1 Definitions

1.1 Misc

Let $m = \left\lceil \left(\frac{N}{93}\right)^{\frac{1}{d}} \right\rceil$, base of the counter

$$
\begin{aligned}
MSR &= \text{ most signifcant digit region} \\
\mathcal{C}_0 &= \text{starting value of counter} \\
d &= \lceil \log_m \mathcal{C}_0 \rceil = \left\lfloor \frac{k}{2} \right\rfloor, \text{ number of digits per row} \\
\mathcal{C}_f &= m^d, \text{ final value of the counter} \\
\mathcal{C}_\Delta &= \mathcal{C}_f - \mathcal{C}_0, \text{ number of rows/ times to count} \\
l &= \lceil \log m \rceil + 2, \text{ bits needed to encode each digit in binary, plus 2 for MSR and MSD}
\end{aligned}
$$

1.2 Determining the starting value $\mathcal{C}_0$

...therefore, let $d = \lfloor \frac{k}{2} \rfloor$, $m = \left\lceil \left(\frac{N}{93}\right)^{\frac{1}{d}} \right\rceil$, $l = \lceil \log m \rceil + 2$, $\mathcal{C}_0 = m^d - \left\lfloor \frac{N-3l-76}{3l+90} \right\rfloor$, where d is the number of digits per row of the counter, m is the base of the counter, l is the number of bits needed to encode each digit in binary plus 2 for indicating whether a digit is in the MSR and is the MSD in that region, and $\mathcal{C}_0$ is the start of the counter in decimal.

In general, the height of a digit region is $3l + 90$. There are two cases when the height is different, namely in the first and last digit regions, where the height is $3l + 91$ and $3l + 75$, respectively. Let h be the height of the construction before any filler/roof tiles are added. If we define $\mathcal{C}_\Delta$ as the number of `Counter` unit rows, then $h = (\mathcal{C}_\Delta - 1)(3l + 90) + (3l + 91) + (3l + 75)$, simplifying to $\mathcal{C}_\Delta(3l + 90) + 3l + 76$. So then the maximum height of the counter is $m^d(3l + 90) + 3l + 76$. Since our goal is to end with a rectangle of height N, we need to pick a base such that the counter can increment so many times that when it stops, it is at least N.

Lemma 1. *$N \leq m^d(3l + 90) + 3l + 76$.*

Proof.

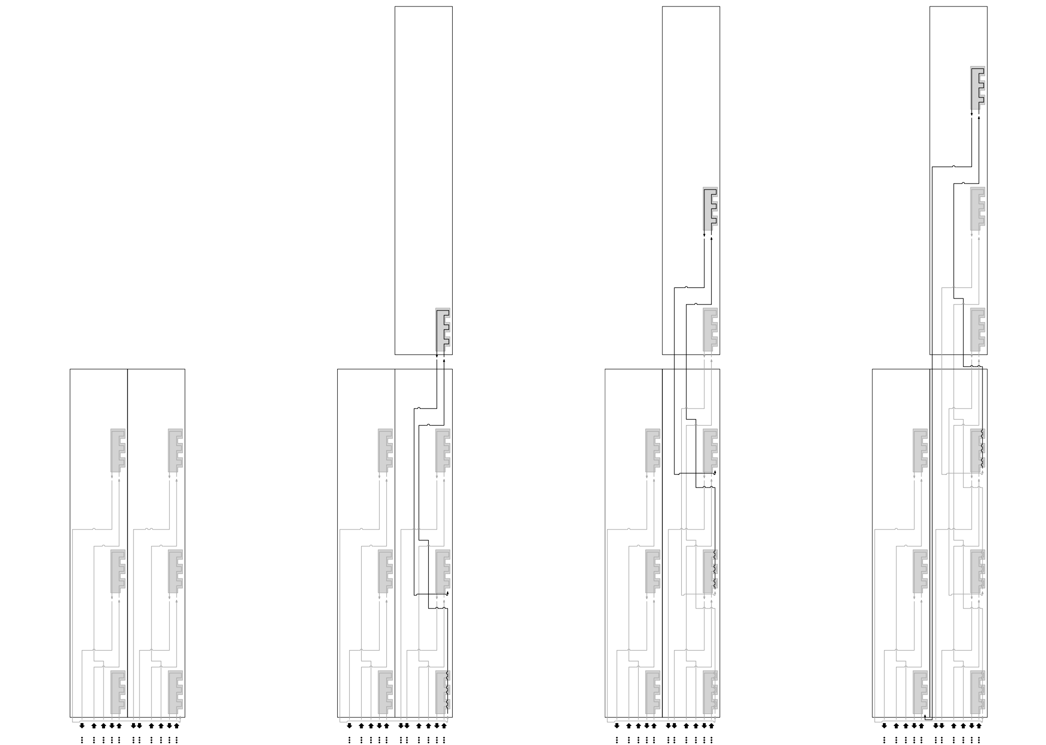
$$
\begin{aligned}
N = 93\left(\frac{N}{93}\right) &= 93\left(\left(\frac{N}{93}\right)^{\frac{1}{d}}\right)^d \leq 93\left\lceil \left(\frac{N}{93}\right)^{\frac{1}{d}} \right\rceil^d \\
&= 93m^d \leq 3lm^d + 90m^d \leq 3lm^d + 90m^d + 3l + 76 \\
&= m^d(3l + 90) + 3l + 76
\end{aligned}
$$

□

1.3 Filling in the gaps

...this means that the number of `Counter` unit rows $\mathcal{C}_\Delta$ is $m^d - \mathcal{C}_0$, where we have defined $\mathcal{C}_0$ as the starting value of the counter. To choose the best starting value, we find the value for $\mathcal{C}_\Delta$ that gets h as close to N without exceeding N. It follows from the equation $h = \mathcal{C}_\Delta(3l + 90) + 3l + 76$, that $\mathcal{C}_\Delta = \left\lfloor \frac{N-3l-76}{3l+90} \right\rfloor$. Thus, $\mathcal{C}_0 = m^d - \left\lfloor \frac{N-3l-76}{3l+90} \right\rfloor$. As a result of each digit requiring a width of 2 tiles, if k is odd, one additional tile column must be added. The number of filler tiles needed for the width is $k \mod 2$, and the number of filler tiles for the height is $N - 3l - 76 \mod 3l + 90$.

2 General counter



(a) A "clean" counter row, before any reading has started.

(b) Read digit 1 in the current row, write digit 1 in the next row.

(c) Read digit 2 in the current row, write digit 2 in the next row.

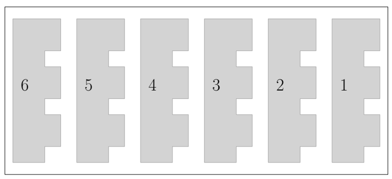
(d) Read digit 3 in the current row, write digit 3 in the next row.

Figure 1: This illustrates how a counter reads and writes a digit region, in a general sense. The counter starts in the rightmost digit region by reading the bottommost digit within that region. After reading digit 1 in the current row, the corresponding digit region in the next row be started in the next row. The counter writes the first digit in the next row, and then returns to the second digit in the current digit region. Once all the digits in the current digit region are read and written into the next row, the counter can then do one of the following: continue reading digits by moving on to the next digit region, cross back all the way to the right of the rectangle and start reading the next row, or halt.

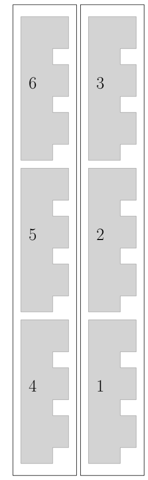
2.1 Digit region explanation (in progress)

Each logical row of the counter is made up of $\lceil \frac{d}{3} \rceil$ "digit regions". A digit region is a group of 1-3 digits, stacked on top of each other vertically. Within a digit region, the digits are sorted in order of significance, thus the most signifcant digit will be the topmost digit in that region, the middle digit will have the second highest significance and finally the bottommost digit is the least signifcant.

The leftmost digit region is most signifcant and the rightmost is the least signifcant. The counter reads the least signifcant digit (1) in digit region 1, and continues in the current row until it detects the final digit, in the MSR...



(a) Digits in a typical counter



(b) Digits in two digit regions, stacked vertically, minimizing the width.

Contrary to a typical counter, each counter row has an approximate height of 3 digits. The digits are stacked up to 3 before increasing the width..

3 Gadgets

When describing a special cases, i.e. "case x - digit y", whatever follows will only apply to the MSR (due to each case only affecting the MSR.)

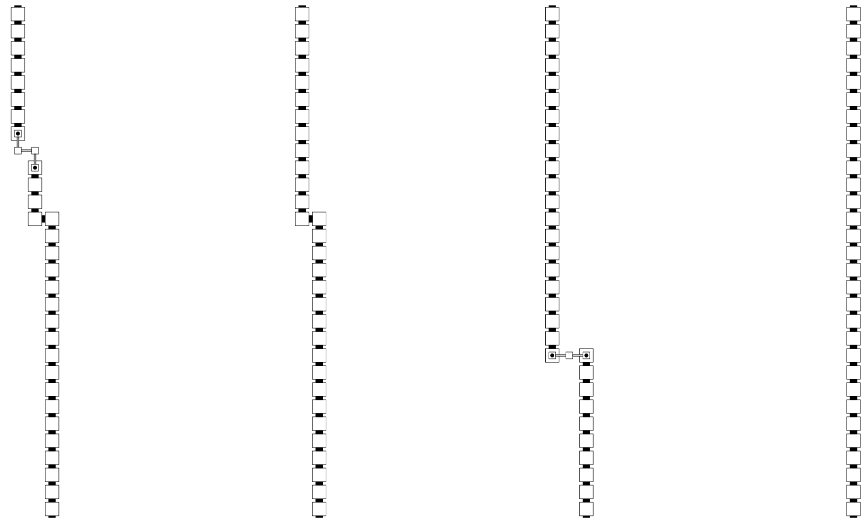
3.1 Counter Unit

3.1.1 Digit readers

3.1.2 Warping

For each $d \in \{0,1\}^l$, $i = 1, 2, 3$

- $\texttt{Pre\_Warp}(\langle d, i, \texttt{carry}\rangle) \approx O(1)$



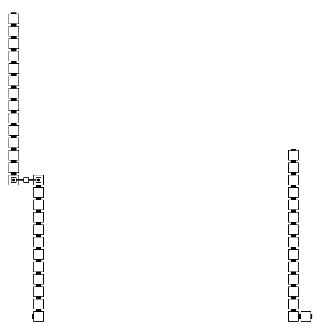
(a) General (b) Case 1 – Digit 1 (c) Case 2 – Digit 1 (d) Case 2 – Digit 2

Figure 3: `Pre_Warp` gadgets

– `First_Warp`$(\langle d, i, \texttt{carry}\rangle) \approx O(1)$ 1 tile is created

– `Warp_Bridge`$(\langle d, i, \texttt{carry}\rangle) \approx O(1)$

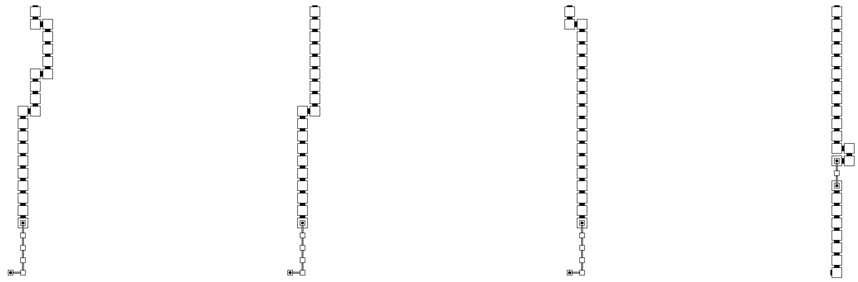
A `Warp_Bridge` gadget binds the last tile of the `First_Warp` gadgets to the first tile of the `Second_Warp` gadgets. In case 1, and case 2 - digit 1, the `Warp_Bridge` is omitted from the `Warp_Unit`.



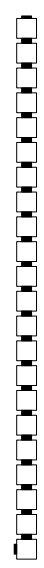
(a) General (b) Case 2 – Digit 2

– `Second_Warp`$(\langle d, i, \texttt{carry}\rangle) \approx O(1)$ 1 tile is created, $O(1)$.

– `Post_Warp`$(\langle d, i, \texttt{carry}\rangle) \approx O(1)$



(a) General Digit 1 (b) General Digits 2 and 3 (c) Case 1 – Digit 1 (d) Case 2 – Digit 1



(e) Case 2 – Digit 2

Figure 5: `Post_Warp` gadgets

Tiles created for the warp units $\approx O(\log m)$

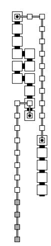
3.1.3 Digit writers

3.1.4 Digit tops

The `Digit_Top` gadgets have specific geometry, such that they allow `First_Warp` and `Second_Warp` units to "wake up" and end their warp journey. A `Digit_Top` is placed on the north end of a digit. These hold a increment/copy signal and the regional index of the next digit to read. $\approx O(\log m)$

For each `carry` $\in \{$`increment`, `copy`$\}$

- General digit tops common to all assemblies



(a) General

– create $\texttt{Digit\_Top}(\langle \texttt{carry}, 1, l \rangle)$

$$\text{input}{\rightarrow}\text{south} = \texttt{Digit\_Top}(\texttt{carry}, 1)$$
$$\text{output}{\rightarrow}\text{south} = \texttt{Return\_From\_Digit1\_Read\_Digit2}(\texttt{carry})$$

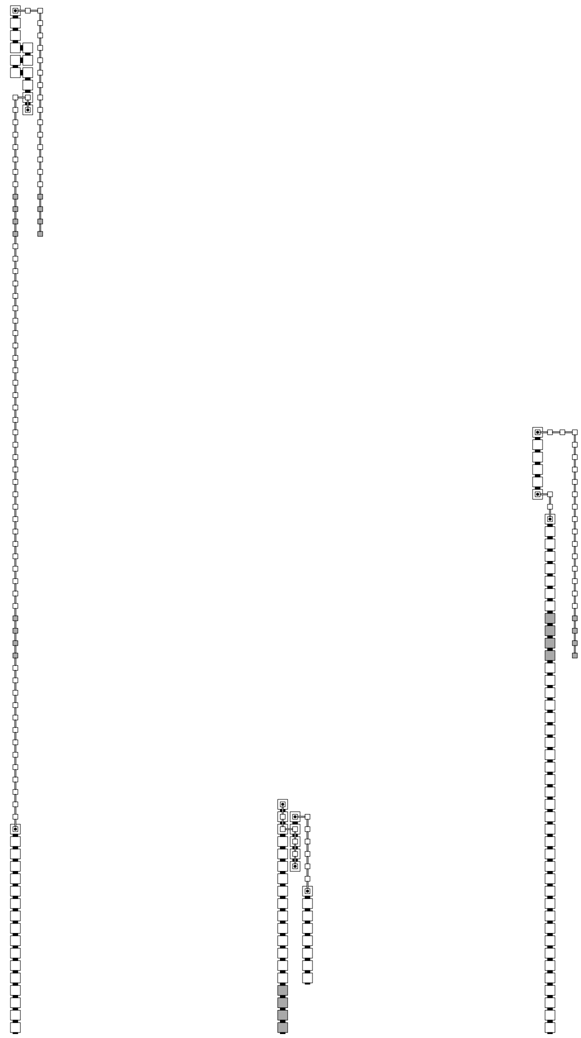
– create $\texttt{Digit\_Top}(\langle \texttt{carry}, 2, l \rangle)$

$$\text{input}{\rightarrow}\text{south} = \texttt{Digit\_Top}(\texttt{carry}, 2)$$
$$\text{output}{\rightarrow}\text{south} = \texttt{Return\_From\_Digit2\_Read\_Digit3}(\texttt{carry})$$

– create $\texttt{Digit\_Top}(\langle \texttt{carry}, 3, l \rangle)$

$$\text{input}{\rightarrow}\text{south} = \texttt{Digit\_Top}(\texttt{carry}, 3)$$
$$\text{output}{\rightarrow}\text{south} = \texttt{Return\_From\_Digit3\_Read\_Digit1}(\texttt{carry})$$

- MSR-specific digit tops. The first tile placed in all digit top gadgets is the rightmost, bottommost tile.



(a) Case 1 – Digit 1 (b) Case 2 – Digit 1 (c) Case 2 – Digit 2

  - if i is 1 and MSR contains 2 digits: create $\texttt{Digit\_Top\_Digit1\_Case2}(\langle \texttt{carry}, l \rangle)$

$$\text{input}\rightarrow\text{south} = \texttt{Digit\_Top\_Digit1\_Case2}(\texttt{carry})$$
$$\text{output}\rightarrow\text{south} = \texttt{Return\_From\_Digit1\_Read\_Digit2\_Case2}(\texttt{carry})$$

  - if i is 2 and MSR contains 2 digits: create $\texttt{Digit\_Top\_Digit2\_Case2}(\langle \texttt{carry}, l \rangle)$

$$\text{input}\rightarrow\text{south} = \texttt{Digit\_Top\_Digit2\_Case2}(\texttt{carry})$$
$$\text{output}\rightarrow\text{south} = \texttt{Return\_From\_Digit2\_Read\_Next\_Row}(\texttt{carry})$$

– if i is 3 and MSR contains 3 digits: create $\texttt{Digit\_Top\_Digit3\_Case3}(\langle\texttt{carry}, l\rangle)$

$$\text{input}\!\to\!\text{south} = \texttt{Digit\_Top\_Digit3\_Case3}(\texttt{carry})$$
$$\text{output}\!\to\!\text{south} = \texttt{Return\_From\_Digit3\_Read\_Next\_Row}(\texttt{carry})$$

3.1.5 Return paths between the MSD and LSD in different rows

The gadgets of this class hold a increment/copy signal. The height of these gadgets is dependent on l and the width is dependent of k. These gadgets are used to begin reading the first digit in the following row, once the MSD has been read in the current row.

For each $\texttt{carry} \in \{\texttt{increment}, \texttt{copy}\}$

1. `Return_From_Digit1_Read_Next_Row`

   $$\text{input}\!\to\!\text{south} = \texttt{Return\_From\_Digit1\_Read\_Next\_Row}(\texttt{carry})$$
   $$\text{output}\!\to\!\text{north} = \texttt{Digit\_Reader}(\texttt{carry}, 1)$$

2. `Return_From_Digit2_Read_Next_Row`

   $$\text{input}\!\to\!\text{north} = \texttt{Return\_From\_Digit2\_Read\_Next\_Row}(\texttt{carry})$$
   $$\text{output}\!\to\!\text{north} = \texttt{Digit\_Reader}(\texttt{carry}, 1)$$

3. `Return_From_Digit3_Read_Next_Row`

   $$\text{input}\!\to\!\text{north} = \texttt{Return\_From\_Digit3\_Read\_Next\_Row}(\texttt{carry})$$
   $$\text{output}\!\to\!\text{north} = \texttt{Digit\_Reader}(\texttt{carry}, 1)$$

3.1.6 Return paths between digits in the same row

The gadgets of this class hold a increment/copy signal and the regional index of the next digit to read. The height of these gadgets is dependent on l. These gadgets are used so that upon writing a digit, the counter is able to move back down to the next digit in the current row, and continue reading.

– `Return_From_Digit1_Read_Digit2`

$$\text{input}\!\to\!\text{north} = \texttt{Return\_From\_Digit1\_Read\_Digit2}(\texttt{carry})$$
$$\text{output}\!\to\!\text{north} = \texttt{Digit\_Reader}(\texttt{carry}, 2)$$

– `Return_From_Digit1_Read_Digit2_Case2`

$$\text{input}{\to}\text{north} = \texttt{Return\_From\_Digit1\_Read\_Digit2\_Case2}(\texttt{carry})$$
$$\text{output}{\to}\text{north} = \texttt{Digit\_Reader}(\texttt{carry}, 2)$$

– `Return_From_Digit2_Read_Digit3`

$$\text{input}{\to}\text{north} = \texttt{Return\_From\_Digit2\_Read\_Digit3}(\texttt{carry})$$
$$\text{output}{\to}\text{north} = \texttt{Digit\_Reader}(\texttt{carry}, 3)$$

– `Return_From_Digit3_Read_Digit1`

$$\text{input}{\to}\text{north} = \texttt{Return\_From\_Digit3\_Read\_Digit1}(\texttt{carry})$$
$$\text{output}{\to}\text{north} = \texttt{Digit\_Reader}(\texttt{carry}, 1)$$

3.2 Seed Unit