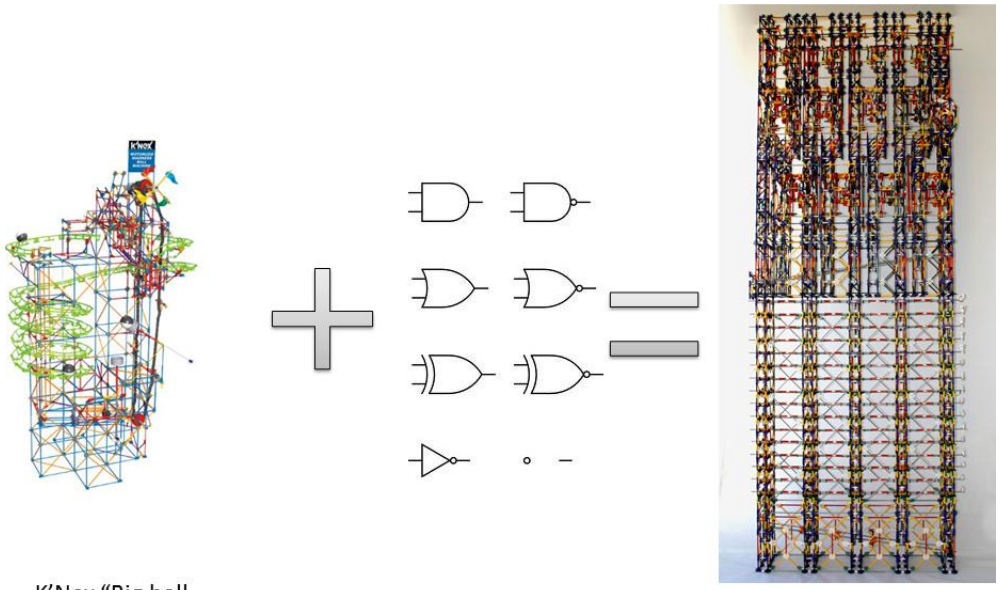
DIFFERENTIAL BOOLEAN CALCULUS USING K'NEX

EVAN MORIKAWA AND STEVEN ZHANG: COMPUTER ARCHITECTURE FALL 2010

INTRODUCTION

What do you get when you cross a K'Nex toy with Boolean logic gates? A K'Nex calculator! (Figure 1).



K'Nex "Big ball machine" toy

Logic gates

K'Nex calculator

Figure 1 Sources: K'Nex toy image from Amazon.com. K'Nex calculator toy image from http://knexcomputer.blogspot.com/

Our team set out to build a D-flip flop simulated by a complex K'Nex ball machine[1], but ended up building a differential AND gate instead. Along the way, we learned the theory behind Boolean calculus and how it applied to our design. Before going any further, we encourage the reader to watch a demonstration video of our calculator, which can be found here: http://www.youtube.com/watch?v=sX1uEp02i_Q

HOW WE'RE DIFFERENT

Donahoe et. al's K'Nex calculator used ball position to represent the state of a logic gate (1). Essentially, this means that the state of the machine is encoded only in the position of the ball. Once the ball leaves the machine, only the final answer remains, while any intermediate state is lost.

For example, here is a schematic of their AND gate:

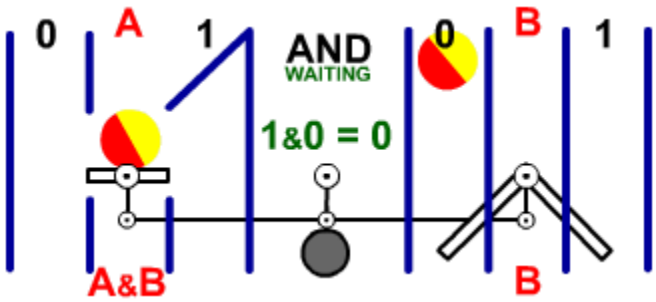


Figure 2 Donahoe et al. used ball position to represent state of a logic gate, in this case an AND gate. Image from: http://serioussandbox.com/media/knexcomputer/images/and_wait.gif

In fact this is the case with most of the ball-based mechanical adders in existence today[2]. There has even been informal research done on “ball logic gates” (2).

While ball logic machines are convenient to build and show Boolean logic in a physical manner, there is one downside: it's very hard to see the change in state over time of a gate. To remedy this problem, we chose to use balls to represent changes in state of the machine. State was encoded into the ball machine itself. That way, a ball falling through the machine would represent a signal propagating through a logic gate in real time. Later, we discovered that we were really using the balls to represent impulses in Boolean differential calculus.

Before we delve into the mechanical implementation of our machine however, we'll delve into the theory behind Boolean differentiation.

---

[1] A K’Nex ball machine is a complex K’Nex construction that uses the kinetic energy of falling balls to power some sort of complex behavior in the system

[2] For example, see the ones here: http://www.youtube.com/watch?v=jrT94IwCo74, http://www.youtube.com/watch?v=SYi9sJkS19Q

BOOLEAN DIFFERENTIATION THEORY

Given a Boolean function $f(x_0, x_1, \ldots x_i \ldots x_{n-1})$, what is its partial derivative $\frac{\partial f}{\partial x_i}$ which also is a Boolean function?

Using intuition from calculus over the real numbers, one would suspect $\frac{\partial f}{\partial x_i}$ should be zero if x_i changes, but f does not. Since Boolean algebra is defined only over the numbers $\{0,1\}$, it is reasonable to assume that in the other case- if x_i changes, and f does, $\frac{\partial f}{\partial x_i}$ should be 1. In fact, this is exactly the intuition built into the formal definition of $\frac{\partial f}{\partial x_i}$.

Definition 1. Let $f(\vec{x})$ be a logic function of n variables, where $\vec{x}$ is the vector of input variables. Also let $(\vec{x}) = f(x_i, \vec{x}_1)$, where $\vec{x}_1$ contains all the variables of $\vec{x}$ except for x_i. Then

$$\frac{\partial f}{\partial x_i} = f(x_i, \vec{x}_1) \oplus f(\overline{x_i}, \vec{x}_1)$$

is the derivative of f (3).

We can use this definition to calculate the partial derivatives of the Boolean operation AND:

$$\frac{\partial AND(A,B)}{\partial A} = AND(A,B) \oplus AND(\bar{A},B)$$

$$\frac{\partial AND(A,B)}{\partial B} = AND(A,B) \oplus AND(A,\bar{B})$$

AN ASIDE: WHY THIS MATTERS IN THE REAL WORLD

We can use this theory to rid of potential inputs on Boolean functions, which could be useful in logic circuit designs. For example, if $\frac{\partial f}{\partial x_i} = 0$, this means that input $x_i$has no effect on the output f, allowing the circuit to be reduced. Other applications of Boolean differential calculus include multilevel circuit design, error correction codes, and cryptography (3).

IMPLEMENTATION

IDEAL IMPLEMENTATION

Using the two equations above, we can calculate the differential, $dAND(A,B)$.

$$\begin{aligned} dAND(A,B) &= \frac{\partial AND(A,B)}{\partial A}\partial A + \frac{\partial AND(A,B)}{\partial B}\partial B \\ &= [AND(A,B) \oplus AND(\bar{A},B)]dA + [AND(A,B) \oplus AND(A,\bar{B})]dB \end{aligned}$$

We used this equation to determine how our differential AND gate should ideally function (Figure 3). Balls represent impulses, dA and dB , depending on which input wire (really, a chute) it is dropped in to. $\mathrm{dAND(A, B)}$ is

represented by the existence of a ball on the output wire of the AND gate. The states A and B are encoded into the state of the machine- the configuration of the flippers.

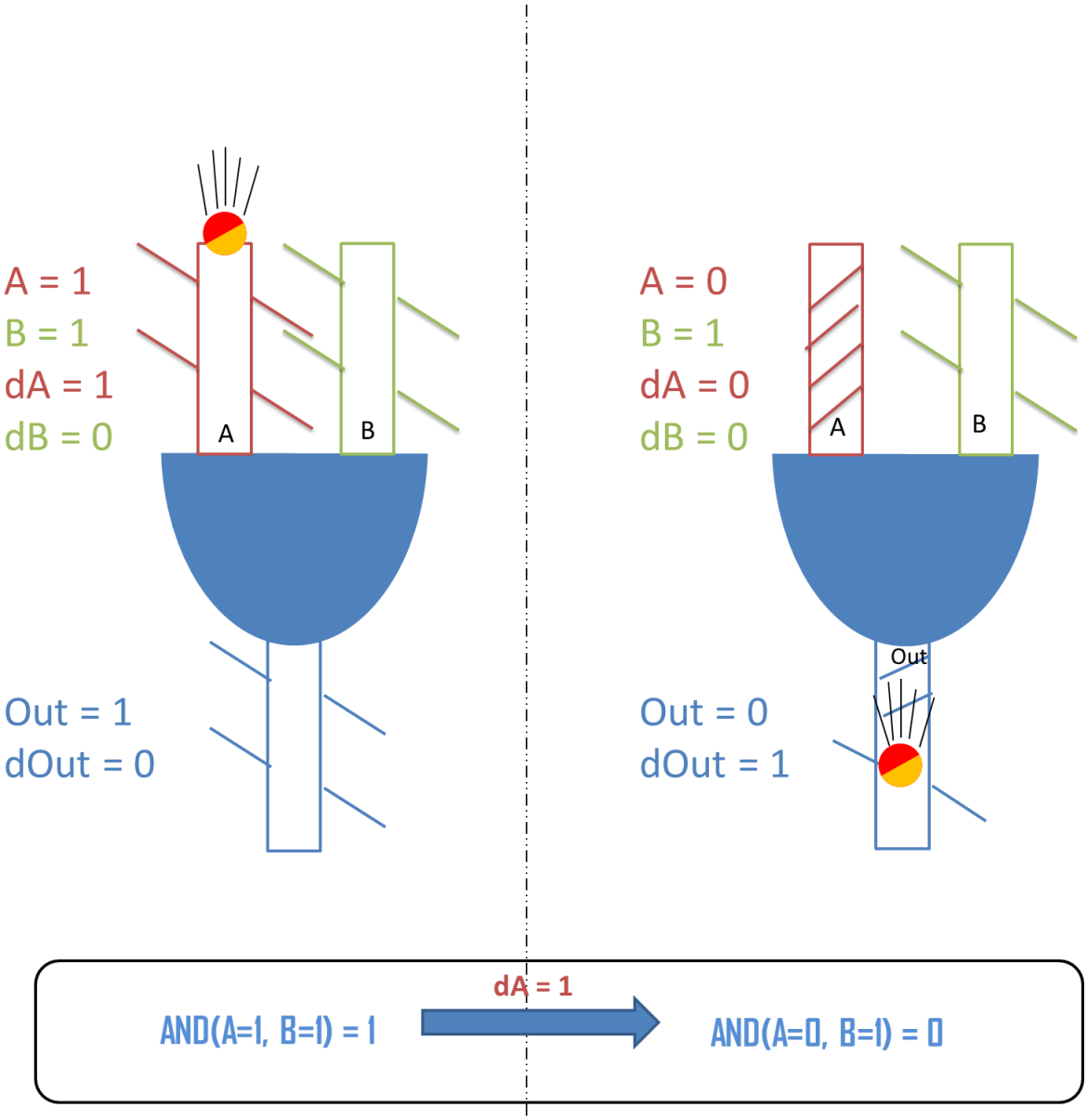


Figure 3 Schematic of our differential AND gate's behavior. Balls are used to represent impulses (dA dB, dOut). State (A, B, Out) is represented by the flippers on each wire.

When a ball is dropped into to the input A "wire", $dA = 1$; at this point, dB should still be zero[3], thus whether the ball drops through the output wire, is determined by the equation

$$dAND(A,B) = \frac{\partial AND(A,B)}{\partial A}\partial A$$

Similar analysis can be determined for when a ball is dropped into the input B wire.

To summarize, this table shows the various states our differential AND gate should be in.

dA	dB	A	B	AND(A,B)	AND(~A, B)	AND(A,~B)	dAND(A,B)/dA	dAND(A,B)/dB	dAND(A,B)
		0	0	0	0		0		0
		1	0	0	0		0		0
		0	1	0	1		1		0
0	0	1	1	1	0	x	1	x	0
		0	0	0	0		0		0
		1	0	0	0		0		0
		0	1	0	1		1		**1**
1		1	1	1	0		1		**1**
		0	0	0		0		0	0
		1	0	0		1		1	0
		0	1	0		0		0	0
0	0	1	1	1	x	0	x	1	0
		0	0	0		0		0	0
		1	0	0		1		1	1
		0	1	0		0		0	0
	1	1	1	1		0		1	1

Figure 4 Truth table of our differential AND gate. Note that dAND(A,B) = 1 only when dA or dB =1.

We've talked about about how our system would represent state. How did we go about implementing it mechanically?

[3] Of course, one could drop both balls at the same time, which would jam the machine. Our mechanical implementation ensures that balls can be dropped within 1s of each other without jamming

MECHANICAL IMPLEMENTATION

STATE IS REPRESENTED BY PIVOTING RAMPS IN THE WIRES

Our wires were a pair of columns, with a series of pivoting ramps. Ramps had rods extending out, to represent state. A ball would travel down one side of the ramp, it would hit the ramp, causing it cross into the other column. Since the ramps were linked, this caused the ramp rods to change state (from extended to contracted, or vice versa). This motion of the rods represents the change in state every time a ball, representing an impulse, is dropped through the wire.

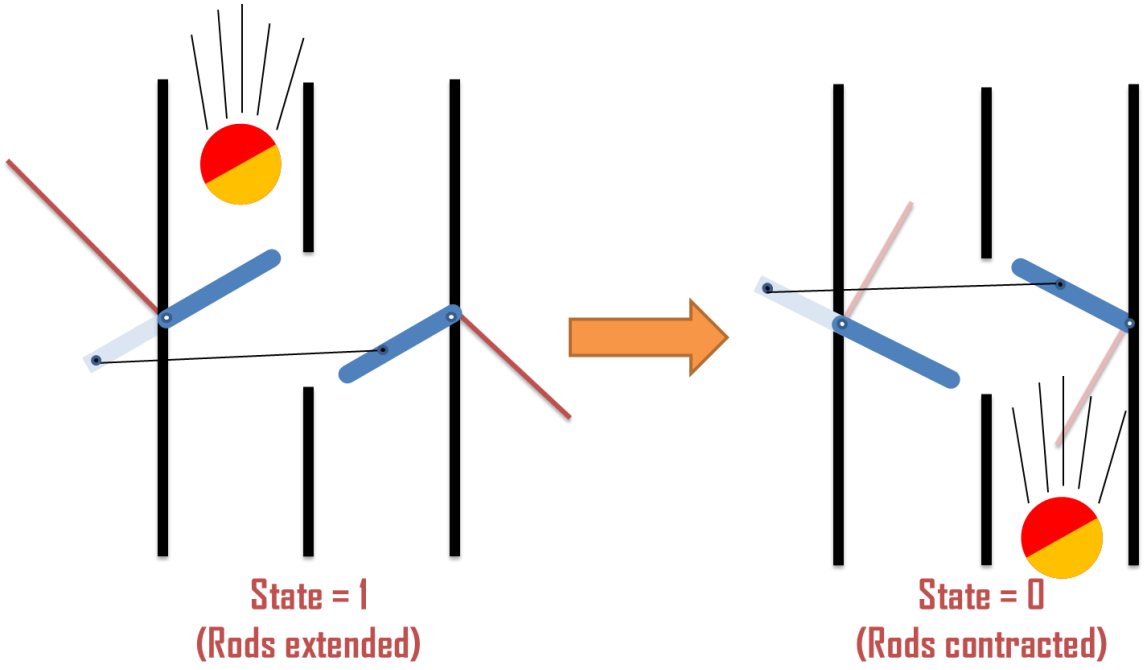


Figure 5 Schematic of how we used linked ramps to represent state. As the ball hits the ramp in its column, it moves over to the next column, and, in the process, changed the state of the rods from 1 to 0. The same process would happen in reverse, if the ball was dropped on the lefthand column

By stacking a series of these pivoting ramps together, we were able to generate a cascade of swinging rods as the ball falls through the wire. This represents the propogation of a signal through a wire.

One more thing to note. The column through which the ball falls is actually determined by the state of the wire. In the above example, the ball will always exit through the left column if the state of the wire remains or changes to 1; it will always exit the right column if the state of the wire remains or changes to 0. We use this effect, which we call the “column-state effect” to generate our logic functionality, described below.

WE USED A SERIES OF LINKED LEVERS TO GENERATE THE DESIRED LOGIC FUNCTIONALITY.

Once the wires were designed, the other part of the mechanical design was the actual AND gate. To generate the complex behavior shown in the truth table of Figure 4, we once again turned to linked levers.

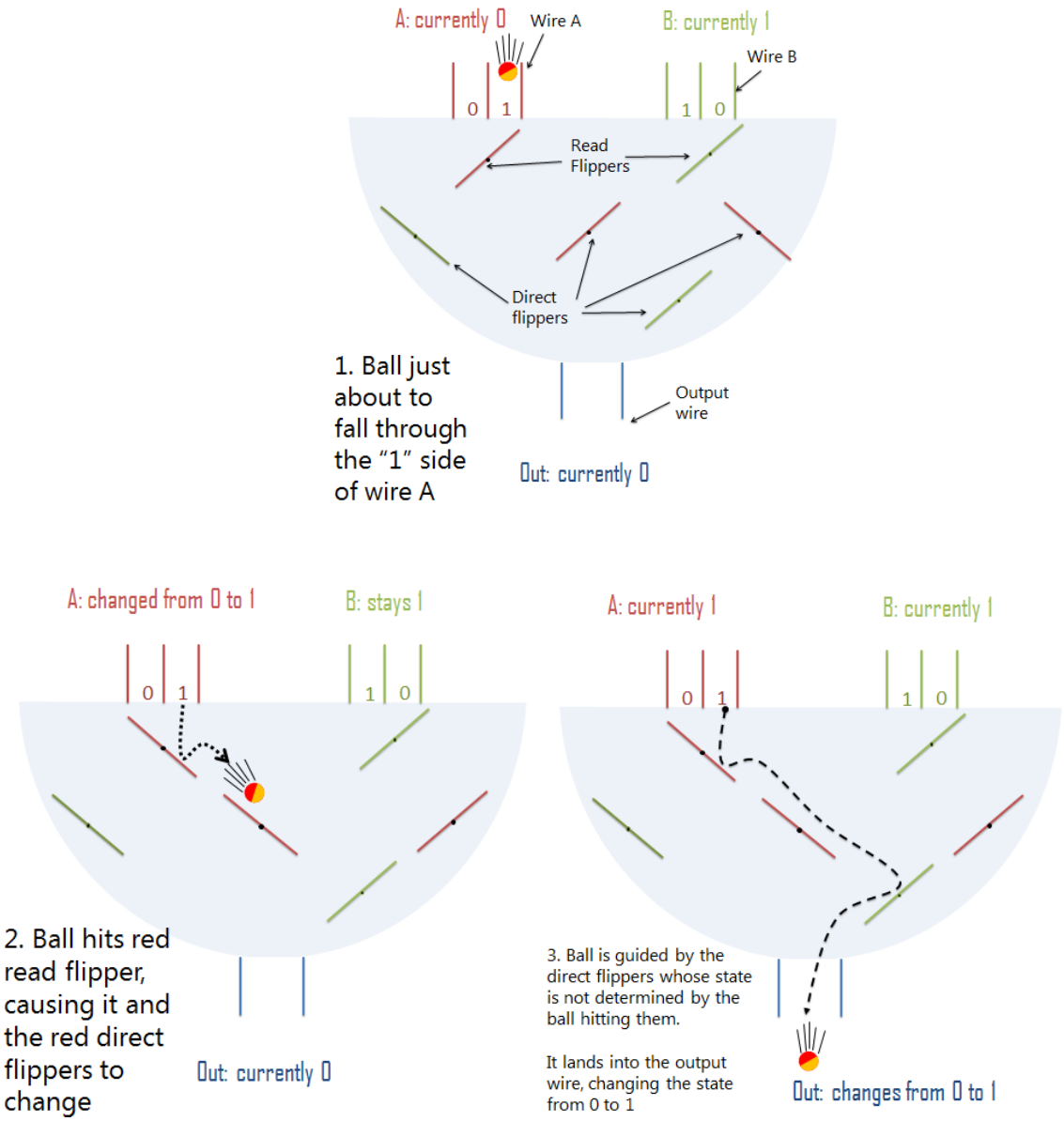


Figure 6 Schematic showing how the ball is directed appropriately by a series of "read" and "direct" flippers

Using the column-state effect described above, we used “read” flippers below each input wire to read the state of the wire, and rotate the “direct” flippers appropriately. The exact mechanics of how these linkages are beyond the scope of the paper. Figure 6 shows how this works, schematically, step by step.

Figure 7Figure 7 shows two more examples of how the ball is directly appropriately so that the truth table in Figure 4 is simulated. We leave it up to the reader to verify that the other 5 conditions work as well.

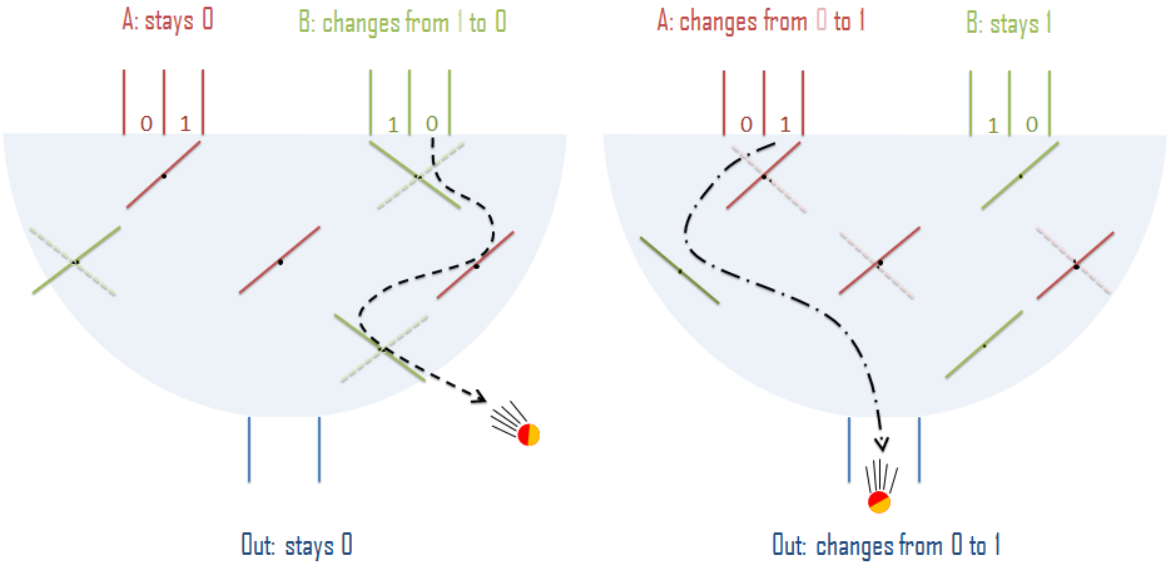


Figure 7 Some other examples of input conditions our AND gate handles, and how it directs the ball accordingly

OUR FINAL PRODUCT

By connecting our AND gate with two input wires, and one output wire, we were able to construct a full differential AND gate show in Figure 8. The reader can now rewatch the full demo of our machine work, this time with a deeper understanding of the machine[4]: http://www.youtube.com/watch?v=sX1uEp02i_Q .

[4] Or you can come to the Olin College Library, basement floor, where the machine will reside until someone decides to use its pieces to build something bigger

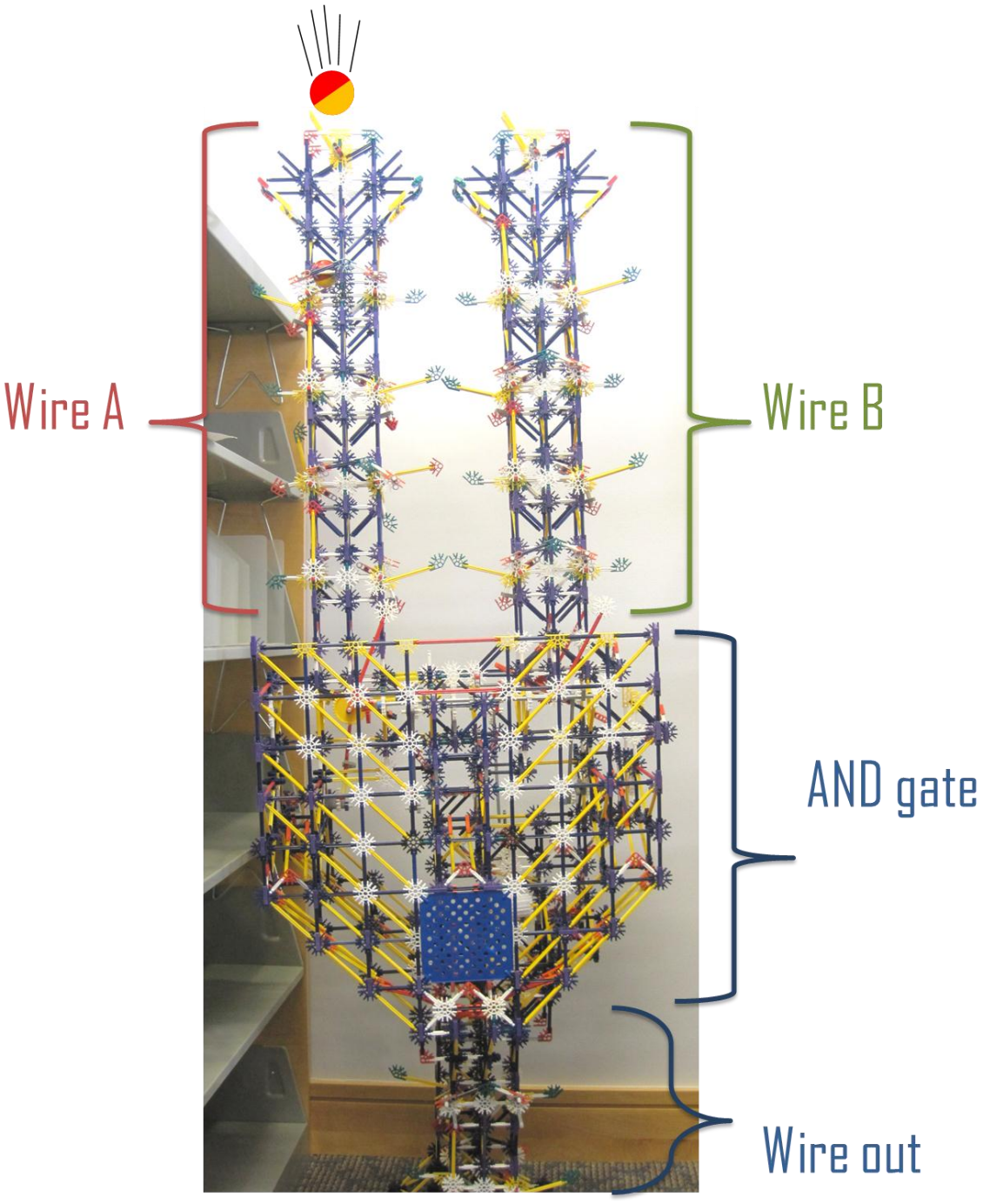


Figure 8 Our final ball machine: A K'Nex differential AND gate

SOME LESSONS

Through the process of simulating computation with a mechanical computer, we learned several key lessons.

IT TAKES SEVERAL PROTOTYPES TO MAKE SOMETHING WORK

Even though the analogues between the physical and the electronic may exist in theory, when it comes to figuring out exactly which Knex piece attaches to which other Knex piece, these analogues suddenly become less useful. In thinking about how to represent the state of the machine, we had to first figure out how Knex all fit together at a fundamental level. For example, some rods are root-2 factors different in length so they can be used as square diagonals. Some rods are even multiple dimension or connecting pieces and some are not.

The first experiment was how to represent the state of the wire as a binary-flipping device. To do this we experimented with several different techniques. One used a ratcheting wheel that spun a ¼ turn each time a ball dropped. Another used a set of interlocking gears to catch the various states of the ball. The final design used a set of flip-flopping levers to keep track of the state. Figure 9 shows a set of images that illustrate some of the early concepts for a segment of one of our wires.

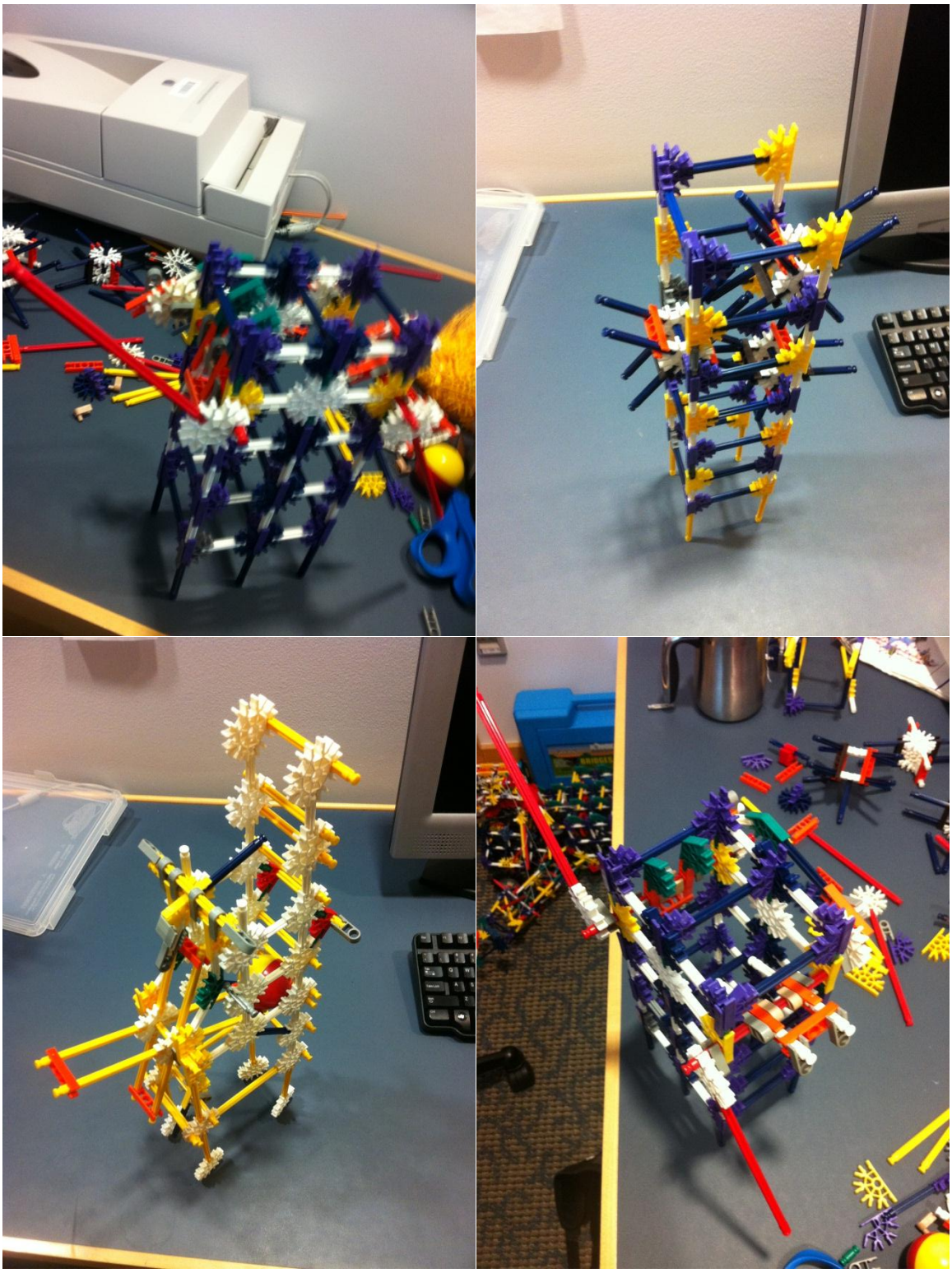


Figure 9 Early prototypes of representing state in wires.

QUIRKS OF THE PHYSICAL WORLD HAMPER EFFORTS AND ROBUSTNESS

One major issue we were not expecting was unexpected behavior from our devices due to real-world physical effects. For example, we had to spend a very large amount of time carefully balancing the system to prevent bias in our levers for the wires. We had to accommodate for natural flexing in the K'Nex pieces that would cause what we thought were basic square to malform. Very troublesome was an inherent elastic bounce when pieces hit each other. This bounce would cause levers to enter invalid states. Fixing all of these problems took a large chunk of the time and led us to begin implicitly categorizing various types of configurations as more robust than others.

THINGS IN MECHANICAL LAND ARE MUCH BIGGER THAN THINGS IN ELECTRICAL LAND

The K'Nex AND gate ended up being much larger than expected. With its total dimensions on par with that of a door, this single gate is far larger than its electronic counterpart. To put this into perspective, begin with the approximate dimensions of our AND gate being about 2m in height by 1m in width and 0.5m in depth. Now, consider the size of an AND gate on the latest line of microprocessors. Assuming a feature size of 32nm like those on the latest Intel i7 processor, and assuming this creates an equivalent gate within about 100nm, then the chip is about 10 billion times smaller on a side.

If one assumed that with the Core i7's 731 million transistors one could make 243 million AND gates, and one assumed the stated dimensions of our Knex AND gate, then to build a Knex "Chip" of equivalent transistor count to the Intel i7 would require an area of about 486 square kilometers, which is about 1/6 the size of Rhode Island.

TESTBENCHES ARE IMPORTANT!

When the machine was small, we were able to work with it on a table or in our hands. However, we quickly realized how large the device had to be. Furthermore, in order to make the levers robust enough, we had to add an additional dimension to our design. The project began requiring extensive work in all 3 dimensions and it was growing large and heavy.

To accommodate this, we had to build a separate structure to support the device on. In Figure 10, one can see the trussed span that was used to suspend the device in mid-air. With this setup we were able to work comfortably on the device and build it from any angle.

This was a unique experience in the need for a physical implementation of a test bench. In many electronics and computer projects one builds test-benches and test-suites. Here was that practice manifesting itself in physical form.

Figure 10 A truss was needed to facilitate construction of our ball machine

WORKS CITED

1. **Donahoe, Matt and Decew, Jeff.** Ball theory. *The K'NEX Computer.* [Online] January 2006. [Cited: December 15, 2012.] http://knexcomputer.blogspot.com/2007/01/how-it-works-part-1.html.

2. **Black Phoenix.** Ball Logic Gates. *Black Phoenix's brain blog.* [Online] May 19, 2010. [Cited: December 15, 2010.] http://brain.wireos.com/?p=2215.

3. *The Boolean Differential Calculus - Introduction and Examples.* **Steinbach, B and Posthoff, Ch.** Naha, Okinawa, Japan : s.n., 2009, Proceedings – Reed-Muller Workshop, pp. 107-117. http://www.informatik.tu-freiberg.de/prof2/publikationen/RM2009_bdcie.pdf.

THANKS

Special thanks go to our professors Mark Sheldon, and Alex Morrow for allowing us to build a mechanical engineering project in a computer architecture class. Also, thanks to Dee Magnoni of the Olin College library for allowing us to borrow K'Nex to build our machine. Finally, thanks to Matt Donahoe and Jeff Decew, Olin College alums, for inspiring us with their K'Nex calculator every time we walk into the Olin College library.