MIDIbox AM (Audio Matrix)

A digitally controlled matrix switched patch router for analog signals.

Discussed here

Below is a collection of comments from the thread. My goal is to organize them for the project.

Concept

From Nomical

I would like to have/make an audio patchbay (having a yet to be determined amount of ins/outs due to any diy building difficulties) that has the option of routing any input to any output digitally. No hardware switches or anything, just hook up all the ins and outs and patch everything through a menu. Ideally, this routing should be done on my Macbook Pro using a software mixer (similar to the Cuemix software of my MOTU) and something like a USB or FW connection. As this connection would be far more difficult to build I guess, routing everything on the machine itself using a display would also be sufficient. Each output should just have individual settings for which input is routed to it.

Outputs should also have the option of summoning/combining any number of the input signals to one single output. But the input should stay available to other outputs for making (sub)mixes on desired outputs. How is this summoning achieved? I guess they would all need an individual gain/level options or something to make sure that the summoned/combined signal doesn't overload the chosen output.

Making it even better is to have the option of selecting balanced/unbalanced signal on any individual in/out. My setup is a mixup of machines having balanced/unbalanced ins/outs and it would be nice to have the option of using a mixup of both of them when desired.

Features

- Interfaces with Core Module for control of the Crosspoint and other modules.
- Modular design, with separate modules for crosspoint, I/O buffer and other non-required modules
- Modular Crosspoint switches in 8×8, 16×16, 24×24 or greater matrixs
- Ability to use several types of cross point switches (MT8816 or AD 75019)
- Non-buffered audio I/O Module
- Buffered audio I/O module – unity gain
- Buffered audio I/O module – adjustable gain (with memory)
- Mixing capability with adjustable gain audio I/O module
- Balanced and unbalanced audio I/O -selectable -mixable
- Insert function – ability to switch patches in and out of a preset chain.
- x number of realtime controllers to control input or output gain on any number of patches at the same time.
- VU Metering
- n number of presets

- many to many electronic patching
- MAC and/or PC based software to configure patches and presets
- Additional relay based footswitch simulation

Hardware Design attributes

Raw Comments from forum thread.

option of selecting balanced/unbalanced signal on any individual in/out

I think this is just a case of having balancing transformers on each in and out, and using a switching plug - ie a 6.5mm stereo headphone-type socket where a stereo jack make a balanced connection and a mono jack makes an unbalanced connection - or one of those combination XLR-jack plugs, which have an XLR socket with a 6.5mm mono socket in the middle. The XLR pins go to the balancing transformer and the 6.5mm socket pins bypass it. Or just two separate sockets. Whichever suits you better.

Once it's in the IC it won't matter that it's not balanced

Crosstalk
What I'm saying is that whatever the size of your matrix, you are still routing one-to-N, not N-to-one *. To do N-to-one would still require a summing mixer, or horrible gain problems will follow. This means that your output is derived from only one crosspoint, on one chip. You don't gain anything by chaining chips. Noise will increase as you add chips, because you are connecting outputs together. But not distortion. You are not running through one chip, into another. Doing this does not increase the size of your crosspoint matrix. It simply makes your path more convoluted.

Buffers
What is the cost of the buffers likely to add up to for that though? If you include the cost of the 32 opamps and the extra board space etc, how does it compare to the AD chip (which has the buffers on-chip)?

Well, I guess that depends on how boutique you want to get with your opamps? I would think 8x TL-074 low noise quads should be sufficient for the job? Again, just using Futurlec as a price example (not saying it's the be-all and end-all place to buy op amps!), at $0.40 per DIP TL074s ($0.20 if we go surface mount), the savings is still pretty hefty…. Considering there is almost no board space for the actual MT chips, 8 quad buffers wouldn't be too out of line in terms of board size?

I'm no guru by any stretch…. But I note in the MIDIBox Mixer project there seems to have been a consensus to go with the OPAx130 series? A little more "boutiquey" than an 074 (probably more expensive as it has that "Burr-Brown" badge of honour). BUT, in the DIP version of OPA4130 (quad) the pinout is the same as the 074, so I guess barring any instability in the OPAs requiring caps to prevent oscillation or what not, you could use the same layout and try either / or depending on your tastes? Remember, many commercial products use lowly 4558s for both amplification and buffer duties, so an 074 HAS to be better than those!

A large number of op-amp designs use same pin outs. I think for a time there was a bit of "me too", "mine's the best for everything" going on in the op-amp world, and the manufacturers used same pinouts to ease prototyping headaches. I think the idea of using a seperate buffer board is a good call, that way if some crazy audiophile or guitar hero wants to use discrete OTAs, germanium transistors, NOS radio valves from a soviet submarine pre-amp, they can.

Since you're going with the buffers – unity gain did you consider adjustable gain controls so input and output gain can be mixed properly? Maybe using DACs or digital pots for the control? So chains of patches can be mixed and saved?

Matrix Board

Possibly, we could also modular-ise the matrix board, say putting a 16×8, or 2 16x8s on one board, with input thrus and output nodes to connect more matrixes as needed. Make the board to get around the one-set-of-inputs, one-set-of-outputs thing.

```
                Inputs
  1-8     9-16    17-24   25-32
    /\       /\       /\       /\
   |  |     |  |     |  |     |  |
   |  #---- |  #---- |  #---- |  #------- 1 -16     Outputs
  |        |        |        |
  #--------#--------#--------#---------- 17-32
```

Each set of 8 inputs is split in two and goes to two chips in parallel, so that each set of 8 inputs is capable of being routed to any one of the 32 outputs, which are in two groups of 16.

The boards are intentionally designed to be modular, and so form a 'macro matrix' of boards joined in x/y directions. The pin headers on the boards themselves are oriented to facilitiate this very easily using IDC headers on ribbon cable. This means the design is fully scalable from 16×16 to 16nx16n

Presets

I think 128 presets is always a good 'round number (not sure what a bankstick could take in terms of the theoretical, but cacaphonous, possibility of all 32 ins routing to all 32 outs (1024 connections) on a 32×32, let alone addressing larger crosspoints!). That would probably be a maximum for most.... 64 is OK, even 32 is reasonable with a memory dump feature (and might allow for better usage of memory).

Insert Function Another UI idea I had was an 'insert' function... Say you have 's2k_6' routed to 'mix_ch28'. You want a delay on that sampler. You select your destination 'mix_ch28', which is already patched to source 's2k_6'. You select the new source, 'Delay2_1', and by using a key combo (hold down preset button while hitting On/Off?) you are prompted for another destination. You select 'Delay2_1' (same label, but remember one's an input and another's an output) and it not only redirects the mixer to receive from the delay, but also redirects the delay's input selected in the prompt, to receive from the sampler.

Use cases OK seeing as I'm thinking about use-cases... Source and Destination makes sense from the matrix perspective, but in the real world we'll be patching “output” to “input” not “source” to “destination”. Maybe we could have a #define that the user can edit, so that they can change it as they see fit. I'd rather my screen say something like :

To 'mix_ch28' Input From 's2k_6' Output

than

Dest 'mix_ch28' Src 's2k_6'

Just because those are the words I would use in real life. Anyway such things are probably too fine a detail at this point...

MIDI Specifications

This brings up something we haven't contemplated much - MIDI implementation. Any volunteers? That work will be a matter of talking a lot and typing up lots of pretty documents.
MIDI implementation of a device usually specifies in detail which MIDI commands are recognized by the device and what the response of the device will be.

For a crosspoint matrix audio/FX patchbay that could be:
- MIDI CC messages for setting switches / selecting FXs: Describe in detail how this is done.
- MIDI program change messages for selecting patches: Describe in detail how this is done.
- What else?

<box 45% blue left> *What about this idea? Its pretty simple to set up on nearly every device I think. cc X: select input (0-127) cc Y: select output (0-127) cc Z: on/off (0-63 off; 64-127 on)*

for example: (x/y/z would easy be changeable in a config menu of the router.) cc 80: select input (0-127) cc 81: select output (0-127) cc 82: on/off

*cc value action ========== 80 27 prepare input 28 for action 81 31 prepare output 32 for action 82 63 switch off!
———————————————— 81 65 prepare output 66 for action 80 121 prepare input 122 for action 81 66 no, better prepare output 67 ;) 82 64 switch on!*
========== </box> <box 45% green right>

Example from the Switchblade manual, page 9: RELAY STATE CC# CCvalue 1 OFF 113 0 1 ON 113 127 2 OFF 114 0 2 ON 114 127 3 OFF 115 0 3 ON 115 127 4 OFF 116 0 4 ON 116 127

I think it will pay to consider the devices that are being used to send the controls. For example you mentioned that guitar pedal boards send CC's, maybe they can send notes too,

I don't care if we switch FX with Notes, but CC messages are easier for guitar players and most pedal systems send CC messages. </box>

Hardware Choices

Crosspoint Switches

Matrix IC's available for this project:

<box 99% left orange|AD8113 infos> ***Audio/Video 60MHz 16 x 16 Crosspoint Switch*** This IC from Analog Devices



The AD8113 is a fully buffered crosspoint switch matrix that operates on ±12V for Audio applications and ±5V for Video applications. It offers a -3dB signal bandwidth greater than 60MHz and channel switch times of less than 60ns with 0.1% settling for use in both analog and digital audio. The AD8113 operated at 20kHz has crosstalk performance of -83dB and isolation of 90dB. In addition, ground/power pins surround all inputs and outputs to provide extra shielding for operation in the most demanding audio routing applications. The differential gain and differential phase of better than 0.1% and 0.1°, respectively, along with 0.1dB flatness out to 10MHz make the AD8113 suitable for many video applications. The AD8113 includes 16 independent output buffers that can be placed into a disabled state for paralleling crosspoint outputs so that off channel loading is minimized. The AD8113 has a gain of +2. It operates on voltage supplies of ±5V or ±12V while consuming only 34mA or 31mA of current, respectively. The channel switching is performed via a serial digital control (which can accommodate daisy-chaining of several devices) or via a parallel control, allowing updating of an individual output without reprogramming the entire array.

it is quite expenisve but if this works it will open up a lot of possibilities </box>

<box 99% left orange|AD75019 infos> ***AD75019 16 x 16 Crosspoint Switch*** The AD75019 contains 256 analog switches in a 16 ´ 16 array. Any of the X or Y pins may serve as an input or output. Any or all of the X terminals may be programmed to connect to any or all of the Y terminals. The switches can accommodate signals with amplitudes up to the supply rails and have a typical onresistance of 150 W. Data is loaded serially via the SIN input and clocked into an onboard 256 bit shift register via SCLK. When all the switch settings have been programmed, data is transferred into a set of 256 latches via PCLK. The serial shift register is dynamic, so there is a minimum clock rate of 20 kHz. The maximum clock rate of 5 MHz allows loading times as short as 52 ms. The switch control latches are static and will hold their data as long as power is applied. To extend the number of switches in the array, you may cascade multiple AD75019s. The SOUT output is the end of the shift register, and may be connected to the SIN input of the next AD75019. </box>

<box 99% left orange|MT88xx infos> ***MT88x Crosspoint Switch*** Well, the MT ones seem to have good crosstalk specs when you keep the impedance low. Also, the X and Y lines are I/Os, there's no assigned input and output pin. They say you can connect the crosspoints in any way you want. It seems more flexible than the 8113. Also, it's not only available as an LQFP package. The only disadvantage I see is an higher THD. </box>

Buffers

The buffer stages are on seperate boards, so you can use them or not. My sense of this is that for my own purpose (mainly studio use, potential for a second for just guitar or even more for a modular synth with patch recall) - the buffer boards will come in handy for splitting signals etc, so as to not load the effect devices excessively.

Control methods

Firmware/Software Attributes

Discussion on crosspoint interface

Hmm you really need to store the position of all the switches I think… Although you could maybe just store the On switches, and clear the matrix when loading a preset… I think that storing all positions will be easier to handle. Given that memory is not a concern…

Well, the way it's arranged is per-chip. 16 bits are used to store the switch states for each row (Y pins on the chip). Each bit represents the state of the switch to the columns (X pins) There are 16 sets of these for each chip. It counts from the last Y pin (y15) and last X pin (x15), then second last X pin (x14), third last (x13)… until it gets to the first X pin (x0), then it starts again on the second last Y pin (y14) to x15, x14, x13…x0……etc….. If there are multiple chips, it starts at the last chip and works it's way up. This means we'd have 256 bits per chip. Obviously we would tell the app how many chips we are running, so it knows to read 256* that many chips from the bankstick.

Sidenote: looking at the timings, there are no maximums for loading the individual bits into the IC, so we probably won't even have to buffer it, just read direct from the bankstick to the chip (OK, it makes sense to use pageread to grab big chunks from the bankstick, but anyway, no need to buffer the whole thing). The only maximum is for the PCLK strobe (which tells the IC “I'm done sending you data, now load it into the switches”) which has a 5ms max - plenty. The minimums are actualy kinda long, I can see us needing a few NOPs in the driver Essentially it goes:

Set the SIN (data input) pin according to the switch position. Hold 20ns min.
Set SCLK (serial clock) pin high. Hold 100ns or more.
Set SCLK (serial clock) pin low. Hold 100ns or more.
Rinse. Repeat.
Allow max 5ms wait here, no more!
Set PCLK (Parallel Clock) pin low. Hold 65ns or more.

Minimum SCLK speed is 20khz, no problems there, might be smart to disable interrupts while it's loading though.

If we only stored the switches set On, then we'd have to calculate all these bits before sending to the matrix. That's not too hard, don't get me wrong, but it is kinda indirect. Also, storing all the bits makes it easy to look up the switch's state - you just jump to the appropriate bit, and see how it's set. If we only store the connected pins, then we would have to read them all, to check. Again that's not hard, and wouldn't take long especially if there weren't many connections, but it is indirect.

More… The MidiBoxers building such a router can create an additional MIOS-Core with the Midio128 application and their own design/implementation for the UI. Now if they want to load snap1, they just have to send Controller xy “127” to the MIDI-In of the RouterCore and “snap1” gets loaded and executed, no matter how they made their MIDIO128 to send the message.

In the aaplication of the RouterCore, we just would implenet the rules and handling. Like for source/target routing:

e.g. connect “sourceX” to “targetY”, the receved messages from the controllerUI would have to be like:
- ControllerY “127”
- ControllerX “127”
#if ControllerY receives “0” before ControllerX received “127”, no connection is executed
#→ existing connection of target gets overwritten (disconnected)
#→ Connection established
#→ LED XY “127” (gets sent back to the controller(s)
- ControllerX “0”

- ControllerY “0”

Presets and patches and inserts You create a patch record, which patches an input to an output. This record also allows adjusting the gain of the patch.
You can create patch records with duplicate inputs or outputs. This allows for multiple inputs to multiple outputs.
Example Patch Records.
SOURCE DESTINATION GAIN
instrument distortion 0dB
instrument delay 0dB
distortion amplifier 0dB
delay amplifier -12dB
Get the picture?

A collection of patch records is a single preset.
Presets are saved and recalled as program changes.

Each input and each output has a record.
You name the input or output and you can assign a midi channel.
Inputs and outputs with the same name are considered a loop, which can be switched in an out with a CC command.
These records are global to the system.

Inputs Jack Device 01 GUITAR 02 DISTORTION 03 DELAY Outputs Jack Device 01 AMP 02 DISTORTION 03 DELAY 04 AMP2

wishlist/ideas (feel free to add yourself)

- It might also be worth considering switchable +4db/-10db pads on the ins and outs.
- As an aside, I should mention that I'm looking at something that can route CV and/or audio.
- Think bigger dude. Think 64×64. Think 128×64. Think 256×256
- A LED matrix which shows the routing status and x/y buttons (like MB-SID) for fast routing

Final Hardware Design

I have already done layouts for:

Input Buffers Board of 16 inputs (Balanced input, unity gain, suitable for balanced line or guitar use (I think- pending testing))
Output Buffers Board of 8 outputs (Unfortunately due to it being done on Eagle Light with size restrictions, etc. again unity gain)

As well as crosspoint matrix boards for
MT8816 DIP 40
AD 75019
I also started a board for MT8816 PLCC 44, but this is a pretty horrible package layout to work with.
I also did a board for the AD 8113, but we all seemed to go cold on it largely due to price.

Final Software Design

Beware of the zipper. If you're not carefull, when you load the matrix you'll get ziiipppppppp. Also, does it load fast enough to switch in a single guitar effect while playing?

It's switches are buffered up (well, latched) so only the ones that change are flipped (IE, if you tell it to be closed, and it's already closed, it won't change) and they all change near-simultaneously. The total load time is in the 4-8ns region (typical to max, so should be 4ns mostly) so that's not too bad.

Other Discussions

There was some discussion awhile ago on this here:

http://www.midibox.org/forum/index.php/topic,10247.0.html

I'm having trouble finding info on the Docmatrix project.

External links

Cascade Mixer http://www.cgs.synth.net/modules/cgs43_cascade_mixer.html
C.V./Audio Mega Mixer http://www.cgs.synth.net/modules/cgs23_cv_mega_mixer.html
D.C. Mixer http://www.cgs.synth.net/modules/cgs04_mix.html
Dev-mod Mixer/Inverter block http://www.cgs.synth.net/modules/cgs70_mix.html
5 x 5 Matrix Mixer http://www.cgs.synth.net/modules/cgs33_matrix_mixer.html

http://www.circuitcellar.com/library/print/hcs-pdf/45-Ciarcia.pdf
http://www.soundsculpture.com/products/switchblade.htm
http://synth.stromeko.net/DIY.html#Matrix32x32

Audio/Video Crosspoint Switch - This is a really good one! It's a PDF thesis of smeone who built theirs at university, complete with code, schems, layout etc, and most importantly discusses his experiences during the build, which is invaluable info at thi stage.
http://innovexpo.itee.uq.edu.au/2003/exhibits/s363125/thesis.pdf

This is a discontinued concussor module.
http://www.users.globalnet.co.uk/~concuss/concussor/pp.htm

Bend Matrix - This is another audio switching matrix project using the MT8816 chip. Only 4×8, but could be 8×16 without much hassle. An AVR chip handles MIDI in/out.
http://www.commonsound.com/kits/doku.php?id=commonsound:bendmatrix

Please note: This page was started by someone else and I am simply cutting and pasting certain topics from the forum page into this wiki to summerize this project.

From:
https://wiki.midibox.org/ - **MIDIbox**

Permanent link:
https://wiki.midibox.org/doku.php?id=midibox_audiomatrix

Last update: **2011/09/17 17:04**