### Algorithms for Threshold Dispersion Minimization of the CHIPIX65 Asynchronous Front-End

**M. Sonzogni, D. De Canio, L. Gaioni, M. Manghisoni, L. Ratti, V. Re, G. Traversi**

**University of Bergamo**, INFN Pavia & University of Pavia

**Abstract**

This front-end is designed according to RD53 requirements: capability of operating at a minimum threshold lower than 1000 e⁻, power consumption not exceeding 5 μW in a silicon area close to 1000 μm² and a mean equivalent noise charge (ENC) lower than 125 e⁻. Some of its features:

- Single amplification stage for minimum power dissipation
- Charge sensitive amplifier with a gain stage featuring a folded cascode architecture, with two local feedback networks designed to increase the output small signal resistance
- Kronmacher feedback, providing a constant current discharge of the feedback capacitor \( C_{fb} \), to comply with the expected large increase in the detector leakage current after irradiation
- High speed, low power current comparator
- Pixel threshold trimming DAC, based on a 4-bit binary weighted architecture, generating the current \( I_{tr} \) which adds to the current \( I_{ref} \) (equal for each pixel in the matrix) in order to obtain the same threshold in all matrix pixels (threshold dispersion minimization)
- Relatively small time-over-threshold (ToT) clock: \( \sim 400 \) ns
- 5-bit, dual edge ToT counter – 400 ns maximum ToT
- In-pixel threshold trimming DAC, based on a 4-bit binary weighted architecture, generating the current \( I_{tr} \) which adds to the current \( I_{ref} \) (equal for each pixel in the matrix) in order to obtain the same threshold in all matrix pixels (threshold dispersion minimization)
- Cons: Different couples
- Complexity: \( O(\log_{2} n_{p}) \)

### ASYNOCHRONOUS ANALOG FRONT-END

**Introduction**

In this section 4 algorithms for threshold minimization are described: 

- Global: all matrix pixels must have a threshold as close as possible to a threshold selected by the user. In the figure, the error function curve used to fit the efficiency data is reported: each pixel threshold will be assigned to the value of \( s_{min} \) which minimizes the function

\[
\text{Error Function} = \sum_{i=1}^{n_{p}} \left| \frac{I_{hit}}{I_{ref}} - 1 \right|
\]

- Local: for each pixel the DAC code which sets the threshold as close as possible to the threshold selected by the user will be assigned. In the figure, the error function curve used to fit the efficiency data is reported: each pixel threshold will be assigned to the value of \( s_{min} \) which minimizes the function

\[
\text{Error Function} = \sum_{i=1}^{n_{p}} \left| \frac{I_{hit}}{I_{ref}} - 1 \right|
\]

**Results & Conclusions**

Different measurements, mainly concerned with threshold dispersion minimization, have been performed on three chips. One of these has been tested after exposure to total ionizing doses (TID) up to 630 Mrad (SiO₂). Threshold dispersion at 600 e⁻, obtained with these four algorithms, is reported in the table below (expressed in electrons). The pictures on the left represent Chip 1 threshold distribution at high threshold before and after tuning (with Algorithm 2).

<table>
<thead>
<tr>
<th>Chip</th>
<th>( s_{min} )</th>
<th>( \delta q_{CD} )</th>
<th>( \delta k )</th>
<th>( \delta k )</th>
<th>( \delta V_{REF} )</th>
<th>( \delta V_{REF} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip 1</td>
<td>519</td>
<td>514</td>
<td>519</td>
<td>514</td>
<td>519</td>
<td>514</td>
</tr>
<tr>
<td>Chip 2</td>
<td>541</td>
<td>537</td>
<td>541</td>
<td>537</td>
<td>541</td>
<td>541</td>
</tr>
<tr>
<td>Infra Chips</td>
<td>519</td>
<td>514</td>
<td>519</td>
<td>514</td>
<td>519</td>
<td>514</td>
</tr>
</tbody>
</table>

### 3 - Root To Leaf Binary Tree Scan

This algorithm is based on the binary search tree shown in figure below. Starting from a DAC code equal to \( n_{comb} \), a fixed charge \( s_{th} \) is injected in each pixel. At each step, the number of comparator hits is counted as an array element; this number defines the direction of the next branch, either \( up \) or \( down \). Finally, the DAC code associated to a hit occupancy closer to \( 50\% \) is selected.

**Pros:**
- Fast algorithm; if all pixels work correctly, results like in algorithm 1 or 2 are obtained
- Cons: More difficult to implement; faulty pixels are hard to be detected

### 4 - Bounded Binary Tree Scan

This algorithm is based on a binary search tree like algorithm 3. The user defines two values: \( s_{min} \) and \( s_{max} \). Like in the last algorithm, if first and last DAC code are selected, a fixed charge \( s_{th} \) is injected in each pixel. At each step, if the comparator hits are more than \( s_{max} \), the DAC code has to be greater (up in the binary tree), if they are less than \( s_{min} \), smaller (down in the binary tree), while if they are between \( s_{min} \) and \( s_{max} \) that DAC code is selected and the scan of next pixel starts.

**Pros:**
- Fast algorithm
- Cons: Different couples \( s_{min} \) and \( s_{max} \) have to be tested in order to find the one which minimize the threshold dispersion (if the range is too wide tree leaf will never be reached, while if it's too narrow the algorithm will stop in the internal nodes, as shown in figures above), the obtained results could be not optimal.

**Results & Conclusions**

Different measurements, mainly concerned with threshold dispersion minimization, have been performed on three chips. One of these has been tested after exposure to total ionizing doses (TID) up to 630 Mrad (SiO₂). Threshold dispersion at 600 e⁻, obtained with these four algorithms, is reported in the table below (expressed in electrons). The pictures on the left represent Chip 1 threshold distribution at high threshold before and after tuning (with Algorithm 2).

<table>
<thead>
<tr>
<th>Chip</th>
<th>( s_{min} )</th>
<th>( \delta q_{CD} )</th>
<th>( \delta k )</th>
<th>( \delta k )</th>
<th>( \delta V_{REF} )</th>
<th>( \delta V_{REF} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip 1</td>
<td>519</td>
<td>514</td>
<td>519</td>
<td>514</td>
<td>519</td>
<td>514</td>
</tr>
<tr>
<td>Chip 2</td>
<td>541</td>
<td>537</td>
<td>541</td>
<td>537</td>
<td>541</td>
<td>541</td>
</tr>
<tr>
<td>Infra Chips</td>
<td>519</td>
<td>514</td>
<td>519</td>
<td>514</td>
<td>519</td>
<td>514</td>
</tr>
</tbody>
</table>