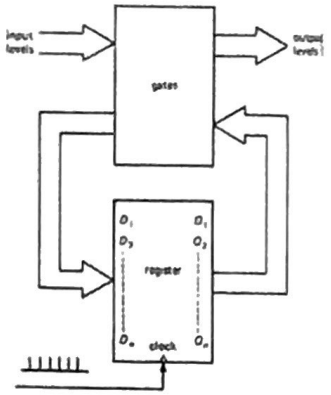
We have discussed his activities...

But who is he?

How is he constructed?

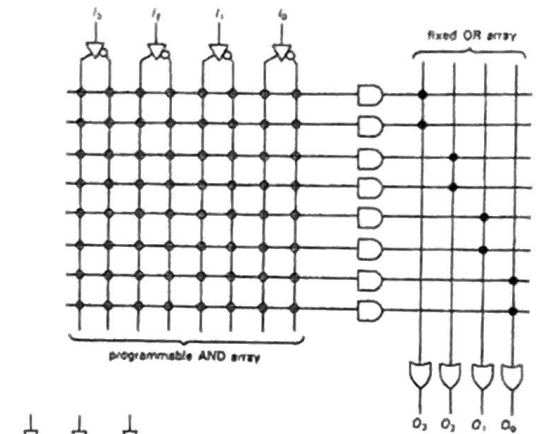
Can we make our own?

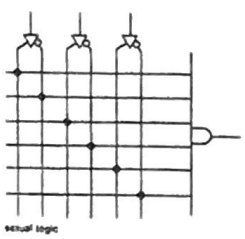
Finite State Machines:



See Horowitz and Hill, 2nd edition, Chapter 8, for more details on these topics and descriptions of these included figures.

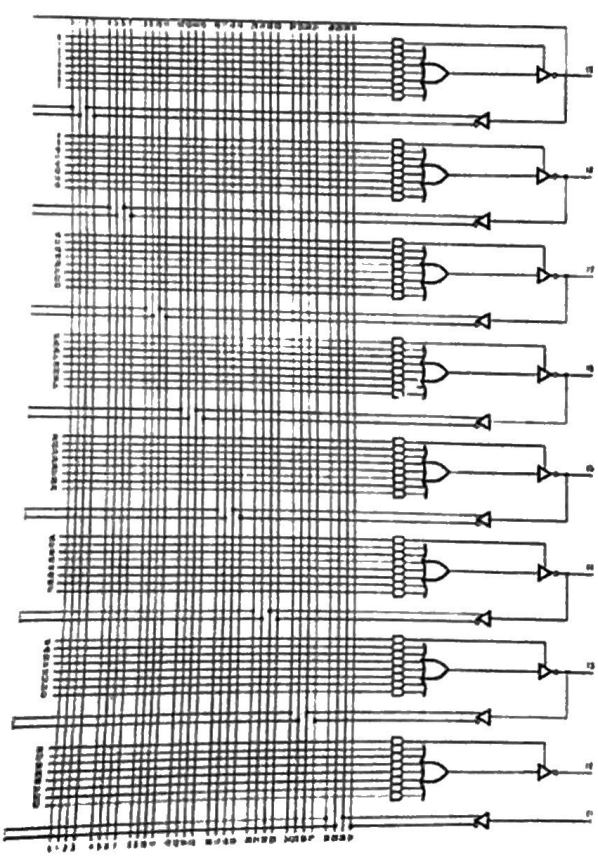
A convenient way to make a bank of combinational logic, or a collection of gates: programmable logic devices. We use one on our R31JP (Xilinx 9536 PLD) to create the XIO select line, and to switch memory positions for mon/run. Here's a simple model of part of a simple PAL.



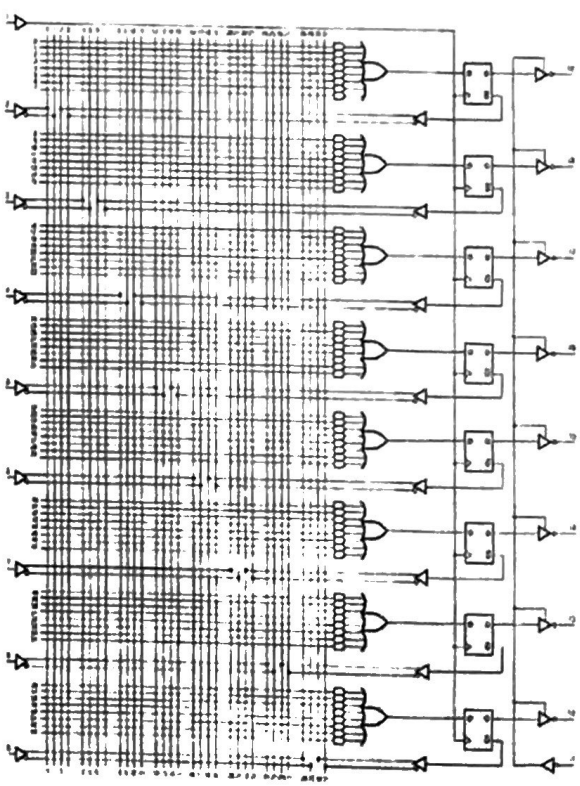


Note the schematic shorthand used to describe the programmable array

A combinational PAL (16L8, with 16 inputs max and 8 outputs max:

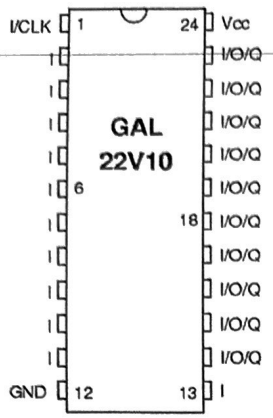


A PAL with registered outputs. The flip-flops can be used to make counters or as memory for "state". This is a 16R8



DIP

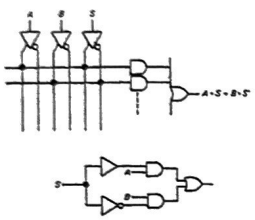
Here's the PAL in your chip kit:



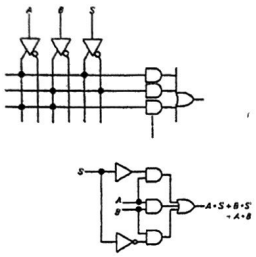
It can be programmed with the CUPL language compiler, similar in many ways to RASM.

Be careful when programming to avoid poor logic, race conditions, designs that won't fit in the device, etc:

Two-Bit Selector:
With race condition:



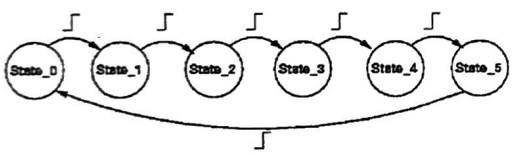
With a fix for static race condition:



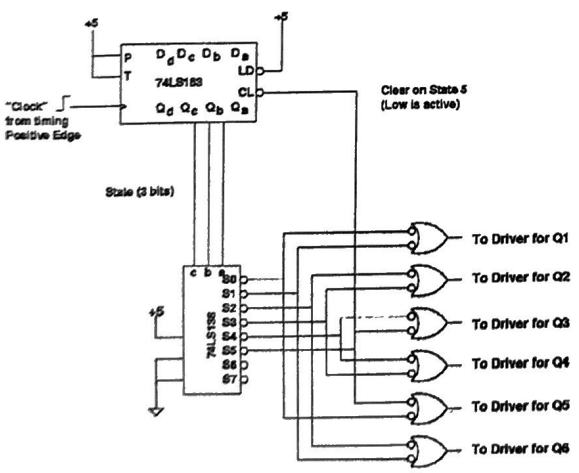
Display decoder, from Horowitz and Hill:

```
/**  Inputs  **/

PIN 1   = a   ;    /*  segment a  */
PIN 2   = b   ;    /*  segment b  */
PIN 3   = c   ;    /*  segment c  */
PIN 4   = d   ;    /*  segment d  */
PIN 5   = e   ;    /*  segment e  */
PIN 6   = f   ;    /*  segment f  */
PIN 7   = g   ;    /*  segment g  */


/**  Outputs  **/

PIN 19  = !D3   ;  /* msb of hex encode   */
PIN 18  = !D2   ;  /*                     */
PIN 17  = !D1   ;  /*                     */
PIN 16  = !D0   ;  /* lsb                 */


/** Declarations and Intermediate Variable Definitions **/

zero = a & b & c & d & e & f & !g ;
one = !a & b & c & !d & !e & !f & !g ;
two = a & b & !c & d & e & !f & g ;
three = a & b & c & d & !e & !f & g ;
four = !a & b & c & !d & !e & f & g ;
five = a & !b & c & d & !e & f & g ;
six = a & !b & c & d & e & f & g ;
seven = a & b & c & !d & !e & !f & !g ;
eight = a & b & c & d & e & f & g ;
nine = a & b & c & !d & !e & f & g
     # a & b & c & d & !e & f & g ;     /* two ways */
hexa = a & b & c & !d & e & f & g ;
hexb = !a & !b & c & d & e & f & g ;
hexc = !a & ! b & !c & d & e & !f & g
     # a & !b & !c & d & e & f & !g ;  /* two ways */
hexd = !a & b & c & d & e & !f & g ;
hexe = a & !b & !c & d & e & f & g ;
hexf = a & !b & !c & !d & e & f & g ;


/**  Logic Equations  **/

D3 = eight # nine # hexa # hexb # hexc # hexd # hexe # hexf ;
D2 = four # five # six # seven # hexc # hexd # hexe # hexf ;
D1 = two # three # six # seven # hexa # hexb # hexe # hexf ;
D0 = one # three # five # seven # nine # hexb # hexd # hexf ;
```


Finite State Machine for a Motor Drive:



Hardware Implemenation:



Implementation of a 138 selector in CUPL:

```
Name            demuxer;
Partno          ;
Revision        01;
Date            8/10/08;
Designer        PLD Expert;
Company         MIT;
Location        None;
Assembly        None;
Device          g22v10;

/* Simple combinatorial logic: 2-to-1 MUX */

pin 2 = a;
pin 3 = b;
pin 4 = c;
pin 5 = xiosel;

pin 14 = d;
pin 15 = e;
pin 16 = f;
pin 17 = g;
pin 18 = h;
pin 19 = i;
pin 20 = j;
pin 21 = k;

d = xiosel # !(!a & !b & !c);
e = xiosel # !(a & !b & !c);
f = xiosel # !(!a & b & !c);
g = xiosel # !(a & b & !c);
h = xiosel # !(!a & !b & c);
i = xiosel # !(a & !b & c);
j = xiosel # !(!a & b & c);
k = xiosel # !(a & b & c);
```