Bitmarker Design v3.1 rev 1

P. D. van den Berg, MSc.
University of Twente
Enschede, The Netherlands
p.d.bergvanden@utwente.nl

I. Definitions

This design is meant as a substrate for nanowire deposition, providing special markers to accurately locate and align subsequent designs. It contains 3 different levels of detail, for optical inspection as well as accurate alignment under an electron microscope. We define the following concepts:

- **Marker**
  Any square or composition of squares which can be used for either EBL or design alignment.
- **EBL marker**
  A 10x10 um square for EBL write field alignment.
- **Bitmarker**
  A 3x3 or 4x4 region of markers for design alignment, encoding an (x, y) coordinate.
- **Macro marker**
  A 4x4 bitmarker, 1x1 um per square. This may encode $(0, 0)$ up to $(63, 63)$.
- **Micro marker**
  A 3x3 bitmarker, 100x100 nm per square. This may encode $(0, 0)$ up to $(7, 7)$.
- **Field**
  A region containing both macro and micro markers.

The coordinate system has $(0, 0)$ in the bottom left, increasing both x and y towards the top right. Numbers in a different base are indicated by a subscript, i.e. $8 = 8_{10} = 1000_2$.

II. Field Generation

A Python script generates a single field out of 4 types, stored as a *KLayout* file. These fields can be of type **A**, **B**, **C**, or **D**. Field A can be seen in Figure 1. The field type is encoded using both optically readable text as well as a binary flag next to the bitmarkers. Inside a field, there is a grid of **32x32 macro markers**. These are spaced **50 um** apart. For each macro marker, there is a subfield of **8x8 micro markers**, spaced **5 um** apart. Every 3rd subfield is replaced by either an EBL marker or text indicating the field type. Thus, we obtain a lattice of **EBL markers**, forming a rectangular grid of **300x300 um**. These markers appear slightly off-center, as their coordinates are (optionally) aligned to multiples of 10 um for ease of use.

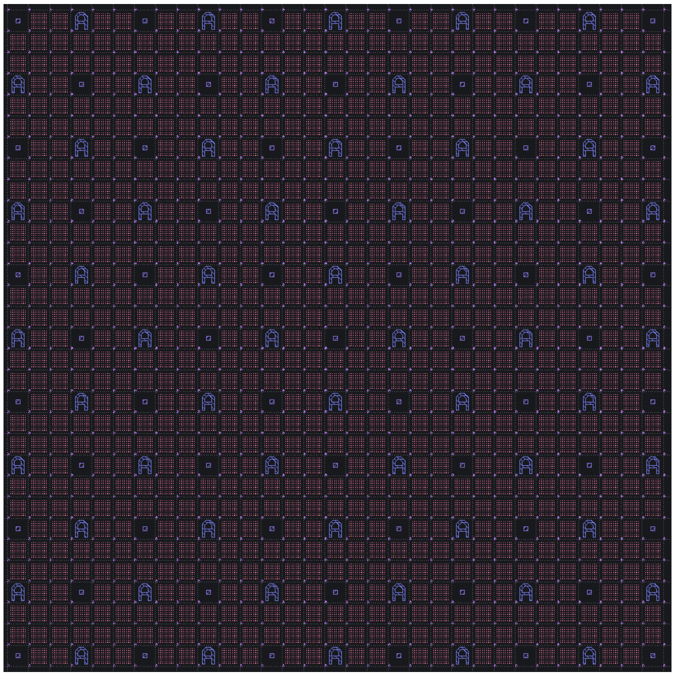

Figure 1: Field A, containing EBL markers and field labels, macro and micro markers.

III. Binary Encoding

The bitmarkers encode an (x, y) pair in binary, i.e. base 2. A binary number consists only of 0 and 1, the “bits”. A 1 is encoded by a square while 0 is encoded as a gap. A bitmarker **always contains a diagonal**, aligned from the bottom left to the top right. This defines the orientation. Around the bitmarker, we encode the field using 3 squares. An example macro marker can be seen in Figure 2.

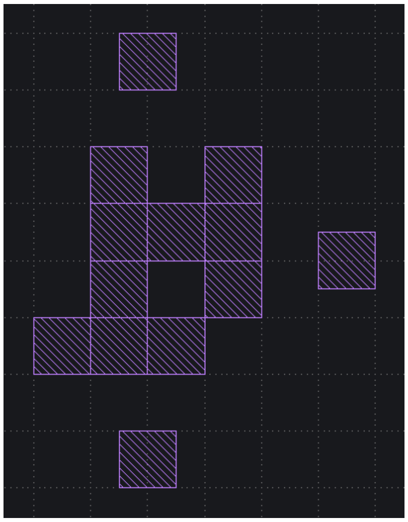

Figure 2: A macro marker. It encodes $(x, y) = (10, 29)$ and is oriented left (field A).

A. *Coordinate Encoding*

The bits of a (x, y) coordinate are encoded "falling down" the diagonal. Each power of two corresponding to the binary encoding is displayed in Figure 3, Figure 4 for the 4x4 macro and 3x3 micro markers. To read any (x, y) coordinate, just sum up the powers of 2 where a square is present, either above or below the diagonal. **Above** the diagonal, we encode x, **below** the diagonal, we encode y. This represents x / y ("x over y"). Then, $(0, 0)$ is a diagonal line, while a $(7, 7) = (111_2, 111_2)$ micro marker is a composite 3x3 square.

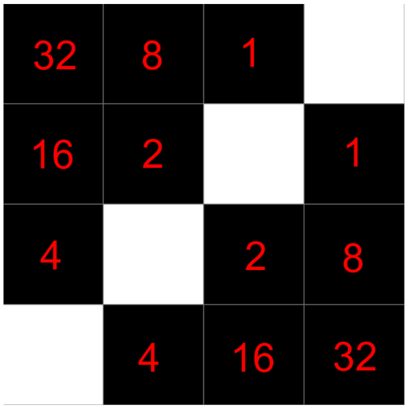

Figure 3: Binary encoding inside a macro marker.

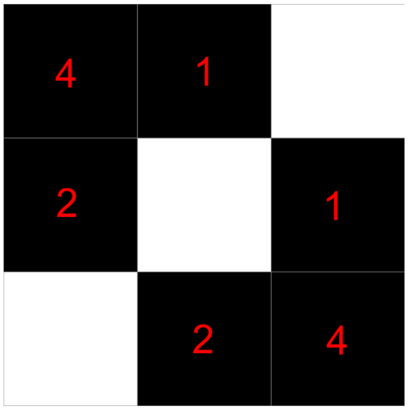

Figure 4: Binary encoding inside a micro marker.

B. *Field Encoding*

Around the bitmarker, 3 additional squares are placed to indicate the field. These are centered to distinguish them from the coordinate encoding. One can imagine a missing fourth square, which would complete the diamond shape. The location of this gap determines the field:

- **Left** = A.
- **Up** = B.
- **Down** = C.
- **Right** = D.

See Figure 2, which is oriented left, thus indicating field A.

IV. GUI

To aid in decoding bitmarkers, a GUI was developed using Lua and the *Love2D* framework. It renders a grid of squares, representing the bitmarker. These squares can be toggled from white (1) to black (0) by clicking on them. For more information and download, go to https://github.com/PvdBerg1998/BitmarkerLove.

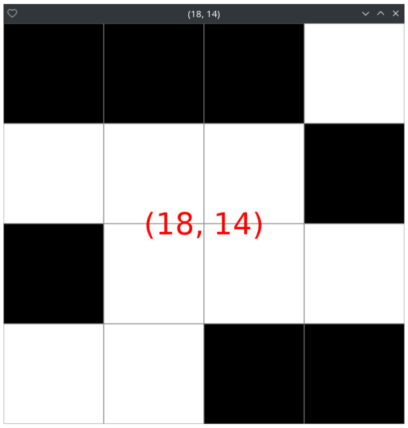

Figure 5: Bitmarker decoder GUI.