Leva Bilim Topluluğu
2025 Leva Matematik ve Kültür Kampları
Genel Yetenek Sınavı

*Metin 2*

# HER ŞEYDEN BİLGİSAYAR YAPABİLİRSİNİZ

1960 yılında IBM 650'nin Karayolları Genel Müdürlüğüne kurulmasıyla birlikte Türk insanının hayatına giren *computer*, Türkçeye ise hâlâ girememişti. Çünkü Türkçe, sayılarla işlem yapan *calculator* adlı makineye "hesap makinesi" diyerek erken davranmış ve "computer" için kelime çevirisi yapma hakkını kaybetmişti. 1969'da Prof. Dr. Aydın Köksal'ın *bilgisayar* kelimesini icat etmesiyle çözülen bu sorun, dilimizin yetersiz kaldığı bir noktaya işaret ediyor: Biz Türkler *calculation* ve *computation* kavramlarını bir tutuyoruz. Bu metinde "hesaplama" dediğimizde ikincisini kastediyor olacağız.

## Hesaplama nedir?

Tanıdık bir örnekle başlayalım: Kullandığımız cihaz ister bir bilgisayar klavyesi isterse daktilo olsun, bastığımız tuşa göre bir kağıdın nasıl boyanacağına cihazın karar verme süreci bir hesaplamadır. Misal, bilgisayar ekranına yazdığımız semboller $16 \times 16$ çözünürlüğünde ise, klavyede bastığımız tuşun ekrana yansıması için bilgisayarın yaptığı işlem, bu 256 pikselin her biri için bir karar vermektir: boyanmak ya da boyanmamak. Bilgisayarımız bu şekilde bize $2^{256}$ farklı çıktı verebilir!

Bu çıktılara numara verelim: $0, 1, 2, \ldots, 2^{256} - 1$. Benzer şekilde, bilgisayara verebileceğimiz girdileri, yani yazı yazarken kullanacağımız tüm sembolleri numaralandıralım. Misal, bu şekilde 1000 adet sembolümüz varsa, kullandığımız sembollerin kümesini $\{0, 1, \ldots, 999\}$ kümesi olarak düşünelim. Bu durumda bilgisayarımızın yaptığı şey aslında bir fonksiyonu gerçek hayata dökmek olur: $i$ numaralı sembolün ekrandaki çıktısının numarası $f(i)$ olacak şekilde tanımlanan $f : \{0, 1, \ldots, 999\} \to \{0, 1, \ldots, 2^{256} - 1\}$ fonksiyonu.

Bu örnekten hareketle *hesaplama* kavramına genel bir tanım verebiliriz: Bir makinemiz olsun. Makineye verebileceğimiz girdilerin kümesine $X$, makinenin verebileceği çıktıların kümesine $Y$ diyelim. $f : X \to Y$ ise hangi girdiye karşılık olarak hangi çıktıyı almak istediğimizi anlatan fonksiyon olsun. Makinenin herhangi bir $x \in X$ verildiğinde $f(x)$'i sonlu zaman içerisinde bulmasını sağlayan sürece **hesaplama**, bu süreci tanımlayan kurallar dizisine ise **algoritma** denir.

## Hesaplanabilirliğin sınırları

Alan Turing, 1936 tarihli *Hesaplanabilir Sayılar Üzerine* isimli makalesinde "hesaplama" kavramının matematiksel olarak formal bir tanımını kurdu (yukarıdaki tanımın ne kadar informal olduğuna dikkatinizi çekerim) ve bu tanıma dayanarak belli başlı bazı fonksiyonların hesaplanamayacağını ispatladı. Günümüzde bu çalışma *halting problem* adıyla geçer.

Şanslıyız ki Turing'in bu çalışmasına rağmen girdi kümesi ($X$) ve çıktı kümesi ($Y$) sonlu olan herhangi bir $f : X \to Y$ fonksiyonu verildiğinde $f$'in hesaplanmasını sağlayan bir algoritma bulunabiliyor. Bu bölümde bunun nasıl her zaman mümkün olduğunu öğreneceğiz.

Başlamadan önce bir parantez açalım: Gerçek dünyada bir makinenin aldığı girdiler de verdiği çıktılar da sonlu sayıda olmak zorundadır. Birazdan anlatacağımız fikirler modern bilgisayar mimarisinin temelinde yatmaktadır. Fakat bu durum, sonsuz girdisi ve çıktısı olan makinelerin insan zihninde bir karşılığı olmadığı anlamına gelmez. En basitinden, herhangi bir $(a, b)$ tam sayı ikilisini girdi alıp $a+b$ sayısını çıktı veren bir toplama makinesi düşünelim. Bu makinenin girdileri ve çıktıları sonsuz çokluktadır. Dolayısıyla böyle bir makineyi inşa etmek için birazdan anlatacağımız yöntemleri kullanamayız (**Problem 2.1**).

Önceki bölümde klavye için yaptığımıza benzer şekilde, bize verilen herhangi bir makinenin girdilerini ve çıktılarını 0'dan başlayarak numaralandıralım. Böylece, $s$ ve $t$ doğal sayılar olmak üzere bu makineyi bir $f : \{0, 1, \ldots, s\} \to \{0, 1, \ldots, t\}$ fonksiyonu şeklinde düşünebiliriz. Ardından bu doğal sayıları 2 tabanında yazalım. $s < 2^n$ ve $t < 2^m$ koşullarını sağlayan $n$ ve $m$ doğal sayılarını alırsak, $f$'in her girdisini 2 tabanında $n$ basamaklı bir sayı olarak, her çıktısını da 2 tabanında $m$ basamaklı bir sayı olarak yazabiliriz.

Örnek olarak $\{0, 1, 2, 3\}$ kümesinden bir $(a, b)$ ikilisini alıp $a + b$ sayısını çıktı olarak veren küçük bir sonlu toplama makinesini ele alalım. Bu makinenin girdilerini aşağıdaki gibi numaralandıralım:

| | | | |
|---|---|---|---|
| $0 \to (0,0)$ | $4 \to (1,0)$ | $8 \to (2,0)$ | $12 \to (3,0)$ |
| $1 \to (0,1)$ | $5 \to (1,1)$ | $9 \to (2,1)$ | $13 \to (3,1)$ |
| $2 \to (0,2)$ | $6 \to (1,2)$ | $10 \to (2,2)$ | $14 \to (3,2)$ |
| $3 \to (0,3)$ | $7 \to (1,3)$ | $11 \to (2,3)$ | $15 \to (3,3)$ |

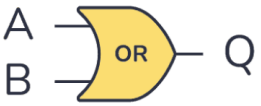
İkilik tabanda bu fonksiyonun tüm girdilerini 4 haneyle, tüm çıktılarını 3 haneyle gösterebiliriz. Bunun sonucunda aşağıdaki $16 \times 7$ boyutlarındaki tabloyu elde ederiz:

| **girdinin haneleri** | | | | **çıktının haneleri** | | |
|---|---|---|---|---|---|---|
| $a_1$ | $a_0$ | $b_1$ | $b_0$ | $s_2$ | $s_1$ | $s_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

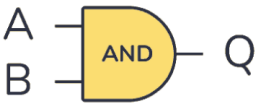
Sizce de bu lisede gördüğümüz mantık tablolarına çok benzemiyor mu? İlk dört satırda dört farklı önermenin alabileceği tüm doğruluk değeri kombinasyonlarına yer verilmiş, son üç satırda da bu önermelerin belli mantıksal kombinasyonlarının doğruluk değerlerine yer verilmiş. Temel fikir bundan ibaret: Her sonlu hesaplamayı bir doğruluk tablosuna dönüştürebiliriz. O hâlde tüm doğruluk tablolarını hesaplamanın bir yolunu bulursak amacımıza ulaşırız.

## Mantık kapıları


| A | Q |
|---|---|
| 0 | 1 |
| 1 | 0 |



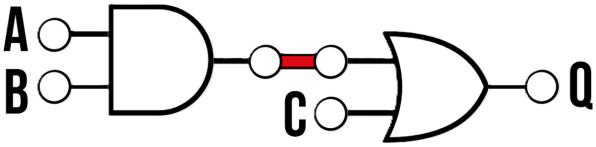

| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

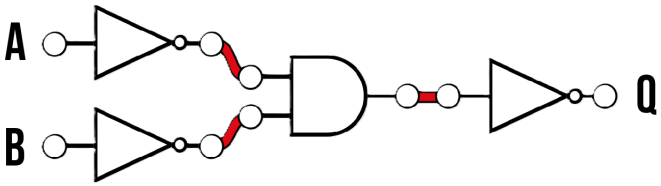



| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Tüm olası doğruluk tablolarını elde etmek için birkaç küçük tabloyu temel yapı taşı olarak kullanacağız ve bu temel tablolara **kapı** adını vereceğiz. Yukarıda, lisedeki mantık derslerinden aşina olduğunuz üç mantık yapısını görüyorsunuz: önermenin değilini almak (NOT kapısı), $\vee$ bağlacı (OR kapısı) ve $\wedge$ bağlacı (AND kapısı).

Bu kapıları yukarıdaki sembollerle göstererek bileşik önermeleri şema hâline getirebiliriz. Örneğin aşağıda gördüğünüz şemalardan soldaki **(1)** önermesini ifade ederken, sağdaki ise **(2)** kurallarına göre OR kapısının yaptığı işi yalnızca NOT ve AND kapılarını kullanarak yapan bir düzeneği gösterir.




Öyleyse tüm doğruluk tablolarını elde etme yolunda OR kapısına ihtiyacımız olmayacak, çünkü herhangi bir şemada OR kapısını sağdaki düzenekle değiştirebiliriz. Örneğin soldaki şemada bunu yaparsak **(1)** önermesinin doğruluk tablosunu sadece NOT ve AND kapılarıyla ifade etmiş oluruz. Acaba bu iki kapı, önceki sayfada yer alan devasa toplama tablosu için de yeterli olur muydu?


| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

XOR kapısı ile tanışın. Bu kapı da yalnızca NOT ve AND'dan oluşan bir şemaya dönüştürülebilir (**Problem 2.3**) ve yaptığı iş toplamaya çok benzer: $(1,1)$ ikilisini 0 yerine $(1,0)$ gibi bir ikiliye götürme imkânı olsaydı $\{0,1\}$ kümesinden alınan herhangi iki sayıyı toplama işlemini yalnızca XOR ile yapabilirdik. Öyleyse $A+B$'yi bulmak için XOR çıktısının başına bir hane daha koyalım. Bunu yapmak çok basit: Başa konacak haneyi **(3)** kapısı veriyor! Dolayısıyla aşağıdaki şemada gösterdiğimiz düzenek, ikilik tabanda bir basamaklı $A$ ve $B$ sayılarını girdi alıp iki basamaklı $A+B$ sayısının hanelerini iki ayrı çıktı olarak verir:

**(4)**

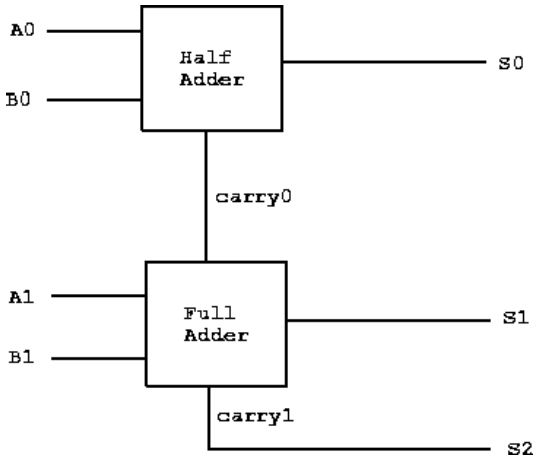
Bu düzeneğe bir *yarı toplayıcı (half adder)* denir.

Metnin geriye kalanında "ikilik tabandaki bir hane" yerine **bit** kelimesini kullanacağız. Örneğin az önce 2 bitin toplanması sürecine karşılık gelen mantıksal önermenin şemasını çizdik. Peki ya 2 yerine 3 adet biti toplayan bir düzeneği nasıl kurabiliriz? Aşağıdaki şema tam olarak bunu yapıyor: $A, B$ ve $C_{in}$ bitlerini girdi alıp $A + B + C_{in}$ sayısının birler basamağını $S$, ikiler basamağını $C_{out}$ adıyla çıktı veriyor.

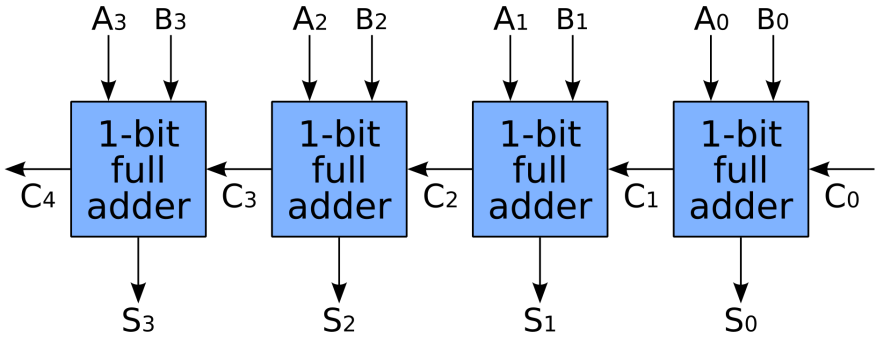
**(5)**

Bunun adı da *tam toplayıcı (full adder)*. Şemada iki adet yarı toplayıcı ve bir adet OR kullandık. Yarı toplayıcıların içindeki XOR'ları ve dışarıdaki OR'u NOT ve AND'lerden oluşan karşılıklarıyla değiştirirsek 3 bitin toplanması sürecini de yalnızca NOT ve AND kullanarak tanımlamış oluruz.

Artık bir önceki bölümde yer verdiğimiz 2-bit toplama işlemi tablosunu şematik olarak göstermek için hazırız. İkilik tabanda $a_1a_0$ ve $b_1b_0$ şeklinde yazılan sayıları alt alta yazıp toplarken önce $a_0 + b_0$'ı hesaplıyor, birler basamağını $s_0$ hanesine yazıyor ve ikiler basamağını elde taşıyoruz ($carry0$). Bunu bir yarı toplayıcı ile yapabiliriz. Geriye kalan şey $a_1, b_1$ ve $carry0$ bitlerini toplamaktır. Bunu da bir tam toplayıcı ile yapabiliriz. Böylece yandaki şema ortaya çıkar.




Unutmayın: Bu düzenek mantıksal bir önermenin şematik gösteriminden ibaret! Böylece $\{0, 1, 2, 3\}$ kümesi üzerindeki toplama işlemini yalnızca NOT ve AND bağlaçlarından oluşan bir önermenin doğruluk tablosuna dönüştürmüş olduk.




Tam toplayıcıları yukarıdaki gibi uç uca ekleyerek elde taşıma mantığını devam ettirebiliriz. Böylelikle, topladığımız sayıların maksimum bit sayısını önden bilmek şartıyla, her türlü toplama işlemini yalnızca NOT ve AND kapılarıyla hesaplayabiliriz. Fakat çok daha fazlası mümkün: Sadece toplama işlemi değil, oluşturulabilecek tüm mantık tabloları yalnızca bu iki kapı kullanılarak inşa edilebilir. Başka bir deyişle: $\{NOT, AND\}$ kümesi **fonksiyonel olarak tamamlanmıştır.** Bu durum kulağa mucizevi gelse de matematiksel yollarla kolaylıkla ispatlanabilir (**Problem 2.4**).

Hesaplanabilirlik konusuna geri dönelim. Fiziksel dünyada her makinenin sonlu sayıda girdisi ve çıktısı olur demiştik. Ardından bu sonluluğu kullanarak her makineyi bir doğruluk tablosuna dönüştürebileceğimizi göstermiştik. Şimdi de her doğruluk tablosunu yalnızca NOT ve AND'den oluşan bir önerme olarak yazabileceğimizi öğrendik. O hâlde fiziksel dünyada NOT ve AND kapılarını hesaplayan bir süreç geliştirirsek, tüm sonlu makineleri fiziksel olarak inşa edebilmemizi sağlayacak olan temeli atmış oluruz. Bilgisayar böyle icat edilir!

## Elektronik bilgisayarların icadı

NOT ve AND kapıları için tasarlayacağımız fiziksel düzeneğin aşağıdaki koşulları sağlaması gerekir ki tüm sonlu makineler teorik olarak inşa edilebilsin.

**Özellik A: Çift durumluluk.** Düzeneğin parçaları, yalnızca iki farklı değer alabilen bir fiziksel özelliğe sahiptir.

**Özellik B: İstikrar.** Söz konusu özelliğin durumu, dışarıdan herhangi bir etki olmadığı sürece sabit kalır.

**Özellik C: Bilgi aktarımı.** Düzeneğin parçaları arasında söz konusu özelliğin durumunu taşıyan bir etki mekanizması vardır.

**Özellik D: Kapılar.** Bu düzenekte NOT ve AND kapılarının karşılık geldiği durum değişikliklerini deterministik şekilde yaratan, kısa zamanda çok kopyası üretilebilen parçalar vardır.

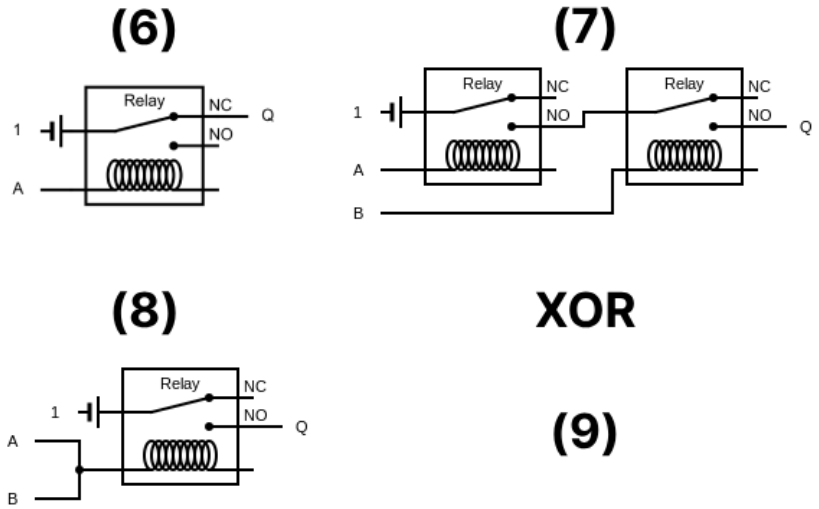
**Özellik E: Bileşkeler.** Bir kapının çıktısı her zaman başka bir kapının girdi ucuna ulaştırılabilir.

**Özellik F: Yayılma.** Bir kapının çıktısı kendisini kopyalayarak birden fazla kapıya girdi olarak ulaştırılabilir.

**Özellik G: Sınırlı zaman.** Düzenekte bilginin bir noktadan başka noktaya ulaşması için geçen zaman üstten sınırlıdır.

Claude Shannon, 1937 tarihli yüksek lisans tezinde *röleli* elektrik devrelerinde bu özelliklerin tamamının sağlandığını fark etti (**Problem 2.7**). Üstelik elektrik devreleriyle mantıksal önermeler arasında *birten* (birebir ve örten) bir eşleme olduğunu gösterdi. Shannon'ın fikirlerinin ilk uygulama alanı, telefon santrallerindeki röleli devrelerin önermelere dönüştürülüp sadeleştirilmesi oldu. Fakat kısa sürede bu ilişkinin diğer yönünden de faydalanıldı ve önermeleri kodlayan devreler inşa edilmeye başlandı. Böylece ilk bilgisayar icat edildi.

Röle, bir bobin ve iki anahtardan oluşan bir sistemdir. Bobinden akım geçtiği zaman normalde kapalı (NC) olan anahtar açılır, normalde açık (NO) olan anahtar kapanır. Devredeki bir noktadan akımın geçmesini 1, geçmemesini 0'a karşılık getirirsek şu ana kadar gördüğümüz mantık kapılarını gerçek dünyada aşağıdaki gibi inşa edebiliriz:

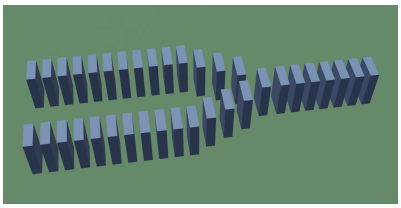



Elektrik akımıyla bilgi taşımanın tek yolu röleli sistemler değildir. 1947'de Bell Labs'ta icat edilen *transistör*, mantık kapılarının çok daha verimli ve küçük hacimli olmasını sağlayarak bilgisayarların seri üretimine giden yolu açmıştır.

Asıl sorumuz şu: Mantık cebrini fiziksel dünyaya taşımanın en verimli yolu elektrik olabilir ama *en eğlenceli* yolu hangisi?

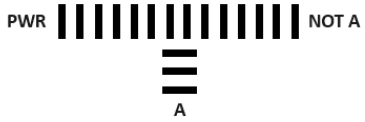
## Domino bilgisayar

Yalnızca domino taşlarını kullanarak Özellik A-G'yi sağlayan bir sistem kurabilir miyiz? Bir domino taşı ya dik durur ya da düşmüştür (A) ve üstelik bu durumlar istikrarlıdır (B). Bir taşın düşmesi önündekinin de düşmesini sağlar (C). Sonlu sayıda dominodan oluşan bir dizinin sonsuz döngüye girmesini de beklemeyiz, bir noktada hareket sonlanacaktır (G). O hâlde geriye yalnızca mantık kapılarını kurallara uygun şekilde kurmak kalıyor.

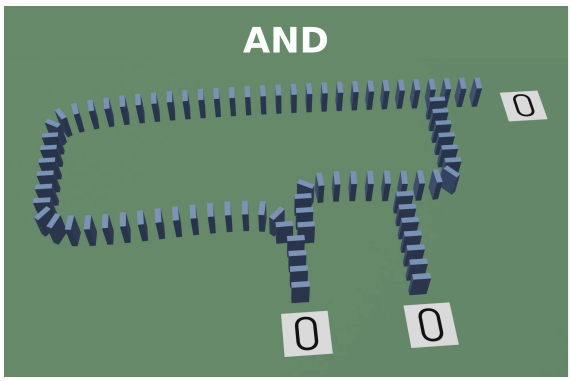


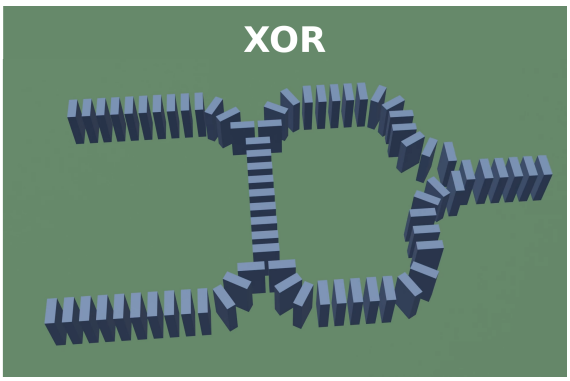
Solda gördüğünüz üzere, dominolarla kolaylıkla bir OR kapısı inşa edebiliriz. Üstelik aynı şekli tersine çevirerek bilgi akışını kopyalayabileceğimizi de görebiliriz (Özellik F). Tıpkı $\{NOT, AND\}$ kümesi gibi $\{NOT, OR\}$ kümesi de fonksiyonel olarak tamamlanmıştır. Dolayısıyla bir NOT kapısı tasarlamayı da başarabilirsek domino bilgisayarımız hazır!

Düşmüş bir dominoyu yerinden kaldırmak kolay değildir. Bunun ilk yan etkisi, domino bilgisayarımızı sadece bir kez çalıştırabilmek olacak. Fakat daha da ciddi bir sorunumuz var: NOT kapısının girdi dominosu dimdik ayakta dururken çıktı dominosu kendiliğinden yere düşmeli. Girdinin düşmesi ise çıktı dominosunu "ayağa kaldırmalı". Bu nasıl mümkün olabilir?




Domino NOT kapısını inşa etmenin tek yolu, elektrik devrelerinde olduğu gibi harici bir "güç kaynağı" kullanmak gibi görünüyor. Sağdaki şekilde $A$'nın değeri ne olursa olsun, NOT kapısına girdi verilme anında PWR dominosunu düşürelim. $A = 0$ durumunda çıktı dominosu düşer, yani çıktı 1 olur. $A = 1$ durumunda ise $A$'yı takip eden dominolar PWR'nin yolunu keserek çıktı dominosunun düşmesini engeller, böylece çıktı 0 olur. Shannon'ın röleli NOT kapısıyla neredeyse aynı düzenek! Fakat elektrik devrelerinin aksine burada zamanlama önemli. PWR çok erken düşerse $A = 1$ olduğu hâlde bu kapı 1 çıktısı verebilir. Dolayısıyla diğer mantık kapılarını inşa ederken OR ve NOT kapılarını birleştirmek yerine buradaki "yol kesme" mantığını kullanarak daha basit ve zamanlamaya ihtiyaç duymayan tasarımlar yapacağız (**Problem 2.9, Problem 2.10**):




Artık bu kapıları birleştirerek her türlü toplama işlemini domino bilgisayarımızda hesaplayabiliriz! Bizden daha erken davranan deliler var tabii. 2012 yılında Matt Parker liderliğindeki bir ekip, tarihin ilk domino 4-bit toplayıcısını inşa etti. İnşa süreci ve sonuçlarını Parker'ın "Stand-up Maths" adlı YouTube kanalında izleyebilirsiniz *(The 10,000 Domino Computer)*.

Şu anda dünyanın en büyük domino bilgisayarı rekorunu Ocak 2024'te inşa ettikleri 6-bit toplayıcı ile Finlandiyalı bir liseli öğrenci grubu elinde tutuyor. Daha işsiz bir güruh cihan yüzü görene dek bu rekorun kırılması olası gözükmüyor. Fakat üzülmeyin! Gayrinizami hesaplama (unconventional computing) teknikleri yalnızca domino bilgisayarlarla sınırlı değil. Bu işi suyla, ışıkla, bilardo toplarıyla yapan deliler de oldu. Siz de tarihin en büyük *ateşli bilgisayarını* yaparak kendinize bir dünya rekoru edinebilirsiniz!