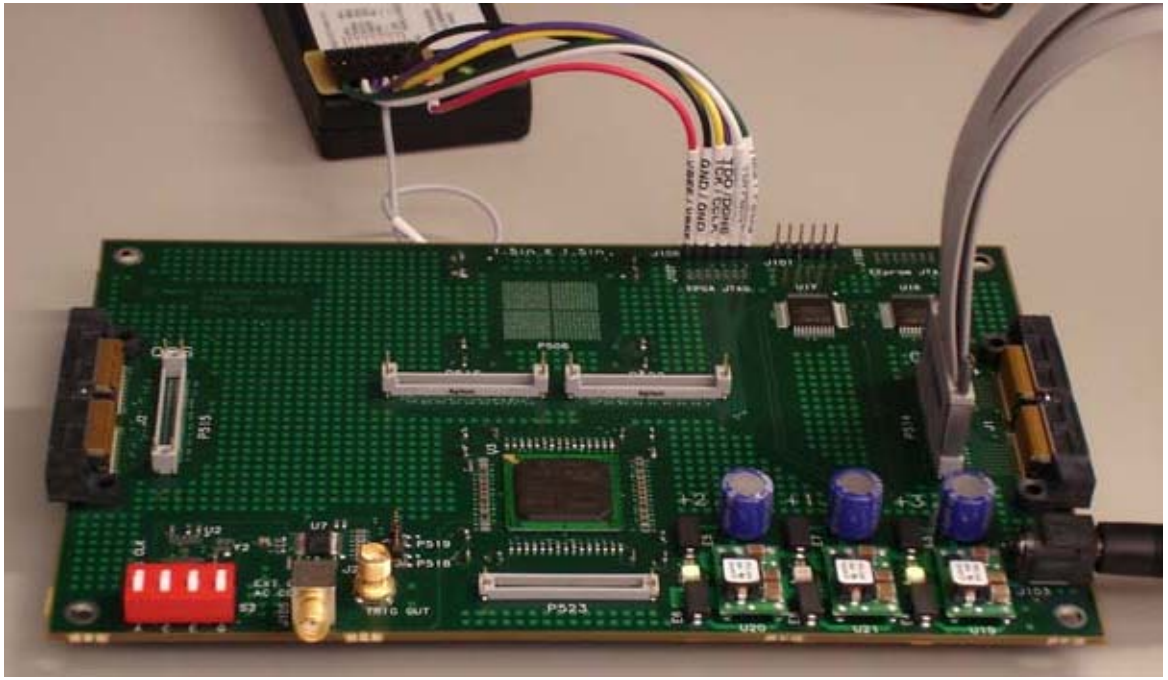
| | Bus Size | Noise Limit | |
|---|---|---|---|
| | - | 5% (aggressive) | 10% (non-aggressive) |
| Delay ($ns$) | 4 | 2.02 | 1.99 |
| | 6 | 2.42 | 2.38 |
| | 8 | 2.85 | 2.79 |
| Area ($um^2$) | 4 | 311k | 310k |
| | 6 | 362k | 345k |
| | 8 | 382k | 368k |



# Bus Stuttering CODEC – Physical Results

- **Xilinx FPGA, 0.35um Implementation Results**

**- RTL design implemented**
**- Xilinx, VirtexIIPro, FPGA**



# Bus Stuttering CODEC – Physical Results

- **Xilinx FPGA, 0.35um Implementation Results**

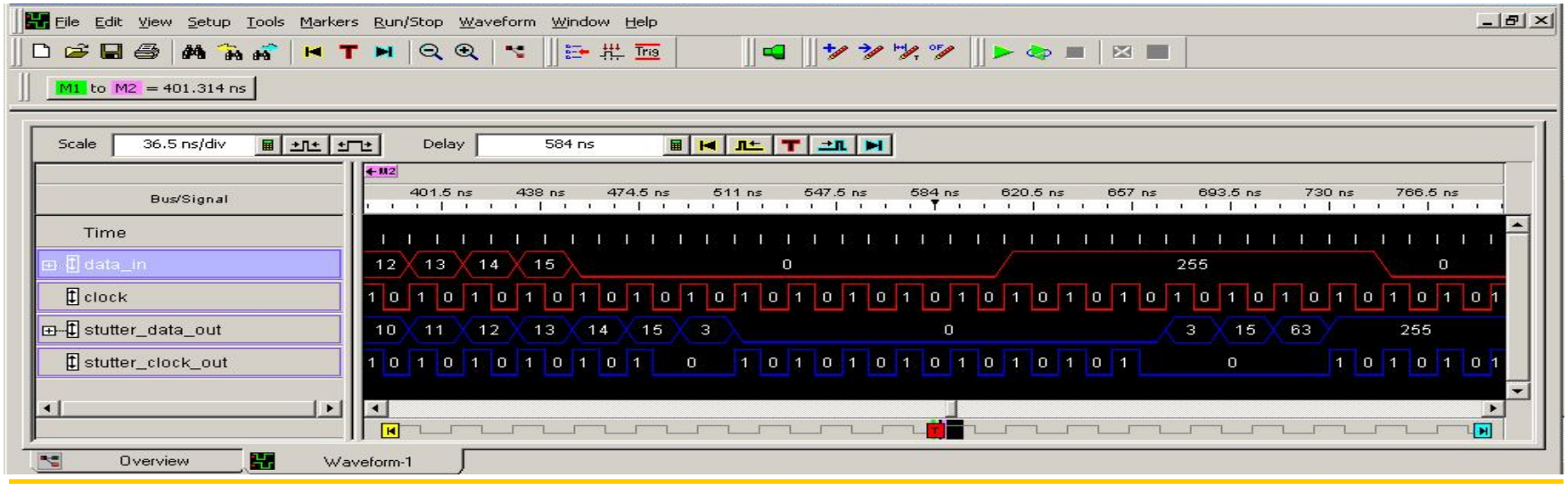
**- RTL design, implemented**

**- Logic operation verified**

**- Noise Reduced from 16% to 4%**

**(4 bit, SPG=4:1:1)**

| | Bus Size | Noise Limit |
|---|---|---|
| | - | 5% (aggressive) & 10% (non-aggressive) |
| Delay ($ns$) | 4 | 4.78 |
| | 6 | 5.29 |
| | 8 | 5.89 |
| FPGA Usage | 4 | $< 1\%$ |
| | 6 | $< 1\%$ |
| | 8 | $< 1.5\%$ |



# Conclusion

- **Packaging Performance is the Largest System Bottleneck**
- **Stutter Encoding Avoids Worst-Case Noise Patterns**
- **Performance Improved Even After Considering Encoding Overhead**