PID (Partial Inversion Data): an M-of-N Level-Encoded Transition Signaling Protocol for Asynchronous Global Communication

Marco Cannizzaro and Luciano Lavagno
Dipartimento di Elettronica
Politecnico di Torino, Turin, Italy
Email: {marco.cannizzaro, luciano.lavagno}@polito.it

ASYNC12
May 7-9, 2012
Asynchronous data communication

- **Delay-Insensitive (DI) Codes** (unordered codes)
  - Provide timing-robust communication:
    - Tolerant to arbitrary bit skew, P/V/T variability.

- **NRZ (2-phase) codes**
  - Potential better throughput and less power than RZ (4-phase)
    - no 'spacer code' is required between any pair of valid codewords.
Level Vs. Transition Encoded

- **Level Encoded**
  - given a codeword, the encoded data can be directly extracted by using a combinational logic function
    - any codeword corresponds to one and only one symbol

- **Transition Encoded**
  - the encoder and decoder need to store at least one past codeword.

<table>
<thead>
<tr>
<th>Level Encoded</th>
<th>Transition Encoded</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-of-2 LEDR</td>
<td>M-of-N Transition Encoded</td>
</tr>
<tr>
<td>1-of-N LETS</td>
<td></td>
</tr>
</tbody>
</table>
DI 2-phase Background: 1-of-2 LEDR

- **2 wires per bit**
- **Level-encoding**
  - Data rail: holds actual data value
  - Parity rail: holds parity value
- **Alternating-phase protocol**
  - Encoding parity alternates between odd and even

<table>
<thead>
<tr>
<th>Phase</th>
<th>Bit value</th>
<th>Data Rail</th>
<th>Parity Rail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Even</td>
<td>0 0</td>
<td>1 1</td>
<td></td>
</tr>
<tr>
<td>Odd</td>
<td>0 1</td>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>
DI 2-phase Background: 1-of-4 LETS

- 4 wires per 2 data bits
- Alternating-phase protocol
  - 2 codewords for each symbol in each phase

<table>
<thead>
<tr>
<th>phase</th>
<th>symbol</th>
<th>codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODD</td>
<td>S0</td>
<td>1000/0111</td>
</tr>
<tr>
<td></td>
<td>S1</td>
<td>0100/1011</td>
</tr>
<tr>
<td></td>
<td>S2</td>
<td>0010/1101</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>0001/1110</td>
</tr>
<tr>
<td>EVEN</td>
<td>S0</td>
<td>1111/0000</td>
</tr>
<tr>
<td></td>
<td>S1</td>
<td>0011/1100</td>
</tr>
<tr>
<td></td>
<td>S2</td>
<td>0101/1010</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>0110/1001</td>
</tr>
</tbody>
</table>
DI 2-phase Background: M-of-N Transition Encoded

- \( m \): number of transitions per transaction
- \( n \): number of bits of the codeword
- \( k \): max. number of bits encoded

→ Any combination of \( m \) transitions in the codeword encodes exactly one symbol

\[
\# symbols = \frac{n!}{m!(n-m)!}
\]

\[
k = \text{floor}(\log_2 \# symbols)
\]
Comparison: Power Vs. Coding Density

1-of-2 LEDR

1-of-4 LETS

Power metric (m/k)

Coding density (n/k)
Comparison: Hardware (decoder) cost

![Graph showing the comparison of hardware cost for different values of k and storage elements/literals. The graph includes lines for 1-of-n, 2-of-n, 3-of-n, and 4-of-n configurations.](image)
## Contribution

<table>
<thead>
<tr>
<th>Evaluation metric</th>
<th>M-of-N Transition Encoded</th>
<th>1-of-2 LEDR 1-of-N LETS</th>
<th>M-of-N PID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding efficiency</td>
<td>good</td>
<td>bad</td>
<td>good</td>
</tr>
<tr>
<td>Power consumption</td>
<td>good</td>
<td>bad</td>
<td>good</td>
</tr>
<tr>
<td>Hardware cost</td>
<td>bad</td>
<td>good</td>
<td>good</td>
</tr>
</tbody>
</table>
PID codeword

The **Parity** field (P) always ensures M transitions in the codeword.

The **Inversion** field (I) carries two pieces of information:
1. whether the data field (D) is inverted or not and
2. the value of k encoded bits which are not in the data field (D).

The **Data** field (D) carries the value of the first d bits of the encoded data (inverted or not according to the inversion field).
PID: the idea

- By optionally inverting all the bits of the data field (D) we reduce the maximum number of transitions to the floor of $d/2$ for any transaction.

- The inversion field (I) (which always has 0 or 1 transitions) is composed of sub-fields $I_{<x>$ and each of them corresponds to one encoded data bit. This increases the number of encoded data bits without increasing the number of transitions $M$ in the codeword (i.e. improves the power efficiency of the code).
Example: the 2-of-7 PID code

Last codeword: 00.11.001
- Encoded data: 0110

Next data to be encoded: 1101
- Next codeword: 00.10.101
### M-of-N PID codes

<table>
<thead>
<tr>
<th># data bits</th>
<th>d=1</th>
<th>d=2</th>
<th>d=3</th>
<th>d=4</th>
<th>d=5</th>
<th>d=6</th>
<th>d=7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-of-2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1-of-4</td>
<td>2-of-4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1-of-8</td>
<td>2-of-6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1-of-16</td>
<td>2-of-10</td>
<td>2-of-7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>1-of-32</td>
<td>2-of-18</td>
<td>2-of-11</td>
<td>3-of-9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>1-of-64</td>
<td>2-of-34</td>
<td>2-of-19</td>
<td>3-of-13</td>
<td>3-of-10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>1-of-128</td>
<td>2-of-66</td>
<td>2-of-35</td>
<td>3-of-21</td>
<td>3-of-14</td>
<td>4-of-12</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>1-of-256</td>
<td>2-of-130</td>
<td>2-of-67</td>
<td>3-of-37</td>
<td>3-of-22</td>
<td>4-of-16</td>
<td>4-of-13</td>
</tr>
<tr>
<td>9</td>
<td>1-of-512</td>
<td>2-of-258</td>
<td>2-of-131</td>
<td>3-of-69</td>
<td>3-of-38</td>
<td>4-of-24</td>
<td>4-of-17</td>
</tr>
</tbody>
</table>

Codes in grey are not Pareto-optimal and can be replaced with other codes which have better coding efficiency.
The hardware for a particular M-of-N1 code can be reused for any M-of-N2 code where N2 < N1, if the extra inputs in the inversion field are not used.
M-of-N PID Encoding algorithm

**Step 1:** The Hamming distance is computed between the first d bits of the new data and the data field (D) of the previous codeword.

**Step 2:** Each one of the other k data bits is compared to the corresponding bit of the previous data and the index of the inversion sub-field that must have a transition is selected.

**Step 3.1:** If one inversion field must be flipped, the algorithm:
   1. Checks whether the data will be inverted in the new data field or not.
   2. Looks for and flips the bit within the inversion sub-field.

**Step 3.2:** If none inversion field must be flipped, the algorithm only checks whether the data will be inverted in the new data field or not.

**Step 4:** Data is inverted or not to generate the data field (D).

**Step 5:** Between 0 and M bits of the parity field are flipped in order to always have M transitions in the codeword.
Results: Power and Coding efficiency

Power and Coding efficiency comparison between Delay-Insensitive NRZ codes
Results: Area overhead (due to decoder)

Area overhead comparison (due to decoder) between Delay-Insensitive NRZ codes

- Transition-encoded
- LETS
- PID
Results: Delay overhead (due to decoder)
Conclusion: the PID code…

- …is a **Delay-Insensitive M-of-N** protocol, where only M wires flip for each data transaction.
- …is a **NRZ** code, having significant power and throughput benefits with respect to Return-to-Zero (RZ) codes.
- …is **Level-encoded**, meaning that the decoding process simply uses the values of the codeword.
- …has a **generic encoding algorithm and decoder implementation** (that works for any M-of-N PID code).
Conclusion: PID results

<table>
<thead>
<tr>
<th>PID comparison</th>
<th>Coding Efficiency</th>
<th>HW overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-of-N LETS</td>
<td>better/equal</td>
<td>equal (but LETS has no generalization)</td>
</tr>
<tr>
<td>M-of-N Transition Encoded</td>
<td>worse/equal</td>
<td>better</td>
</tr>
</tbody>
</table>

In particular, the **2-of-7 PID code**, which encodes 4 data bits in 7 codeword wires, *Pareto dominates all other DI NRZ codes.*
Thank you for your attention!

Marco Cannizzaro
Dipartimento di Elettronica
Politecnico di Torino, Turin, Italy
Email: marco.cannizzaro@polito.it