University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

# ECE 120: Introduction to Computing

## Caring About Don't Cares and Glue Logic

## Let's Have Some Ice Cream!

Let's build something.

Anyone like ice cream?

Let's build an ice cream dispenser.

Mango and pistachio!

Ice cream for photos courtesy of Annapoorna Stores

## Start by Specifying the Inputs and Outputs

inputs: **three buttons**
- **M**(ango): 1 when it's pushed
- **B**(lend): 1 when it's pushed
- **P**(istachio): 1 when it's pushed

outputs: **two 2-bit unsigned numbers**
- **$C_M$[1:0]**: number of ½ cups of mango
- **$C_P$[1:0]**: number of ½ cups of pistachio

## The User Has Three Choices (and One Non-Choice)

Help fill in the truth table…

Push M, get one cup of mango.

Push B, get ½ cup of each.

Push P, get one cup of pistachio.

Push nothing, get nothing.

| M | B | P | $C_M$ | $C_P$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 00 | 00 |
| 0 | 0 | 1 | 00 | 10 |
| 0 | 1 | 0 | 01 | 01 |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | 10 | 00 |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

## Fill the Rest with Don't Cares

**What about the rest?**

Who cares?

**Fill with x's.**

| M | B | P | $C_M$ | $C_P$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 00 | 00 |
| 0 | 0 | 1 | 00 | 10 |
| 0 | 1 | 0 | 01 | 01 |
| 0 | 1 | 1 | xx | xx |
| 1 | 0 | 0 | 10 | 00 |
| 1 | 0 | 1 | xx | xx |
| 1 | 1 | 0 | xx | xx |
| 1 | 1 | 1 | xx | xx |

## We Need to Solve for Each Output Bit

Now we can copy to K-maps. First, $C_M[1]$.

| $C_M[1]$ | | BP | | |
|---|---|---|---|---|
| M | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | x | 0 |
| 1 | 1 | x | x | x |

**$C_M[1] = M$**

| M | B | P | $C_M$ | $C_P$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 00 | 00 |
| 0 | 0 | 1 | 00 | 10 |
| 0 | 1 | 0 | 01 | 01 |
| 0 | 1 | 1 | xx | xx |
| 1 | 0 | 0 | 10 | 00 |
| 1 | 0 | 1 | xx | xx |
| 1 | 1 | 0 | xx | xx |
| 1 | 1 | 1 | xx | xx |

## Solve the Low Bit of Mango

Next, $C_M[0]$.

| $C_M[0]$ | | BP | | |
|---|---|---|---|---|
| M | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | x | 1 |
| 1 | 0 | x | x | x |

**$C_M[0] = B$**

| M | B | P | $C_M$ | $C_P$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 00 | 00 |
| 0 | 0 | 1 | 00 | 10 |
| 0 | 1 | 0 | 01 | 01 |
| 0 | 1 | 1 | xx | xx |
| 1 | 0 | 0 | 10 | 00 |
| 1 | 0 | 1 | xx | xx |
| 1 | 1 | 0 | xx | xx |
| 1 | 1 | 1 | xx | xx |

## Solve the High Bit of Pistachio

And $C_P[1]$.

| $C_P[1]$ | | BP | | |
|---|---|---|---|---|
| M | 00 | 01 | 11 | 10 |
| 0 | 0 | 1 | x | 0 |
| 1 | 0 | x | x | x |

**$C_P[1] = P$**

| M | B | P | $C_M$ | $C_P$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 00 | 00 |
| 0 | 0 | 1 | 00 | 10 |
| 0 | 1 | 0 | 01 | 01 |
| 0 | 1 | 1 | xx | xx |
| 1 | 0 | 0 | 10 | 00 |
| 1 | 0 | 1 | xx | xx |
| 1 | 1 | 0 | xx | xx |
| 1 | 1 | 1 | xx | xx |

## Solve the Low Bit of Pistachio

And, finally, $C_P[0]$.

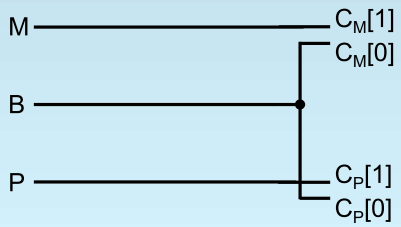
| $C_P[0]$ | BP 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| M 0 | 0 | 0 | x | 1 |
| 1 | 0 | x | x | x |

$C_P[0] = B$

| M | B | P | $C_M$ | $C_P$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 00 | 00 |
| 0 | 0 | 1 | 00 | 10 |
| 0 | 1 | 0 | 01 | 01 |
| 0 | 1 | 1 | xx | xx |
| 1 | 0 | 0 | 10 | 00 |
| 1 | 0 | 1 | xx | xx |
| 1 | 1 | 0 | xx | xx |
| 1 | 1 | 1 | xx | xx |

## The Solution Requires No Gates!

The don't cares made the functions so simple that **we don't even need any gates**!



## What if a User Pushes Two Buttons?

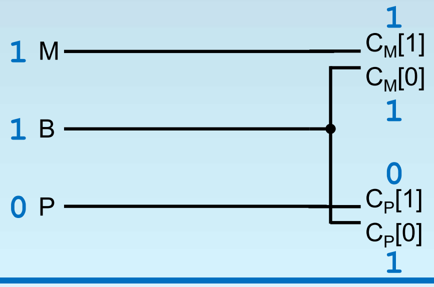
Let's be careful.

What happens if a user **presses M and B** at the same time?

**1.5 cups of mango**

AND

**0.5 cups of pistachio!**



## Don't Cares: Not for Human Behavior!

In the best case, the cup overflows (2 cups of ice cream instead of 1 cup).

In the worst case,

- the engineer of the mechanical system
- assumed that we would not send 11, and
- something worse happens when we do.

**So we DO care.**

Generally, **using don't cares when humans are involved is a bad idea**.

# Let's Clean Up the Inputs

**How can we fix the problem?**

One approach:

- **choose specific outputs** for each combination of inputs,
- then solve the K-maps again.

Another approach:

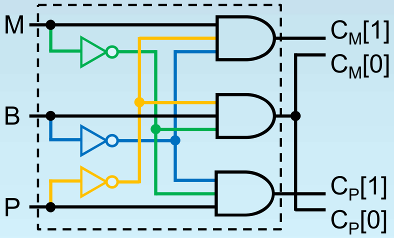
- clean up the inputs with **more logic**
- **prevent** humans from ever producing **bad combinations**.

# Use Glue Logic to Ensure that Assumptions Hold

For example, we can force all inputs to zero if the human presses more than one button.

Each of the AND gates produces a 1 iff the corresponding button is the ONLY 1 entered.

**glue logic (dashed box)**



# The Inputs Can be Cleaned Up in Many Ways

Forcing invalid input combinations to zero is just one strategy.

We could also choose a priority on the buttons (six possible choices).

For example:

- Pistachio overrides other buttons, and
- Mango overrides Blend.

Or use a combination of approaches.

# What About Picking Specific K-Maps?

In the case of our ice cream dispenser and the strategy shown, the two approaches are the same (just remove the dashed box!).

In general, however,

- these approaches **vary**
- **in area, speed, and/or power**.

Cleaning up the inputs is perhaps **easier to understand**.