Creating your own custom ICs, the MIDI music interface, ASCII/HEX/decimal conversions, finding exotic metals

By Don Lancaster

Sometimes a really big hacker breakthrough will happen instantly in a blinding flash. Other times, an improvement here, a second source there, a big price reduction over there, and some day you wake up and realize that something really great is all of a sudden hacker feasible.

This month's breakthrough is that hackers can finally design and build their own full-feature custom integrated circuits, quickly and cheaply, even on a kitchen table.

We'll jump right into this.

How Can I Build my own Custom ICs?

There have long been three main routes to creating custom integrated circuits. These are the EPROM, the PLA, and the Gate Array. All three have been modified and improved in many ways to the point that you can now create your own full-feature custom integrated circuits. Best of all, you can now do this on your own and even on a kitchen table at reasonable prices, without any arcane emulation software or ridiculous setup charges.

The EPROM is probably the single custom integrated circuit that is already the most familiar to hardware hackers. As you can see from the ads right here in *Modern Electronics*, EPROMs are now cheap, dense and easy to get.

Figure 1 shows what is inside an EPROM. An EPROM is simply a table lookup device. For every possible input address, a data word is output.

Output words might be a sequence of computer software commands, the values for a sine wave or other trig function, a fixed ASCII message, the allophones in a speech synthesizer, or just about anything else where you need a complete set of often irrational "input address" to "output data" conversions.

To program an EPROM, you first erase it with strong short-wavelength ultraviolet light. Then you go through a programming process that connects ones or zeros to the data lines for every input address combination.

Programming can be done on a stand-alone programmer, on a plug-in card inside a personal computer, or by using one of many custom programming services. *E-Tech Services* is one place I have found that does fast, low cost and no-hassle programming. Chances are your local computer club has at least one EPROM burner available for your use.

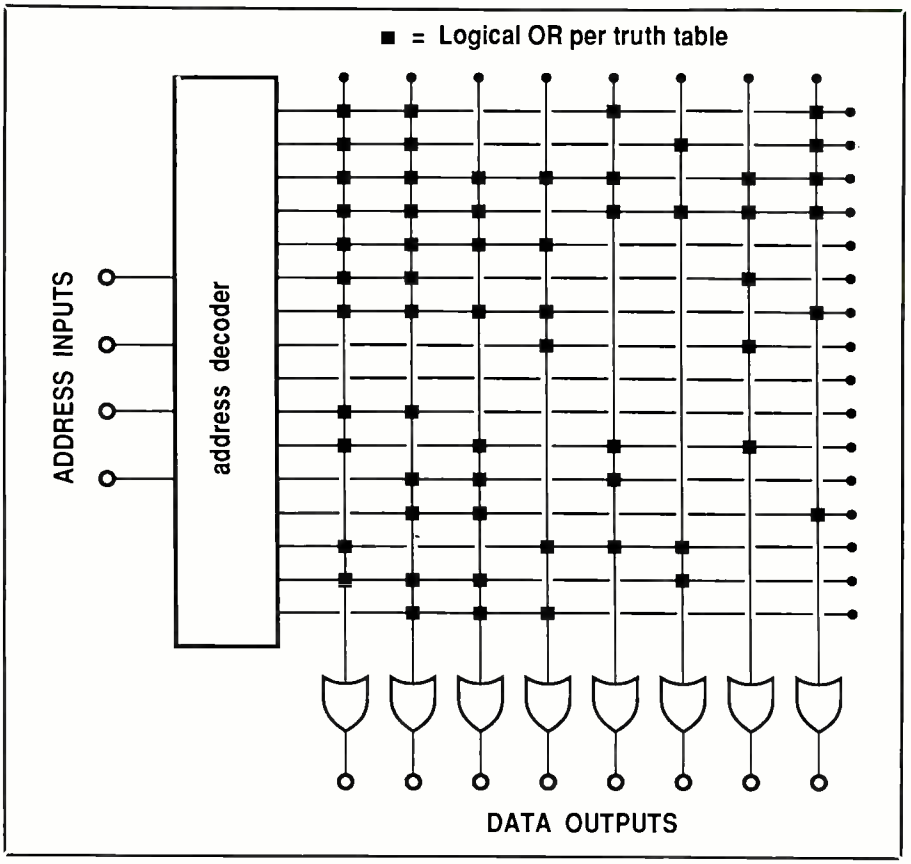

Fig. 1. An EPROM provides outputs for all possible input states.

There are several things that make EPROMs more attractive today than ever before. You can get EPROMs up to one megabyte in size, although the 64K (8K × 8) 2764 and the 128K (16K × 8) sizes are by far the most popular today. Pricing is so low that you can easily "waste" most of the humongous internal storage of an EPROM and still outperform a more specialized circuit made up of ordinary digital logic chips.

Until very recently, EPROM erasers have been obscenely priced. *JKL Components*, however, has just introduced a miniature and easy-to-use BF727 ultraviolet lamp that lets you snap the eraser over the EPROM without even removing the chip from its socket. More on this beauty after I check it out.

A big safety reminder if you plan to use a UV eraser: Short-wavelength ultraviolet light can instantly cause permanent blindness. Make sure that the erasing scheme you use has a positive safety interlock prevents direct viewing.

A relative of the EPROM, the EEPROM can have single words electrically erased and reprogrammed, without having to erase the whole chip and start over again. Additionally, no high programming voltage is needed.

Even after several false starts, EEPROMS still are not quite here yet for mainstream use. While they are much cheaper than they have been in the past, EEPROMs are still rather expensive compared to ordinary EPROMS. They have reliability and standardization

problems, though. Many are guaranteed for only a few thousand write cycles. So, today most people use EPROMs for most everything, but EEPROMs are pretty much still limited to niche uses such as electronic TV tuners, and power-down system constant storage in personal computers and laser printers.

The next step up in complexity is called the PLA, which is short for Programmable Logic Array. These were pioneered by *Monolithic Memories*.

As Fig. 2 shows, PLAs are used to first logically AND combinations of inputs, and then to logically OR the various AND outputs together. It turns out that many Boolean logic equations can be converted into this special form of AND-ed combinations followed by ORed recombinations.

Until recently, PLAs were pretty gruesome. They were expensive, hard to program, and there were dozens of different specialized PLA variations that were almost, but not quite, identical to each other. Their power needs were just plain excessive. Ad claims grossly distorted the PLA's capabilities through ludicrously inflated "gate counts." Worst of all, programming was a one-shot deal that blew fuses in the chip.

Much of that is changing now. Old style one-shot PLAs are now under a dollar in quantity; programmers for them are becoming available for personal computers; and a language called PALASM is available to simplify the logic equations needed.

Second-generation reusable PLA devices are now available. *Cypruss Semiconductor* is one source of erasable EPROM-style PLAs, as is *Altera*.

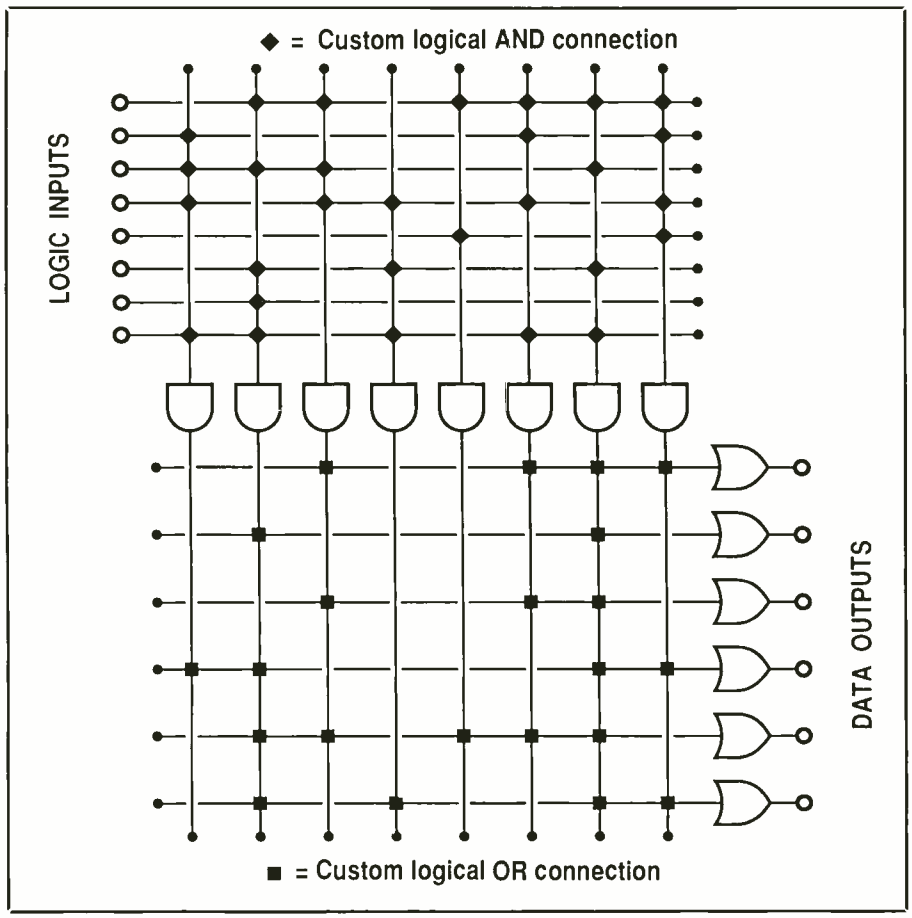

Fig. 2. A PLA performs custom Boolean AND/OR logic on its inputs.

Better still, the GAL series from *Lattice Semiconductor* is both erasable and reprogrammable. Even more impressive is that one single reusable device can be used to emulate just about *any* of the older dedicated one-shot PLA chips. A fancy output stage on each line can be configured to behave as a latching register, a counter, or as a direct logic output, in your choice of "active-high" or "active-low" states.

The ultimate custom integrated circuit is the gate array. You can build *anything* digital using nothing but NAND gates in combination. As Fig. 3 shows, a gate array is nothing but a massive collection of logical NAND gates—hundreds, thousands, or even tens of thousands of them.

Since you can literally do anything with them, gate arrays are far more powerful and far more flexible than EPROMs or PLA devices.

Until recently, the gate array was far and away the most violently, viciously, and vehemently anti-hacker device available anywhere ever. Little things like $35,000 setup charges, 42-week delivery, complex and arcane emulation software that would not run on same computers, and the "it has to be right the first time or forget it" mentality tended to put something of a damper on hacker gate array enthusiasm. Fortunately, that is also changing. *Xylinx* is now offering erasable and reprogrammable gate arrays. These work by downloading the array connections when power is first applied. While pricing is still way out of line, it is only a mater of time until you'll be able to buy program-your-own gate arrays at EPROM prices directly from *Modern Electronics* advertisers.

What is on Special this Month?

Field effect transistors are yet another example of components that have quietly gotten better, ridiculously cheaper, and much more hacker-usable. For example, *Siliconix* has a pair of switching field effect transistor samples that are absolutely ideal hacker parts. The samples are free in singles when you phone or write for them using a letterhead request. First is the 2N7000 Fetlington. This is a small

package field effect transistor that is rated 60 volts and up to 200 milliamperes of continuous current. In quantity, this beauty sells for less than a dime. It is ideal for such things are driving relays, small solenoid valves, alarms, incandescent lamps and such.

Siliconix's second sample is the 2N7004 FETDIP. This is a slightly larger device with a much higher power rating. It can handle 100 volts and up to 6 watts of continuous power dissipation with suitable heat sinking. Maximum current is 1 ampere. "On" resistance is a low 0.6 ohm. Uses would include line drivers, high-power switches, current sinks, and just about any other application where you need to switch a "fair to middlin" power load.

Several test circuits are included on each of the sample cards. Higher voltage versions are also available, up to the 2N7006 that is rated to 350 volts. Since negligible input current is needed, these are well suited for microprocessor control of medium power loads.

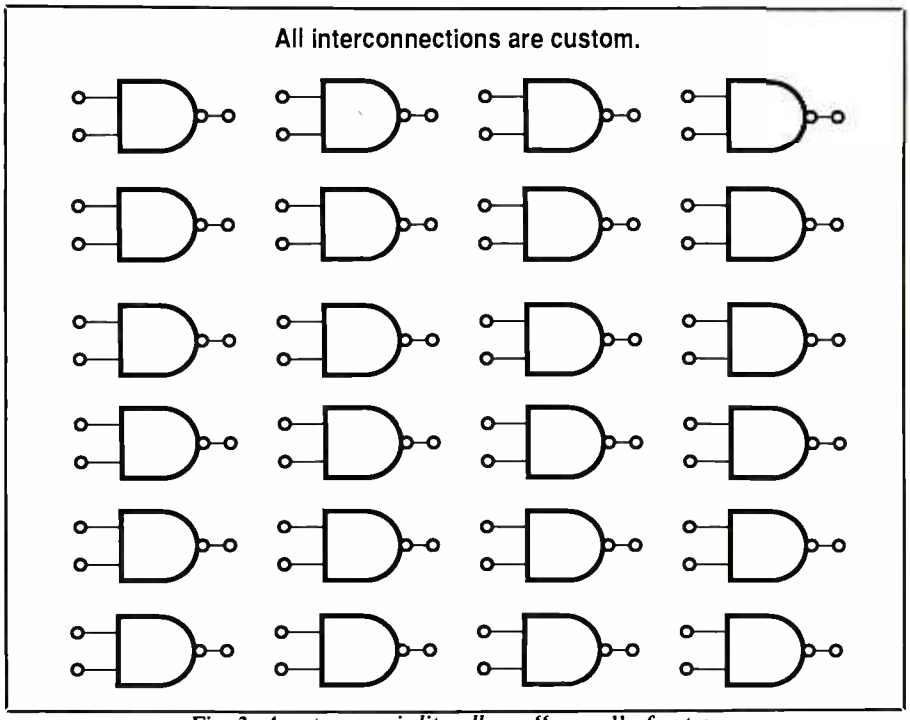

Fig. 3. A gate array is literally an "ocean" of gates.

What is a MIDI Music Interface?

The MIDI interface is used in electronic music to interconnect synthesizers, music modules, and personal computers. As you might suspect, MIDI is becoming quite popular.

Two good books on this are Craig Anderton's *MIDI For Musicians* and Joseph Gadoury's *MIDI Made Simple.* Both are available through *Roland.*

How do you Convert Decimal to ASCII?

There are lots of times and places in microcomputer software when you might like to convert a decimal value to its printable ASCII equivalent, and vice-versa. For instance, you might have the decimal score of a game stashed somewhere and want to route it to a video screen. Or you might want to show the number of characters already used or still available in a word processor.

Decimal numbers are usually represented by 4-bit bytes with 0000 = 0, 0001 = 1, 0010 = 2 . . . up to 1001 = 9. One byte is used for each decimal decade. A single 4-bit byte coded this way is called BCD, short for binary coded decimal. Each byte holds one decimal digit.

Figure 4 shows how to get from BCD decimal to ASCII, the standard character code we looked at in a previous column. The rules here are simple: To get from BCD to ASCII, just add decimal 48 or hex $30. To get from ASCII to BCD, just subtract decimal 48 or hex $30. That's all there is to it.

Sometimes a pair of BCD bytes might be combined into an 8-bit word. This is called packed BCD. If you are using packed BCD, you must first unpack the bytes before ASCII conversion, and later repack them after being converted from the ASCII format.

To unpack a low BCD byte, logically AND it with hex $0F. To unpack a high BCD byte, shift or rotate the word to the right four times, and then once again AND it with hex $0F. To pack a BCD word, shift the high BCD byte to the left four times and then OR it with the low BCD byte.

0	0000 - 0011 0000	$30
1	0001 - 0011 0001	$31
2	0010 - 0011 0010	$32
3	0011 - 0011 0011	$33
4	0100 - 0011 0100	$34
5	0101 - 0011 0101	$35
6	0110 - 0011 0110	$36
7	0111 - 0011 0111	$37
8	1000 - 0011 1000	$38
9	1001 - 0011 1001	$39

Fig. 4. Decimal-to-ASCII conversion.

The 6502 microprocessor has a special decimal mode that works directly in packed BCD. Other micros usually have ways of faking the same thing. These days, though, you usually do *not* pack the BCD decades.

How do you Convert Hexadecimal to ASCII?

Things get a tad more complicated when you try to convert between hexadecimal

and ASCII. You might want to do this when you are displaying a "hex dump" of a computer's monitor program on a video screen. Some stand-alone EPROM burners also require their hex bytes be passed back and forth as ASCII characters, as does the bit image information on laser printers.

There are both advantages and penalties to converting hex to ASCII before sending it somewhere. The big advantage is that everything is sent as ordinary numbers or letters. There is thus no possibility of confusing, say a hex $0D for a carriage return, or a $00 for a NUL or a break. Illegal characters cannot be sent since there aren't any. As a further bonus, an ordinary word processor can also be used to process hex characters in ASCII form.

Penalties of hex to ASCII conversion involve both speed and storage size. It takes *two* printable characters to show a single 8-bit hex word pair. Thus, it will take you *twice* as long to send converted bytes over a serial interface than it would to send straight hex.

It also takes twice the space to store hex values in ASCII form. Which gets particularly nasty when printing video or photographs on a laser printer.

To review, a 4-bit byte can be shown in hexadecimal. There are 16 possible 4-bit states. The first 10 states, 0000 through 1001, are shown as decimal digits, exactly as is done with BCD. The final six states are often shown as upper-case letters, with $1010 = A, $1011 = B, up through $1111 = F.

Figure 5 shows how to do these conversions. To get from hex to ASCII, add decimal 48 or hex $30 if the number is 10 or more, add decimal 54 or hex $36.

To get from ASCII to hex, you first have to test to find out if you are converting a letter or a numeral. Such "range checking" is always a good idea anyway. If you have a numberal (CHR$ 48-58 or hex $30-39), subtract decimal 48 or hex $30. If you have a letter (CHR$ 65-69 or hex $41-46), subtract decimal 54 or hex $36 instead.

Actual conversion details depend on your programming style and the microprocessor and language in use. In the Apple IIe monitor, there are several built-in machine-language routines for hex-to-ASCII and ASCII-to-hex conversion. Specifically, $FDE3 will convert the low hex accumulator byte to ASCII. $FDDA will convert the entire accumulator to a hex ASCII pair. $F941 will print first the X register and then the accumulator as four successive hex digits. Finally, $FFA7 will reverse the process and convert the ASCII characters in $003E (low) and $003F (high) into a two-byte hexadecimal value.

0	0000 -	0011 0000	$30
1	0001 -	0011 0001	$31
2	0010 -	0011 0010	$32
3	0011 -	0011 0011	$33
4	0100 -	0011 0100	$34
5	0101 -	0011 0101	$35
6	0110 -	0011 0110	$36
7	0111 -	0011 0111	$37
8	1000 -	0011 1000	$38
9	1001 -	0011 1001	$39
A	1010 -	0100 0001	$41
B	1011 -	0100 0010	$42
C	1100 -	0100 0011	$43
D	1101 -	0100 0100	$44
E	1110 -	0100 0101	$45
F	1111 -	0100 0110	$46

Fig. 5. Hex-to-ASCII conversions.

NAMES AND NUMBERS

Altera
3525 Monroe Street
Santa Clara, CA 95051
(408) 984-2800

Cypress Semiconductor
3901 North First Street
San Jose, CA 95134
(408) 943-2600

ESPI
31194 La Baya Drive
Westlake, CA 91362
(800) 638-2581

E-Tech Services
Box 2061
Everett, WA 98203
(206) 337-2370

JKL Components
13343 Paxton Street
Pacoima, CA 91331
(800) 421-7244

Lattice Semiconductor
15400 NW Greenbrier
Beaverton, OR 97006
(503) 629-2131

Monolithic Memories
2175 Mission College
Santa Clara, CA 95054
(800) 247-6527

Roland
7200 Dominion Circle
Los Angeles, CA 90040
(213) 685-5141

Siliconix
2201 Laurelwood Road
Santa Clara, CA 95054
(408) 988-8000

Xilinx
2069 Hamilton Street
San Jose, CA 95125
(408) 559-7778

You can tear apart this code for a quick study on conversion details. The innermost secrets of easily and rapidly tearing apart machine-language code appear in my *Enhancing your Apple II,* volume I (SAMS #21822).

Where can I buy "exotic" Metals and Rare Earths?

No problem, so long as you are willing to pay exotic prices for them. There's an outfit called *ESPI* that will sell you most any ultra-high-purity element or compound. You might like to write them for their current catalog and price list.

Some examples: A sheet of gadolinium or hafnium costs $35. Indium wire costs $7 a gram, while Lanthanum Fluoride goes for $4 a gram. Neodymium oxide is a steal at 42 cents per gram. Scandium ingots are $88 each. Zirconium wire goes for $1.70 a gram, and so on.

One obvious caution: Before you try playing with *any* exotic element or chemical compound, be sure to *thoroughly* study its properties ahead of time, particularly the reactivity and toxicity. The good old *Handbook of Chemistry and Physics* is an obvious place to begin your research work.

Finally here's our usual reminder that this is your column and you can get free technical help from usper the "Need Help?" box.

My current free handout is so spectacular I'm not even going to tell you what it is. You may even want to frame it. Just ask for your free copy of *Meowwrrrr,* otherwise known as my *puss de resistance.*

NEED HELP?
Phone or write your **Hardware Hacker** questions directly to:
Don Lancaster
Synergetics
Box 809
Thatcher, AZ 85552
(602) 428-4073