

1 Definitions

1.1 Misc

Let $m = \left\lceil \left(\frac{N}{102} \right)^{\frac{1}{d}} \right\rceil$, base of the counter

MSR = most significant digit region

\mathcal{C}_0 = starting value of counter

$d = \lceil \log_m \mathcal{C}_0 \rceil = \left\lfloor \frac{k}{2} \right\rfloor$, number of digits per row

$\mathcal{C}_f = m^d$, final value of the counter

$\mathcal{C}_\Delta = \mathcal{C}_f - \mathcal{C}_0$, number of rows/ times to count

$l = \lceil \log m \rceil + 2$, bits needed to encode each digit in binary, plus 2 for MSR and MSD

1.2 Determining the starting value \mathcal{C}_0

...therefore, let $d = \lfloor \frac{k}{2} \rfloor$, $m = \left\lceil \left(\frac{N}{102} \right)^{\frac{1}{d}} \right\rceil$, $l = \lceil \log m \rceil + 2$, $\mathcal{C}_0 = m^d - \left\lfloor \frac{N-12l-76}{12l+90} \right\rfloor$, 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 $12l + 90$. There are two cases when the height is different, namely in the first and last digit regions, where the height is $12l + 91$ and $12l + 75$, respectively. Let h be the height of the construction before any filler/roof tiles are added. If we define \mathcal{C}_Δ as the number of **Counter** unit rows, then $h = (\mathcal{C}_\Delta - 1)(12l + 90) + (12l + 91) + (12l + 75)$, simplifying to $\mathcal{C}_\Delta(12l + 90) + 12l + 76$. So then the maximum height of the counter is $m^d(12l + 90) + 12l + 76$. 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. $N \leq m^d(12l + 90) + 12l + 76$.

Proof.

$$\begin{aligned} N &= 102 \left(\frac{N}{102} \right) = 102 \left(\left(\frac{N}{102} \right)^{\frac{1}{d}} \right)^d \leq 102 \left\lceil \left(\frac{N}{102} \right)^{\frac{1}{d}} \right\rceil^d \\ &= 102m^d \leq 12lm^d + 90m^d \leq 12lm^d + 90m^d + 12l + 76 \\ &= m^d(12l + 90) + 12l + 76 \end{aligned}$$

□

1.3 Filling in the gaps

...this means that the number of **Counter** unit rows \mathcal{C}_Δ 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}_Δ that gets h as close to N without exceeding N . It follows from the equation $h = \mathcal{C}_\Delta(12l + 90) + 12l + 76$, that $\mathcal{C}_\Delta = \left\lfloor \frac{N-12l-76}{12l+90} \right\rfloor$.

Thus, $\mathcal{C}_0 = m^d - \left\lfloor \frac{N-12l-76}{12l+90} \right\rfloor$. 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 $k \bmod 2$, and the number of filler tiles for the height is $N - 12l - 76 \bmod 12l + 90$.

2 General counter



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 vertically on top of one another. Within a digit region, the digits are sorted in order of significance, thus the top digit is the most significant digit, the middle digit is second most significant and the bottommost digit is the least significant.

The leftmost digit region is most significant and the rightmost is the least significant. The counter reads the least significant digit (1) in digit region 1, and continues in the current row until it detects the final digit, in the most significant digit region (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 $\approx 12l$. The digits are stacked up to 3 before increasing the width.

2.2 Detecting the edges

The counter must detect if a digit is in the MSR and if it's in the MSR, whether or not it is the most significant digit. To do this, all digits are encoded with two additional bits on the least significant end. If bit 0 is 1, the reader tiles know they could be reading the most significant digit (MSD) or in case 2, the second most significant digit. If bit 1 is 1, the digit currently being read is the MSD, otherwise the digit is digit 1 in case 2.

bit ₁	bit ₀	Meaning
0	0	digit is not in MSR
0	1	digit is in the MSR but is not the MSD
1	0	
1	1	digit is in the MSR and is MSD

2.3 Tile set

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

2.3.1 Line Gadgets

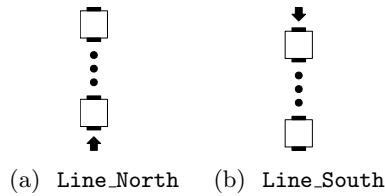
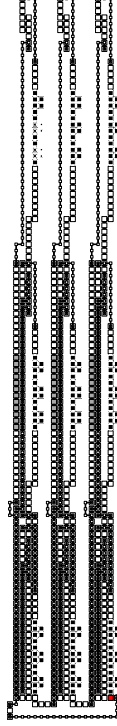


Figure 3: Line gadgets

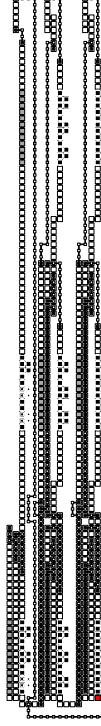
We will use the notation `LineN_North` and `LineN_South` where `N` corresponds to the length of a specific line gadget.

2.4 Initial Value (updated to assemble right to left like the other gadgets)

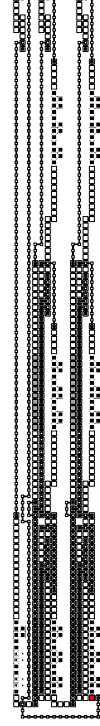
We begin by encoding C_0 with the Seed unit. It has $\lceil \frac{d}{3} \rceil$ digit regions. Each digit region has three digits, except for the most significant digit region (MSR) which has $d \bmod 3$ if $d \bmod 3 \neq 0$, otherwise it has 3 digits.



(a) Initial value case 3



(b) Initial value case 2



(c) Initial value case 1

For each $i = 0, \dots, d$

- Create `Seed_Start(⟨DigitWriter, Seed, i, op⟩)`
- **Digit:** For each $j = 0, \dots, l$ and each b in $\text{bin}(C_0[i])[j]$:
 - if $j = 0$: create `Digit_Writer(⟨Digit_1⟩, ⟨Seed.Bit, i, j + 1⟩)` from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.
 - if $0 \leq j \leq l$: create `Digit_Writer(⟨Seed.Bit, i, j⟩, ⟨Seed.Bit, i, j + 1⟩)` from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.
 - if $j = l$: `Digit_Writer(⟨Seed.Bit, i, j⟩, ⟨Seed.DigitTop_1, i⟩)` from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.
- **Digit_Top:** The following statements create the gadget shown in Figure 11a
 - Create `North_Line5(⟨Seed.DigitTop_1, i⟩, ⟨Seed.DigitTop_1.A, i⟩)` from the micro-gadget shown in Figure 3a
 - Create `Topper(⟨DigitTop_1.A, i⟩, ⟨DigitTop_1.B, i⟩)` from the micro-gadget shown in Figure 10a

- Create `South_Line4l(⟨DigitTop_1_B, i⟩, ⟨ReturnD3ReadD1, i⟩)` from the micro-gadget shown in Figure 3b
- Create `Return.From.Digit1.Read.Digit2(⟨TODO⟩, ⟨TODO⟩)`
- Create `SingleTile(⟨TODO⟩, ⟨TODO⟩)`
- Create `Second.Warp(⟨TODO⟩, ⟨TODO⟩)`
- Create `Post.Warp(⟨TODO⟩, ⟨TODO⟩)`
- Create `Digit.Writer(⟨TODO⟩, ⟨TODO⟩)`
- Create `Digit.Top(⟨TODO⟩, ⟨TODO⟩)`
- Create `Return.From.Digit2.Read.Digit3(⟨TODO⟩, ⟨TODO⟩)`
- Create `Three.Digit.Region.Connector(⟨TODO⟩, ⟨TODO⟩)`
- Create `First.Warp(⟨TODO⟩, ⟨TODO⟩)`
- Create `Warp.Bridge(⟨TODO⟩, ⟨TODO⟩)`
- Create `Second.Warp(⟨TODO⟩, ⟨TODO⟩)`
- Create `Post.Warp(⟨TODO⟩, ⟨TODO⟩)`
- Create `Digit.Writer(⟨TODO⟩, ⟨TODO⟩)`
- Create `Digit.Top(⟨TODO⟩, ⟨TODO⟩)`
- Create `Return.From.Digit3.Read.Next.Row(⟨TODO⟩, ⟨TODO⟩)`
- Create `Digit.Region.Bridge(⟨TODO⟩, ⟨TODO⟩)`

2.5 Counter Unit

2.5.1 Digit readers

- For each $i = 1, 2, 3$, $j = l - 1, \dots, 1$, $u \in \{0, 1\}^j$, and $\text{op} \in \{\text{increment}, \text{copy}\}$:
 - if $j = 0$: Create `Digit.Reader(⟨DigitReader, i, λ, op⟩, ⟨DigitReader, i, 0, op⟩, ⟨DigitReader, i, 1, op⟩)` from the general gadget in Figure 5
 - if $1 \leq j \leq l - 2$: Create `Digit.Reader(⟨DigitReader, i, u, op⟩, ⟨DigitReader, i, 0u, op⟩, ⟨DigitReader, i, 1u, op⟩)` from the general gadget in Figure 5
 - if $j = l - 1$: Create `Digit.Reader(⟨DigitReader, i, u, op⟩, ⟨PreWarp, i, 0u, op⟩, ⟨PreWarp, i, 1u, op⟩)` from the general gadget in Figure 5

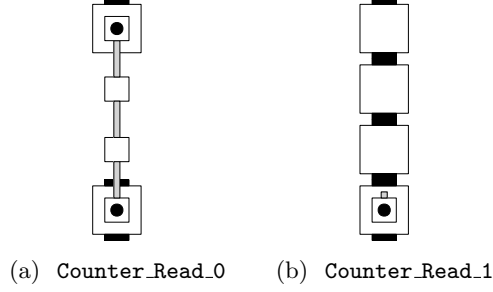


Figure 5: Counter_Read

2.5.2 Warping

For each $i = 1, 2, 3$, $u \in \{0, 1\}^l$, and each $op \in \{\text{increment}, \text{copy}\}$

- **Pre_Warp:** These gadgets take the bits read from the **Digit_Reader** gadgets and convert it into a signal used until the **Digit_Top** gadgets are attached after writing the current digit. The signal started by this gadget is used to tell the counter whether to begin reading another digit in the current row, or cut across the rectangle and begin reading the first digit in the next row.
 - if u ends with 00: Create $\text{Pre_Warp}(\langle \text{PreWarp}, i, u, op \rangle, \langle \text{FirstWarp}, i, u, op \rangle)$ from the general gadget in Figure 6a
 - if u ends with 01: Create $\text{Pre_Warp}(\langle \text{PreWarp}, i, u, op \rangle, \langle \text{FirstWarp}, i, u, op, msr \rangle)$ from the general gadget in Figure 6c
 - if u ends with 11: Create $\text{Pre_Warp}(\langle \text{PreWarp}, i, u, op \rangle, \langle \text{FirstWarp}, i, u, op, msr, msd \rangle)$ from the general gadget in Figure 6b if $i = 1$ (case 1), or Figure 6d. $i = 2$ (case 2), or Figure 6a if $i = 3$ (case 3).

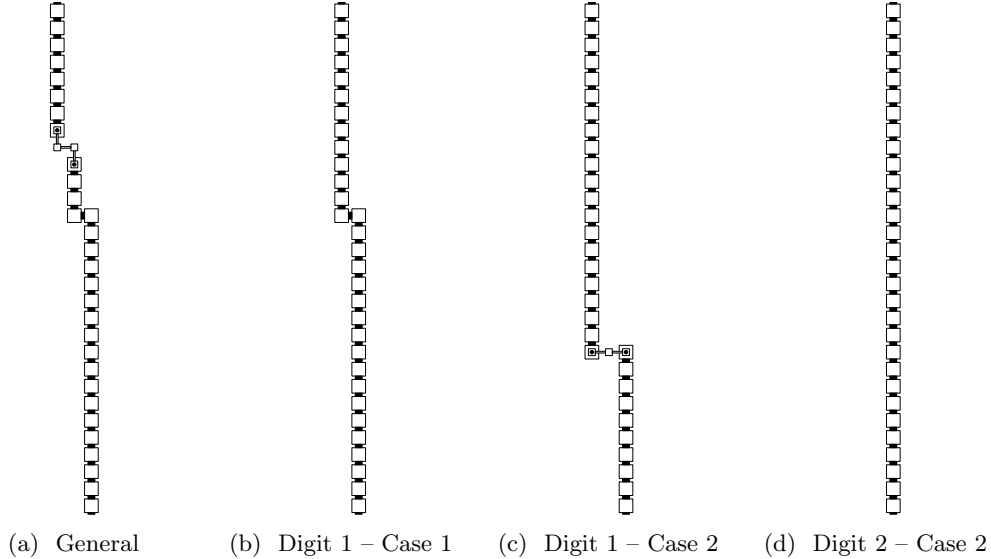


Figure 6: Pre_Warp gadgets

- **First_Warp:** A **First_Warp** connects to a **Warp_Bridge** gadget in all cases except when it's assembling in the MSR and it is digit 1 in case 1 or 2, in which the **First_Warp** gadget attaches directly to a **Post_Warp**.
 - Create **First_Warp**($\langle \text{FirstWarp}, i, u, \text{op} \rangle$,
 $\langle \text{FirstWarp}, i, u, \text{op} \rangle$,
 $\langle \text{WarpBridge}, i, u, \text{op} \rangle$)
 - (digit 1, case 2): Create **First_Warp**($\langle \text{FirstWarp}, i, u, \text{op}, \text{msr} \rangle$,
 $\langle \text{FirstWarp}, i, u, \text{op}, \text{msr} \rangle$,
 $\langle \text{PostWarp}, i, u, \text{op}, \text{msr} \rangle$)
 - (digit 1, case 1): Create **First_Warp**($\langle \text{FirstWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$,
 $\langle \text{FirstWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$,
 $\langle \text{PostWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$)
 - (digit 2, case 2): Create **First_Warp**($\langle \text{FirstWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$,
 $\langle \text{FirstWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$,
 $\langle \text{WarpBridge}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$)
 - (digit 3, case 3): Create **First_Warp**($\langle \text{FirstWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$,
 $\langle \text{FirstWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$,
 $\langle \text{WarpBridge}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$)
- **Warp_Bridge:** a **Warp_Bridge** gadget binds the last tile of the **First_Warp** gadgets to the first tile of the **Second_Warp** gadgets. For digit 1 in cases 1 and 2, the **Warp_Bridge** is omitted from the **Warp_Unit**.
 - if u ends with 00:
 Create **Warp_Bridge**($\langle \text{WarpBridge}, i, u, \text{op} \rangle$, $\langle \text{SecondWarp}, i, u, \text{op} \rangle$)
 from the general gadget in Figure 7a
 - if u ends with 11 and i is 2:
 Create **Warp_Bridge**($\langle \text{WarpBridge}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$, $\langle \text{SecondWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$)
 from the general gadget in Figure 7b
 - if u ends with 11 and i is 3:
 Create **Warp_Bridge**($\langle \text{WarpBridge}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$, $\langle \text{SecondWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$)
 from the general gadget in Figure 7a

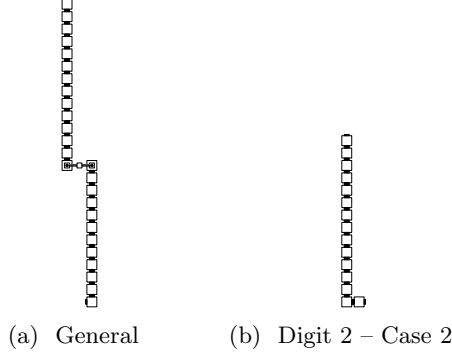


Figure 7: Warp_Bridge gadgets

- Second_Warp:

- Create Second_Warp($\langle \text{SecondWarp}, i, u, \text{op} \rangle$,
 $\langle \text{SecondWarp}, i, u, \text{op} \rangle$,
 $\langle \text{PostWarp}, i, u, \text{op} \rangle$)
- Create Second_Warp($\langle \text{SecondWarp}, i, u, \text{op}, \text{msr} \rangle$,
 $\langle \text{SecondWarp}, i, u, \text{op}, \text{msr} \rangle$,
 $\langle \text{PostWarp}, i, u, \text{op}, \text{msr} \rangle$)
- Create Second_Warp($\langle \text{SecondWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$,
 $\langle \text{SecondWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$,
 $\langle \text{PostWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$)

- Post_Warp:

- if u ends with 00:
Create Post_Warp($\langle \text{PostWarp}, i, u, \text{op} \rangle$, $\langle \text{DigitWriter}, i, u, \text{op} \rangle$)
Depending on i the gadget created in this step will differ: If i is 1 use from the general gadget in Figure 8a otherwise (i is 2 or 3) use from the general gadget in Figure 8b.
- if u ends with 01:
Create Post_Warp($\langle \text{PostWarp}, i, u, \text{op}, \text{msr} \rangle$, $\langle \text{DigitWriter}, i, u, \text{op}, \text{msr} \rangle$)
from the general gadget in Figure 8d.
- if u ends with 11:
Create Post_Warp($\langle \text{PostWarp}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$, $\langle \text{DigitWriter}, i, u, \text{op}, \text{msr}, \text{msd} \rangle$)
Depending on the number of digits in the MSR, the gadget created in this step will differ. If i is 1 (case 1) use the general gadget in Figure 8c. If i is 2 (case 2) use the general gadget in Figure 8e. If i is 3 (case 3) use the general gadget in Figure 8b.

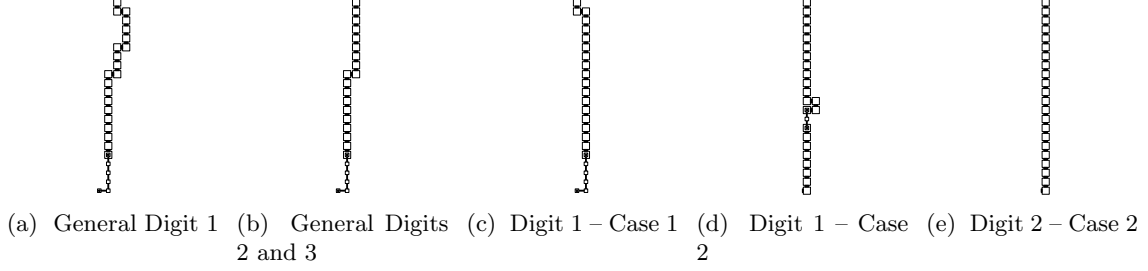


Figure 8: Post_Warp gadgets

2.5.3 Digit writers

- For each $i = 1, 2, 3$, $j = l - 1, \dots, 1$, $u \in \{0, 1\}^j$, and $op \in \{\text{increment}, \text{copy}\}$:
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, u0, op \rangle, \langle \text{DigitWriter}, i, u, op \rangle$)` from the general gadget in Figure 9a
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, u1, op \rangle, \langle \text{DigitWriter}, i, u, op \rangle$)` from the general gadget in Figure 9b
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, u0, op, msr \rangle, \langle \text{DigitWriter}, i, u, op, msr \rangle$)` from the general gadget in Figure 9a
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, u1, op, msr \rangle, \langle \text{DigitWriter}, i, u, op, msr \rangle$)` from the general gadget in Figure 9b
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, u0, op, msr, msd \rangle, \langle \text{DigitWriter}, i, u, op, msr, msd \rangle$)` from the general gadget in Figure 9a
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, u1, op, msr, msd \rangle, \langle \text{DigitWriter}, i, u, op, msr, msd \rangle$)` from the general gadget in Figure 9b
- For each $i = 1, 2, 3$ and each $op \in \{\text{increment}, \text{copy}\}$:
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, 0, op \rangle, \langle \text{DigitTop}, i, op \rangle$)` from the general gadget in Figure 9a
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, 1, op \rangle, \langle \text{DigitTop}, i, op \rangle$)` from the general gadget in Figure 9b
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, 0, op, msr \rangle, \langle \text{DigitTop}, i, op, msr \rangle$)` from the general gadget in Figure 9a
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, 1, op, msr \rangle, \langle \text{DigitTop}, i, op, msr \rangle$)` from the general gadget in Figure 9b
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, 0, op, msr, msd \rangle, \langle \text{DigitTop}, i, op, msr, msd \rangle$)` from the general gadget in Figure 9a
 - Create `Digit_Writer($\langle \text{DigitWriter}, i, 1, op, msr, msd \rangle, \langle \text{DigitTop}, i, op, msr, msd \rangle$)` from the general gadget in Figure 9b

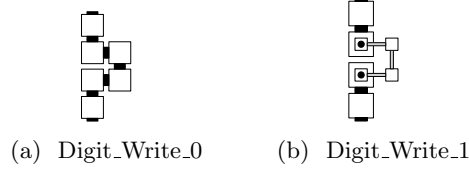


Figure 9: Digit_Write gadgets

2.5.4 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. After **Digit_Top** has assembled on the north end of a digit we

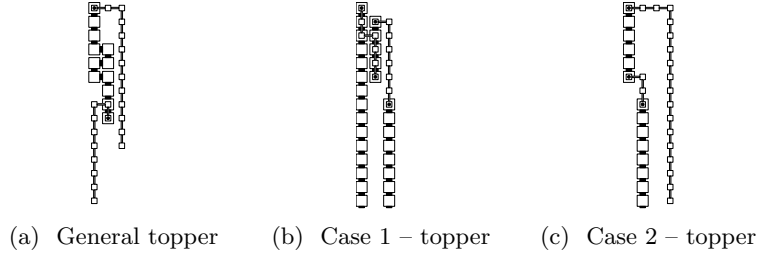


Figure 10: Topper micro-gadgets

For each $op \in \{\text{increment}, \text{copy}\}$

- Digit 1 (general): the following statements create the gadget shown in Figure 11a
 - Create **North_Line5**($\langle \text{DigitTop}, 1, op \rangle, \langle \text{DigitTop_1_A}, op \rangle$) from the micro-gadget shown in Figure 3a
 - Create **Topper**($\langle \text{DigitTop_1_A}, op \rangle, \langle \text{DigitTop_1_B}, op \rangle$) from the micro-gadget shown in Figure 10a
 - Create **South_Line4**($\langle \text{DigitTop_1_B}, op \rangle, \langle \text{ReturnD1ReadD2}, op \rangle$) from the micro-gadget shown in Figure 3b

- Digit 1 (MSR): the following statements create the gadget shown in Figure 11d
 - Create **Topper**($\langle \text{DigitTop}, 1, op, msr \rangle, \langle \text{DigitTop_1_MSR_A}, op \rangle$) from the micro-gadget shown in Figure 10b
 - Create **South_Line4**($\langle \text{DigitTop_1_MSR_A}, op \rangle, \langle \text{ReturnD1ReadD2} - \text{Case2}, op \rangle$) from the micro-gadget shown in Figure 3b

- Digit 1 (MSD): the following statements create the gadget shown in Figure 11c
 - Create **North_Line4**($\langle \text{DigitTop}, 1, op, msr, msd \rangle, \langle \text{DigitTop_1_MSD_A}, op \rangle$) from the micro-gadget shown in Figure 3a

- Create `North_Line4`($\langle \text{DigitTop_1_MSD_A}, \text{op} \rangle$, $\langle \text{DigitTop_1_MSD_B}, \text{op} \rangle$) from the micro-gadget shown in Figure 3a
 - Create `Topper`($\langle \text{DigitTop_1_MSD_B}, \text{op} \rangle$, $\langle \text{DigitTop_1_MSD_C}, \text{op} \rangle$) from the micro-gadget shown in Figure 10a
 - Create `South_Line4`($\langle \text{DigitTop_1_MSD_C}, \text{op} \rangle$, $\langle \text{DigitTop_1_MSD_D}, \text{op} \rangle$) from the micro-gadget shown in Figure 3b
 - Create `South_Line30`($\langle \text{DigitTop_1_MSD_D}, \text{op} \rangle$, $\langle \text{DigitTop_1_MSD_E}, \text{op} \rangle$) from the micro-gadget shown in Figure 3b
 - Create `South_Line4`($\langle \text{DigitTop_1_MSD_E}, \text{op} \rangle$, $\langle \text{DigitTop_1_MSD_F}, \text{op} \rangle$) from the micro-gadget shown in Figure 3b
 - Create `South_Line14`($\langle \text{DigitTop_1_MSD_F}, \text{op} \rangle$, $\langle \text{DigitTop_1_MSD_G}, \text{op} \rangle$) from the micro-gadget shown in Figure 3b
 - Create `South_Line17`($\langle \text{DigitTop_1_MSD_G}, \text{op} \rangle$, $\langle \text{ReturnD1ReadNextRow}, \text{op} \rangle$) from the micro-gadget shown in Figure 3b
- Digit 2 (general): the following statements create the gadget shown in Figure 11a
 - Create `North_Line5`($\langle \text{DigitTop}, 2, \text{op} \rangle$, $\langle \text{DigitTop_2_A}, \text{op} \rangle$) from the micro-gadget shown in Figure 3a
 - Create `Topper`($\langle \text{DigitTop_2_A}, \text{op} \rangle$, $\langle \text{DigitTop_2_B}, \text{op} \rangle$) from the micro-gadget shown in Figure 10a
 - Create `South_Line4`($\langle \text{DigitTop_2_B}, \text{op} \rangle$, $\langle \text{ReturnD2ReadD3}, \text{op} \rangle$) from the micro-gadget shown in Figure 3b
- Digit 2 (MSD): the following statements create the gadget shown in Figure 11b
 - Create `North_Line4`($\langle \text{DigitTop}, 2, \text{op}, \text{msr}, \text{msd} \rangle$, $\langle \text{DigitTop_2_MSD_A}, \text{op} \rangle$) from the micro-gadget shown in Figure 3a
 - Create `Topper`($\langle \text{DigitTop_2_MSD_A}, \text{op} \rangle$, $\langle \text{DigitTop_2_MSD_B}, \text{op} \rangle$) from the micro-gadget shown in Figure 10c
 - Create `South_Line4`($\langle \text{DigitTop_2_MSD_B}, \text{op} \rangle$, $\langle \text{DigitTop_2_MSD_C}, \text{op} \rangle$) from the micro-gadget shown in Figure 3b
 - Create `South_Line30`($\langle \text{DigitTop_2_MSD_C}, \text{op} \rangle$, $\langle \text{ReturnD2ReadNextRow}, \text{op} \rangle$) from the micro-gadget shown in Figure 3b
- Digit 3 (general): the following statements create the gadget from Figure 11a
 - Create `North_Line5`($\langle \text{DigitTop}, 3, \text{op} \rangle$, $\langle \text{DigitTop_3_A}, \text{op} \rangle$) from the micro-gadget shown in Figure 3a
 - Create `Topper`($\langle \text{DigitTop_3_A}, \text{op} \rangle$, $\langle \text{DigitTop_3_B}, \text{op} \rangle$) from the micro-gadget shown in Figure 10a

- Create `South_Line4`($\langle \text{DigitTop_3_B}, \text{op} \rangle, \langle \text{ReturnD3ReadD1}, \text{op} \rangle$) from the micro-gadget shown in Figure 3b
- Digit 3 (MSD): the following statements create the gadget from Figure 11a
 - Create `North_Line5`($\langle \text{DigitTop}, 3, \text{op}, \text{msr}, \text{msd} \rangle, \langle \text{DigitTop_3_MSD_A}, \text{op} \rangle$) from the micro-gadget shown in Figure 3a
 - Create `Topper`($\langle \text{DigitTop_3_MSD_A}, \text{op} \rangle, \langle \text{DigitTop_3_MSD_B}, \text{op} \rangle$) from the micro-gadget shown in Figure 10a
 - Create `South_Line4`($\langle \text{DigitTop_3_MSD_B}, \text{op} \rangle, \langle \text{ReturnD3ReadNextRow}, \text{op} \rangle$) from the micro-gadget shown in Figure 3b

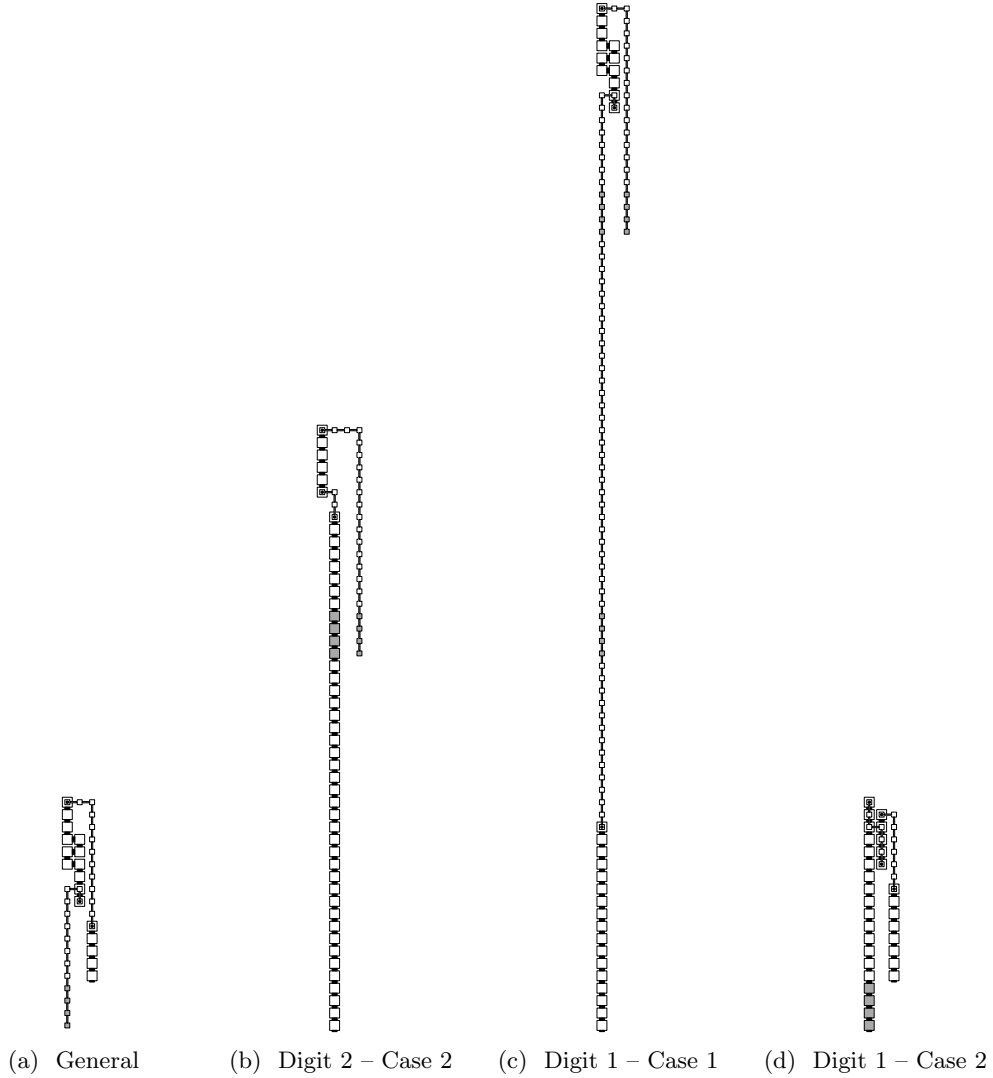


Figure 11: `Digit_Top` gadgets

2.5.5 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.

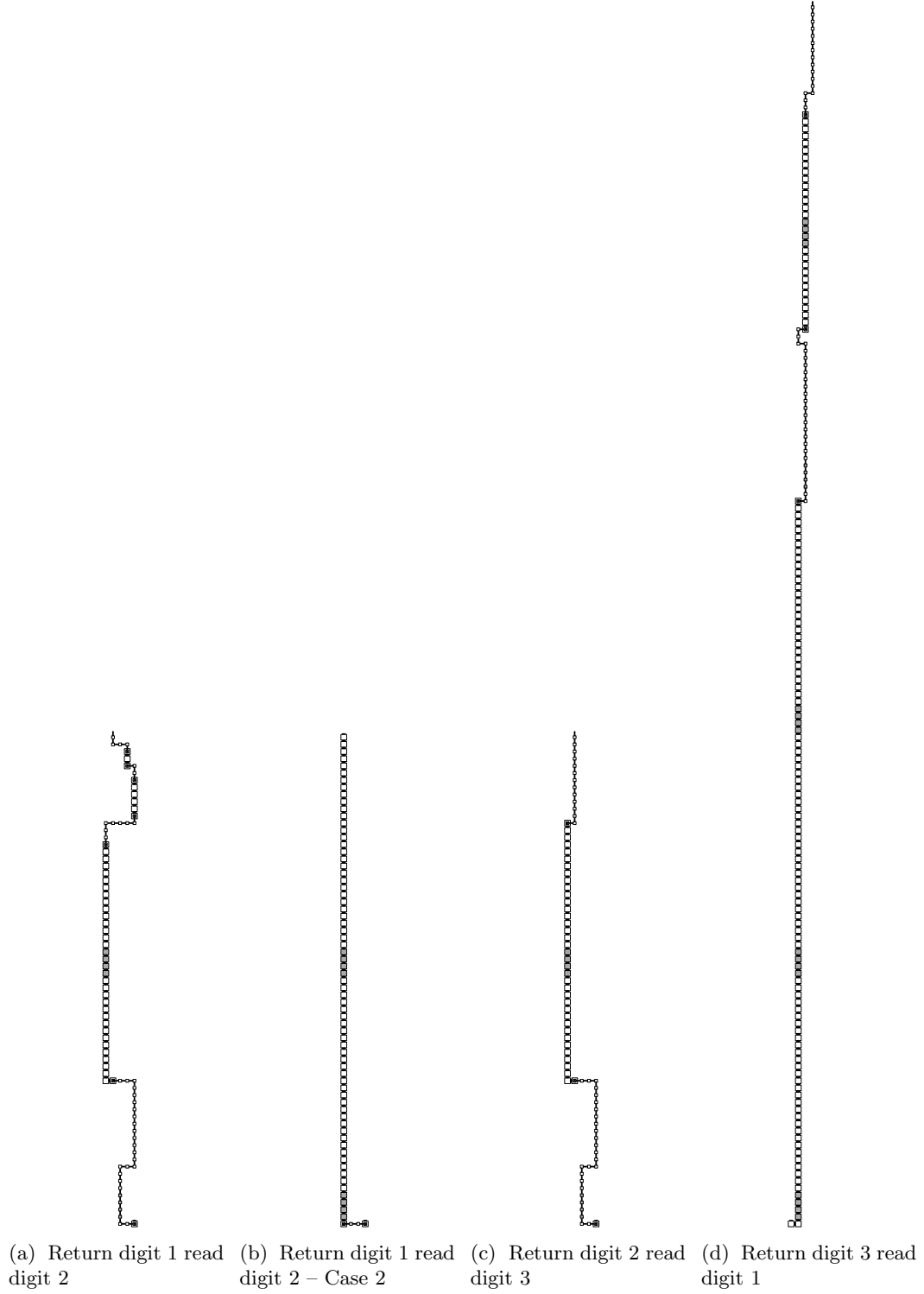


Figure 12: `Return_From_Digit_Read_Digit` gadgets. These gadgets assemble north to south, starting on the south side of a digit top.

For each $op \in \{\text{increment}, \text{copy}\}$.

- Create $\text{Return_From_Digit1_Read_Digit2}(\langle \text{ReturnD1ReadD2}, op \rangle, \langle \text{DigitReader}, 2, \lambda, op \rangle)$ from the general gadget in Figure 12a
- Create $\text{Return_From_Digit1_Read_Digit2_Case2}(\langle \text{ReturnD1ReadD2} - \text{Case2}, op \rangle, \langle \text{DigitReader}, 2, \lambda, op \rangle)$ from the general gadget in Figure 12b
- Create $\text{Return_From_Digit2_Read_Digit3}(\langle \text{ReturnD2ReadD3}, op \rangle, \langle \text{DigitReader}, 3, \lambda, op \rangle)$ from the general gadget in Figure 12c
- Create $\text{Return_From_Digit3_Read_Digit1}(\langle \text{ReturnD3ReadD1}, op \rangle, \langle \text{DigitReader}, 1, \lambda, op \rangle)$ from the general gadget in Figure 12d

2.5.6 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.

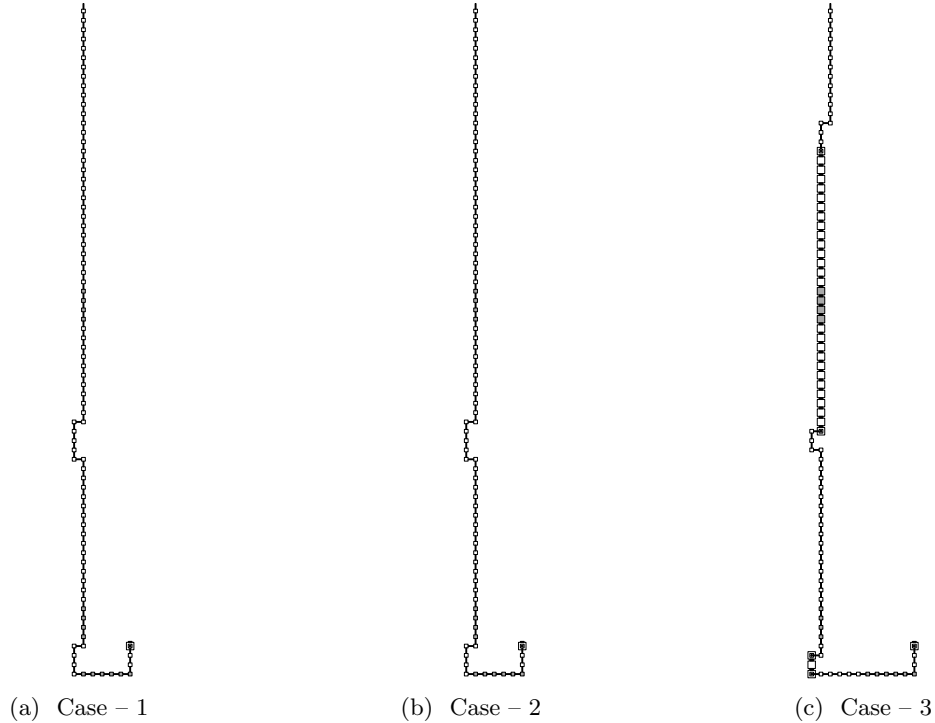
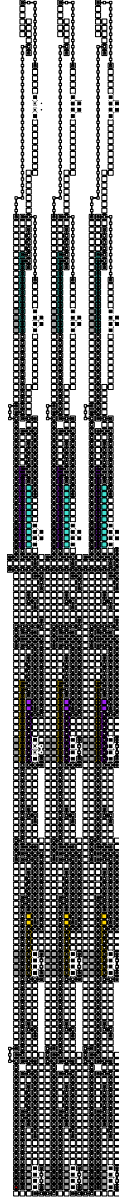


Figure 13: $\text{Return_From_Digit_Read_Next_Row}$ gadgets. All of these gadgets assemble north to south. The vertical gray lines tiles have a height that depends on l and the horizontal gray lines depend on k . (cases 1 and 2 are geometrically equivalent)

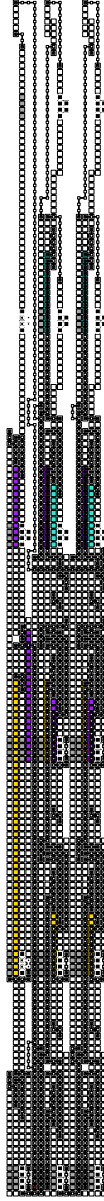
For each $op \in \{\text{increment}, \text{copy}\}$

- Create `Return.From.Digit1.Read.Next.Row`(`ReturnD1ReadNextRow, op`), $\langle \text{DigitReader}, 1, \lambda, op \rangle$) from the general gadget in Figure 13a
- Create `Return.From.Digit2.Read.Next.Row`(`ReturnD2ReadNextRow, op`), $\langle \text{DigitReader}, 1, \lambda, op \rangle$) from the general gadget in Figure 13b
- Create `Return.From.Digit3.Read.Next.Row`(`ReturnD3ReadNextRow, op`), $\langle \text{DigitReader}, 1, \lambda, op \rangle$) from the general gadget in Figure 13c

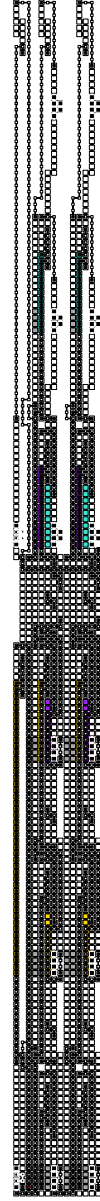
2.6 Overviews



(a) Full overview case 3



(b) Full overview case 2



(c) Full overview case 1