CNS/CS/EE 188a Computation Theory and Neural Systems - Homework #3

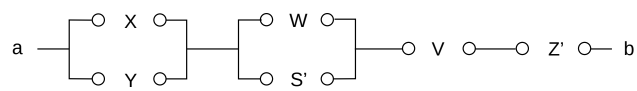
Ling Li, ling@cs.caltech.edu

February 13, 2001

3.1 Simplification of Relay Circuits.

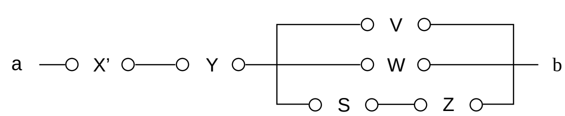
(a)

$$
\begin{aligned}
X_{ab} &= (X+Y+Z)(X+V)(X'+Y) + (W+S+ZW)(S'+Z'+V) + VZ + Z' \\
&= (X+Y+1)(X+V)(X'+Y) + (W+S+W)(S'+0+V) + V + Z' \\
&= (X+0)(X'+Y) + (W+S)(S'+0) + V + Z' \\
&= XY + WS' + V + Z'.
\end{aligned}
$$



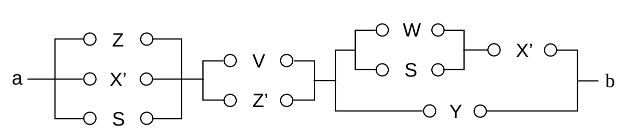
(b) It is easy to see that the first 2 blocks are just X' and Y.

$$
\begin{aligned}
X_{ab} &= X'(S+V+X') + Y(S+Y+Z') + SY + (S+Y+X+W)Y'VW(S+Z) \\
&= X' + Y + (S+0+1+W)VW(S+Z) \\
&= X' + Y + VW(S+Z)
\end{aligned}
$$



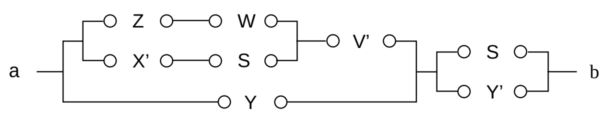
(c)

$$
\begin{aligned}
X_{ab} &= (V'Z+SV)X'S + (X'+Y+SX+SY'W)SWXY + X'Y \\
&\quad + (XY+WSV+VZ)VZ' + VZ' \\
&= (V'Z+1V)X'S + (X'+1+SX+SY'W)SWXY + X'Y + VZ' \\
&= ZX'S + VX'S + SWY + X'Y + VZ' \\
&= ZX'S + Z'V + (SW+X')Y.
\end{aligned}
$$



(d)

$$\begin{aligned}
X_{ab} &= (S' + Y + XYZ)(X + Y)(Z + VW)(Y + S)(X' + S) + (Y + Z)Y'SV \\
&\quad + (X + Z)XZ'SY' + V'SY + YV' + SY' + V'S \\
&= (S' + Y)(S + Y)(X + Y)(Z + VW)(X' + S) + YV' + SY' \\
&= Y(Z + VW)(X' + S) + YV' + SY' \\
&= Y((Z + W)(X' + S) + V') + SY'
\end{aligned}$$



3.2 Boolean Function and Switching Circuits.

(a) It should be $4(n-1)$, already indicated in Shannon's paper.

(b) $TH_i = 1$ iff the number of 1's in the input is larger than or equal to i. Thus when the number of 0's in the input is larger than $n-i$, $TH_i = 0$. So $TH_i = S_{n-i+1,\ldots,n}$ and $\overline{TH}_i = S_{0,1,\ldots,n-i}$, for $0 < i \leq n$.

From Shannon's construction (Fig. 23 in his master paper), we can construct a circuit for $S_{0,1,\ldots,i}$ by two approaches. One way is to use all the switches in the 'parallelogram' consisting of the left-most $(n-i+1)$ slope lines and the lowest $(i+1)$ horizontal lines. Note that all switches on the $(n-i+1)^{\text{th}}$ left-most line is substituted by a short-cut line. Thus we use $(i+1)\times(n-i)$ elements on the $(i+1)$ horizontal lines and $(n-i)\times i$ switches on the slop lines. The total number of elements is $(n-i)(2i+1)$.

The other way is to construct the dual of the circuit for $S_{i+1,\ldots,n}$. Similar to the previous one, the circuit for $S_{i+1,\ldots,n}$ uses all switches within the parallelogram consisting of the left-most $(n-i)$ slope lines and the lowest $(i+2)$ horizontal lines, however, with the $(i+2)th$ lowest line substituted by a short-cut line. Thus the total number of elements is $(n-i)\times(i+1)+(i+1)\times(n-i-1) = (i+1)(2n-2i-1)$.

Note that the difference in the number of elements used in these two approaches is

$$(n-i)(2i+1) - (i+1)(2n-2i-1) = 2i+1-n.$$

So we will use the first approach for $i = 1, 2, \ldots, \lfloor\frac{n-1}{2}\rfloor$, and use the second approach for $i = \lfloor\frac{n-1}{2}\rfloor + 1, \ldots, n$.

Thus, either of $\overline{TH}_i$ and TH_i can be implemented with no more than

$$\min\{(n-i+1)(2i-1), i(2n-2i+1)\}$$

elements.

(c) To construct an arbitrary LT function, we can use the property that each LT function is a unate function. That is, only one of X_i and X_i' will appear in the expanding of the function. Then, a function $f(X_1, X_2, \ldots, X_n)$ may be expanded as

$$f(X_1, X_2, \ldots, X_n) = X_1 f(1, X_2, \ldots, X_n) + f(0, X_2, \ldots, X_n).$$

However, this doesn't help much.

Another thing we can do: consider *LT* function $f(x_1, x_2, \ldots, x_n) = \text{sgn}\left(\sum_{i=1}^{n} w_i x_i - w_0\right)$. All the weight can be changed into non-negative integers. (We know how to change the weights into integers. And, for those x_i with negative weights, using x_i' instead of x_i and vice verse, and adjusting w_0 accordingly, we can get all w_i for $i > 0$ non-negative. If w_0 now is negative, then the function is 1 and w_0 can be set to 0.) So without loss of generality, we only consider that all the weights are positive integers.

Consider all the subsets of $\{x_i\}_{i=1}^{n}$ whose corresponding weights sum is larger than or equal to w_0. Among these subsets, some are not super-sets of other sets. We call such sets 'support sets'. Then we can construct a parallel circuit for each support set and then connect them into a serial circuit. That is the circuit for this *LT* function.

For example, $f(x_1, x_2, x_3) = \text{sgn}(2x_1 - x_2 + x_3 - 1)$. First convert it into $f(x_1, x_2', x_3) = \text{sgn}(2x_1 + x_2 + x_3 - 2)$. All the subsets with weights sum larger than or equal to 2 is $\{x_1\}$, $\{x_1, x_2\}$, $\{x_2, x_3\}$, $\{x_1, x_3\}$. Thus the support sets are $\{x_1\}$, $\{x_2, x_3\}$. So the circuit is constructed as $f = x_1 + x_2' x_3$. Note that we changed x_2 back to x_2'.

The size of the circuit is the sum of the sizes of all support sets. However, it is difficult to be represented as an explicit function of all the weights.

3.3 Some Size Bounds on Switching Circuits. We use X_i to denote variables, where i is an integer.

(a) Assume that any function of n variables requires no more than g_n switching elements. Then for any function of $n+1$ variables $X_0, X_1, \ldots, X_n$, it can be expanded as follows:

$$f = f(X_0, X_1, \ldots, X_n) = X_0 f(1, X_1, \ldots, X_n) + X_0' f(0, X_1, \ldots, X_n).$$

Now $f(1, X_1, \ldots, X_n)$ and $f(0, X_1, \ldots, X_n)$ are functions of n variables, and either of them can be implemented with at most g_n elements. Thus we can implement f with no more than $(2g_n + 2)$ elements.

(b) From (a), we get $g_{n+1} \leq 2g_n + 2$, if g_n is the minimum number of elements needed to implement any function of n variables. From $g_0 = 0$, we get

$$g_n + 2 \leq 2(g_{n-1} + 2) \leq \cdots \leq 2^n (g_0 + 2) = 2^{n+1}.$$

So $g_n \leq 2^{n+1} - 2$, implying that any Boolean function of n variables can be implemented with at most $2^{n+1} - 2$ elements.

(c) By expanding only the first $n-1$ variables, $\sum_{i=1}^{n-1} 2^i$ elements are used, since the first expanded variable appears 2 times, and the second expanded variable appears 2^2 times, and so on. Since only two switching elements are required to implement all possible functions of the unexpanded variable, and those elements can be shared by different partial minterms, the total number of elements needed is $2 + \sum_{i=1}^{n-1} 2^i = 2^n$. (If we use the form "product of sum" to expand the function, we will see a "binary tree" in the circuit. Each path from the root to leafs corresponds to a partial maxterm. Each leaf is connected to one of those 4 'elements' representing four different functions of X_n, that is, X_n, X_n', 1 and 0. The last 2 'elements' are indeed not switches. So the total number is 2^n.)

(d) Figure 1 gives how to use 20 switches to compute all 16 functions of two variables. The key is to re-use some elements, provided no 'bad' shortcuts happen. Then, using the similar reason in (c), for any function of $n > 2$ variables, only $\sum_{i=1}^{n-2} 2^i + 20 = 2^{n-1} + 18$ elements are required.

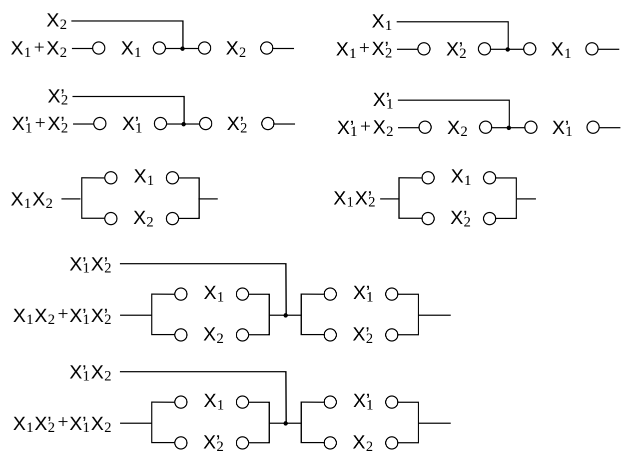


Figure 1: 20 switches to compute all 16 functions of two variables. Two trivial functions, 0 and 1, are not shown.

(e) I don't know how to get $2 \times 2^{2^m}$. I tried induction.

Using the same way to count all the elements in (c) and (d), we get the number of elements used, if we expand about all but m variables, is

$$\sum_{i=1}^{n-m} 2^i + 2 \times 2^{2^m} = 2^{n-m+1} + 2 \times (2^{2^m} - 1).$$

If $m = 3$, it is $2^{n-2} + 510$. By solving $2^{n-2} + 510 < 2^{n-1} + 18$, we get $n \geq 11$. That is, $n = 11$ is the smallest n for which we should expand about all but 3 variables.

3.4 Multiplication and Squaring.

(a) $Z = X^2$. $Z - X = X(X-1)$ is an even number, since either X or $(X-1)$ is even. Thus $z_0 = x_0$.

X can be either even or odd. If X is even, say $X = 2\hat{X}$, then $Z = 4\hat{X}^2$ is a multiple of 2^2. So $z_1 = 0$. If X is odd, say $X = 2\hat{X} - 1$, then $Z = 4(\hat{X}^2 - \hat{X}) + 1$ and the last two bits of Z are 01. In fact, the last 3 bits of Z are 001, since we've already known $(\hat{X}^2 - \hat{X})$ is even. Thus we also have $z_1 = 0$. So for any X and $Z = X^2$, $z_1 = 0$.

(b) From

$$(X+Y)^2 - (X-Y)^2 = 4XY,$$

we can first calculate $4XY$ by using ADD, SUB, and $SQUARE$ elements. Then we can right shift $4XY$ for 2 bits, or just drop the last two bits of $4XY$, we get XY.

3.5 Computing the $MAXIMUM$ function.

(a) We can do comparison of two m-bit integers with one LT gate. That is,

$$C(X,Y) = \operatorname{sgn}\left(\sum_{i=0}^{m-1} (x_i - y_i) 2^i\right).$$

Thus we can implement $FIRST$ as $AND(C(Z_1, Z_2), C(Z_1, Z_3), \ldots, C(Z_1, Z_n))$, which can be done by LT_2. For this kind of construction, the size of the circuit is n, i.e., $n-1$ gates for comparisons and 1 gate for AND. However, for $n = 2$, the size is 1 since now the AND gate is not needed.

(b) In layers 1 and 2, we can use n $FIRST$ functions, with Z_i as the first input number for the ith $FIRST$ function (denoted by $FIRST_i$), to find out which Z_i (may not be unique) is equal to $MAXIMUM(Z_1, Z_2, \ldots, Z_n)$. Then in the third layer, we can use AND gates. The gate $AND_{i,j}$ takes the output of $FIRST_i$ and $z_{i,j}$ as inputs, and outputs $z_{i,j}$ if Z_i is the maximum, or 0 if not. Then we can assemble the maximum by using OR gates. The gate OR_j takes the outputs of AND_{ij} for $i = 1, 2, \ldots n$ and output the jth bit of the maximum. For this construction, the size of the circuit is $n^2 + mn + m$.

However, it is obvious we do not need n^2 gates in the first 2 layers. In stead of using $C(Z_i, Z_j)$ for all $i \neq j$, we can use $C(Z_i, Z_j)$ only for $i < j$. Thus we still can construct the n $FIRST$ functions, by only $n + \frac{n(n-1)}{2} = \frac{n(n+1)}{2}$ gates. We can even drop one $FIRST$ function, since it can be calculated by the other $FIRST$ functions, and such calculation can be merged into layer 3.

In fact, since we can collapse layer 2 and 3 into one layer (Homework 2.2(d)), we can use LT_3 to construct $MAXIMUM$. For layer 1, we use $C(Z_i, Z_j)$ for $i < j$. For layer 2, the gate $AND_{i,j}$ takes the outputs of $C(Z_{k'}, Z_i)$ and $C(Z_i, Z_k)$ for all $k' < i < k$, and $z_{i,j}$ as the inputs. The corresponding weights are -1, 1, and 1. And the threshold is $(i - n - 1)$. Thus, the output of $AND_{i,j}$ is 1 iff $C(Z_{k'}, Z_i) = 0$ and $C(Z_i, Z_k) = 1$ for all $k' < i < k$ and $z_{i,j} = 1$. That is, $AND_{i,j}$ is $z_{i,j}$ if Z_i is the one with smallest i such that $Z_i = MAXIMUM$, and is 0 otherwise. Layer 3 is constructed as layer 4 in the LT_4 construction. Thus, the total number of gates is

$$\frac{n(n-1)}{2} + nm + m.$$

(c) The LT_3 circuit for $n = 3$ and $m = 4$ is shown in Figure 2.

(d) $MAXIMUM \in LT_3$, which is proved in (b).

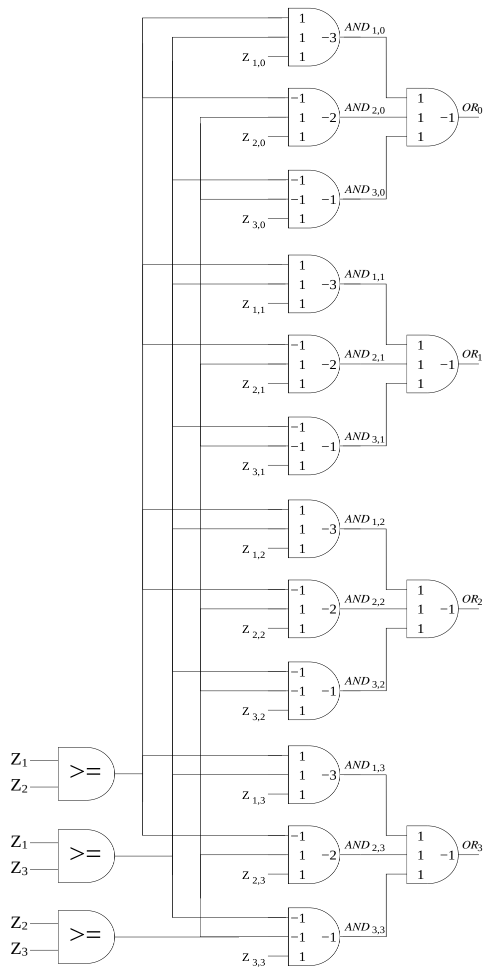


Figure 2: LT_3 circuit for $n = 3$ and $m = 4$. 19 elements are used.