Version 4.22, Jan. 26, 1999

- Changes "pr.h" to "vpr\_types.h" and "ext.h" to "globals.h".

The old names were around just for historical reasons; the new filenames

are more clear.

- Changed the input netlist format slightly so that subblock output pins

can be hooked directly to unused (open) CLB output pins. When the netlist

specifies this, it means that a CLB OPIN of that class must be used by

that subblock, since the subblock is directly connected to an OPIN. This

correctly models Altera LABs, for example. The router has been altered so

that it ensures any "locally used CLB OPINs" specified by the netlist are

properly reserved (i.e. are not used by other connections).

This change lets me correctly model both types of CLB output pin logical

equivalence. One equivalence type corresponds to there being muxes between

a set of subblocks in a logic block and a set of CLB output pins. In this

case, if some of the subblock outputs don't have to go outside the CLB,

the other subblocks can use more than one CLB OPIN if it helps routability

(for routing a high-fanout net for example). There is another type of

logically equivalent outputs, however. If the subblock outputs are hooked

directly to the CLB outputs, but a set of subblocks are all identical and

muxes let you make any connection you want to their inputs, then all

the CLB OPINs connected to these subblocks are logically equivalent. In this

case, however, a subblock whose output is only used locally (never goes

outside the CLB) still consumes the OPIN connected to it, so other

subblocks can not use this OPIN. This is the case in Altera LABS, for

example. The changes I've made let the netlist specify either type of

logical equivalence (or even a mix of both types).

- Made the delay of a routing switch 1 second when doing a global routing

(old value was 0). A routing resource must have a delay > 0 for the

timing-driven router to work properly.

Version 4.21, Nov. 19, 1998

- Freed the button array in close\_graphics (graphics.c) -- prevents a small

memory leak that would otherwise occur if VPR started and shut down the

graphics window several times. Noticed by Paul Leventis.

- Changed place\_cost\_type from int to an enumerated type (just for

cleanliness).

- Changed read\_arch.c so purify doesn't complain about me printing

out unset values in print\_arch (the values were uninitialized only

when they weren't relevant for the architecture used, so this is

just a cosmetic cleanup).

Version 4.20, Sept. 10, 1998

- Simplified equation used to count transistors in pass transistor muxes --

Sandy noticed that the "approximate" equation I listed was actually exact,

and simpler than the summation used in VPR.

- Changed all "class" variables to iclass so the code compiles with a C++

compiler. Paul Leventis wanted this.

- Changed draw.c to correctly draw architectures in which connected x-directed

(or y-directed) wire segments can overlap some in a channel. Needed by

Steve Trimberger for some of the architectures he's looking at.

Version 4.19, July 7, 1998

- Put an fabs around one of the tests for perturbing an input switch

pattern (in rr\_graph.c). The lack of an fabs meant that I was

perturbing the switch patterns half the time when I didn't need to.

That could have hurt routability a little bit, although probably not

a whole lot.

- Made a CLB output pin driving a global net only a warning when the

CLB pin isn't global. This allows locally generated clocks to be

generated by CLBs, and put on global resources. Paul Leventis wanted

this so he could run one of his new circuits (des) without changing the

architecture file around.

Version 4.18, June 1, 1998

- Changed the rr\_graph generator so that it perturbs the input pin switch

pattern whenever the Fc\_input is a perfect multiple of Fc\_output (so that

the output and input switch patterns are prevented from lining up perfectly).

- Added -timing\_analysis\_only\_with\_net\_delay <float> parameter. This let's

you just use VPR's timing analyzer from the command line, with the delay

of each net set to a constant and other delays taken from the architecture

file. Sandy and Catherine Wong both wanted this.

Version 4.17, April 30, 1998

- Fixed a minor problem in the rr\_graph generator -- the perturbed IPIN

connection block routine could create two switches from the last track

in a channel to one IPIN under certain (weird) conditions (basically very

low track counts). Fixed it by changing a min to a mod.

- Changed the information printed out about the critical path slightly -- now

counts of normal nets and of global nets (i.e. the clock) on the critical

path are computed and printed out separately.

- Cleaned up the code some by making chan\_width\_io, chan\_x\_dist and chan\_y\_dist

part of the chan\_width\_dist structure and passing that structure around to

the places that need it, rather than having them be global variables.

These variables weren't used widely enough to justify their being global.

- Cleaned up the code a bit by moving a few functions that didn't really

belong in place.c into place\_and\_route.c. This file now contains the

overall control routines that start the placer and router, etc. Place.c

now just has the placer itself in it.

Version 4.16, April 20, 1998

- Fixed bug where architectures with both pass transistors and buffered

segments, and paths from buffered to unbuffered segments and back again,

could go into an infinite loop in the router. Problem: the cost is not

monotonic in my directed-timing driven router under certain cases. The

cost of an unbuffered node can be less after you go from it to a buffered

segment and back, due to the reduction in upstream resistance. To fix it,

I remember the backward\_path\_cost back to the start of the routing

connection, and never allow a node to be re-expanded if it would make the

backward\_path\_cost go down. Now loops can never occur.

Version 4.15, April 15, 1998

- Added extra info to the timing graph so I can tell what kind of resource

each timing node is. This let's me print out and display the critical

path. Added code to print out the critical path and graphically display

it.

Version 4.14, April 9, 1998

- Added the optional keyword "global" to CLB inpin statements in the .arch

file. This let's pins used only for global signals (e.g. clocks) be

flagged as special pins that shouldn't connect into the general purpose

routing. This stops muxes, etc. from being built for them, so the area

numbers are more accurate. Also, it means they don't disrupt the input

pin switch pattern -- since input pin switches are evenly distributed

across the tracks, for low Fc's hooking the clock ipins into the normal

routing meant some tracks connected to the clock pin and not to any of the

normal ipins.

The netlist checker has been beefed up to check that global signals only

connect to global CLB pins and so on.

- Changed the switch pattern used for CLB input pins when Fc\_input = Fc\_output.

The old VPR would make perfectly regular switch patterns, so when Fc\_input

= Fc\_output, an output could only talk to certain inputs if Fc was low.

This would result in an FPGA with very poor routeability (at least if

the switch box was subset (planar) -- the Wilton switch box would probably

work OK even with this IPIN switch pattern since it let's every track get

to every other one).

The change checks for the case Fc\_input = Fc\_output and perturbs the switch

pattern for CLB input pins to make it different than the output pin pattern.

Hence an output can talk to more input pins, and the FPGA is more

routable. The perturbation is as small as I could think to make it, since

I still wanted the switches nicely distributed over the tracks and logically

equivalent inputs on the same side should try to hit different tracks.

Version 4.13, March 26, 1998

- Changed the router defaults (acc\_fac, first\_iter\_pres\_fac, pres\_fac\_mult)

to the values my experimentation determined were the best (from experiments

on a buffered, unit-length wire architecture). Also, the timing-driven

router now uses acc\_fac = 0 for the first router iteration.

- Changed the area model slightly. It now assumes that the buffers from tracks

to ipin\_cblocks are 4x minimum width, as are the buffers from the output

muxes to logic block input pins.

- Changed rr\_graph\_timing\_params.c so it shares the pull-up, pull-down part

of tri-state buffer switches in the routing. This means the input capacitance

of a buffer is added into a node's capacitance only once at a given (i,j)

location for that node, since other switches at the same spot using buffers

will share that buffer.

Version 4.12, Jan. 26, 1998

- Added code to allow several different base cost types; some appropriate

for area-based routing, some appropriate for timing-driven routing.

- Wrote the timing-driven router. It uses an A-star directed search algorithm

and keeps track of the Elmore delay of each node in the partial route

tree as it constructs it. Everything is done except the dynamic base cost

changing net by net.

- Split route.c into route\_common.c, route\_timing.c and route\_breadth\_first.c.

- Code cleanup. Removed the net\_block\_pin\_num array and made that data a

part of the net structure (member blk\_pin). Changed the name of net.pins

to net.blocks. Removed the net.tempcost and net.ncost members (since they

were only used by the placer) and added two static arrays to place.c instead.

- Changed rr\_base\_cost to a member of the rr\_indexed\_data structure. Anything

that is the same for all segments of a given type can be stored in this

structure to save data. Put various timing values and such for quick

computation of expected costs to a target here for use by the timing-driven

router.

- Changed segment\_stats.c to use the length information stored in rr\_indexed\_

data, so it doesn't need local static stuff anymore.

- Fixed a minor bug in read\_arch.c -- I wasn't setting the loneline parameter

properly when setting up an FPGA architecture for global routing only.

Thanks to Russ Tessier at MIT for finding this.

Version 4.11, Dec. 4, 1997

- Stopped storing the cost of an rr\_node in the router, and now store only

acc\_cost and pres\_cost. This enables dynamic costing of resources by only

changing the rr\_base\_cost array. The cost of a node is then computed as each

node is expanded during routing. This slows the router down by 7%.

Version 4.10, Dec. 3, 1997

- Changed the base cost of an rr\_node to be looked up through an cost\_index.

Now every wire of a given type (e.g. x-directed, segment type #3) has the

same cost\_index, so to change the cost of all segments of this type during

routing you just change rr\_base\_cost[cost\_index]. This will make dynamic

resource cost changes efficient.

- Deleted the rr\_node\_cost\_inf structure, and moved its members into the

rr\_node (for info needed outside the router) and rr\_node\_route\_inf (for stuff

needed only within the router).

Version 4.00, Beta #6, Nov. 5, 1997

- Added code to build a timing graph of the circuit and do path analysis.

Both the critical path and all the net slacks are determined.

- Split the code that checks the netlist for validity off into a separate

module, check\_netlist. Beefed up the error checking so that all the

connections between subblocks and clbs and amongst subblocks are checked

for errors.

- Changed the net\_pin\_class data structure to the net\_block\_pin\_num data

structure -- it now stores the pin number to which the netlist says each net

should connect at a certain block. For routing I really just have to connect

to a pin of this \*class\* on the specified block, but for timing analysis I

really need to know the actual pin specified to build the timing graph

correctly. For the places where I want to check the class of what I have

to connect to, I just have to take clb\_pin\_class[pnum], where pnum comes from

this new structure.

- Changed the timing analysis parameters specified in the architecture file

to make them easier to understand and more flexible. Each subblock can

now have its own delays (different from that of other subblocks).

- Changed the input netlist format slightly so that subblock outputs that

are routed within a subblock to the inputs of other subblocks don't have

to connect to any clb output pin.

Version 4.00, Beta #5, Sept. 19, 1997:

- Made an Elmore-delay based net delay calculator (in net\_delay.c).

- Restructured routing\_stats (in stats.c) slightly to make the control routine

simpler.

- Sept. 19, 1997: Incorporated Zoom Fit code of Haneef Mohammed at Cypress.

Makes it easy to zoom to a full view of the graphics.

Version 4.00, Beta #4, Sept. 12, 1997:

- Added a Congestion button to the graphics display. It shows all the

overused routing resources in red. This required some changes to

graphics.c to allow buttons to be created and deleted on the fly.

- Made the coordinates (e.g, 4,3) of the block you click on appear in the

text area of the graphics display. Requested by Jordan and Yaska.

Version 4.00, Beta #3, August 28, 1997:

- Added code to make VPR suitable as a CPU benchmark for the SPEC consortium.

Add the line -DSPEC in the makefile to turn on the SPEC code. Normal

users of VPR should never define SPEC, so you can ignore all the code

after an #ifdef SPEC line.

The SPEC specific stuff doesn't do much -- it just rips out the

graphics and makes sure different roundoff behaviour on different

machines doesn't lead to different routings.

- Added a "NO\_GRAPHICS" define. If you define NO\_GRAPHICS in pr.h, all the

X11 code is ripped out allowing VPR to be compiled on machines without X11

graphics.

- Wrote my own random number generation routine, based on rand. It's tuned

for speed, not spectral quality. This lets me get rid of the ARCH\_TYPE

flag, and makes the code more portable across architectures.

- Added a segment usage statistics routine. This routine needs to know

what the type and length of each track of an FPGA is. The rr\_graph

builder saves that information once it has built the graph. If you don't

want to generate this extra information (because you're building your

own graph), just don't call the segment usage routine after routing.

- Moved around some of the rr\_graph data elements. rr\_node\_draw\_inf is

now gone -- .type and .ptc\_num are now part of rr\_node. rr\_node gives all

the physical information about a node except its capacity (which is still

in rr\_node\_cost\_inf). I also moved .prev\_node, .prev\_edge, .cost,

.target\_flag, and .path\_cost to a new structure, rr\_node\_route\_inf. This

structure is local to the router, and is allocated, initialized and freed

by the router, so you don't have to worry about it. Just load up

rr\_node and rr\_node\_cost\_inf in the rr\_graph.c routines.

- Added a transistor based area model. It goes through the rr\_graph and

figures out how many minimum width transistor areas are needed to build the

FPGA routing.

- Changed rr\_graph generator so that pass transistors always create two edges

in the graph. This makes the area model etc. a lot easier.

Version 4.00, Beta #2, July 15, 1997:

- Changed rr\_graph so that IO pads now have all the switches necessary to

be both an input pad and an output pad built. The EMPTY\_PAD rr\_type has

been eliminated. This change doesn't make any difference to the router,

but it will give more accurate transistor counts.

- Changed the rr\_graph to allow each switch coming off a node to be a

different type. Changed the architecture definition file format so that

each .segment line specifies the type of switch used by a wire to drive

that segment, and the type of switch used by an output pin to drive that

segment.

Version 4.00, Beta #1, July 3, 1997:

- Removed router limitation that it couldn't route more than 1 pin of a net

to the same sink (same pin class on a logic block). For the LUT-based

logic blocks I look at there should never be more than one pin from a net

to any sink, but for other logic blocks this could occur. I still warn

the user about this though, since most logic blocks shouldn't have this

feature. I sped up the check for multiple connections to a sink (in

read\_netlist) as well.

- Added ability to graphically see which switches are buffers and which are

pass transistors.

- Added some of the hooks for timing analysis (capacitance loader, timing

parameter parser, etc.)

- Added routing graph sanity checker.

- Added segmentated architecture for routing.

- Fixed the read\_netlist bug where the parser would complain if a subblock

line started with white space (noticed by Kevin Skahill at Cypress).

Version 3.99a

- Added -fast option, and -route\_chan\_width option. -route\_chan\_width

option allows only one routing to be done, rather than a binary search.

- Fixed fact that bounding box clipping of the routing region was too

aggressive (noticed by Russ Tessier).

Version 3.99

- Added D. F. Wong's Universal switch block as one of the switch blocks

supported.

- Code cleanup. Put the placer options in a placer\_opts structure, and

moved the input and output code from place.c to read\_place.c.

- Added the ability to read in the location of the IO pads and force the

placer to keep them there.

- Changed the placement output format slightly. Block numbers are now

only in comments. The placement input parser was completely rewritten.

Comments and line continuation are now allowed in placement files. More

importantly, the order of the blocks is now irrelevant; the file can list

the blocks in any order. This should be a lot more convenient for people

using VPR to route placements generated by other tools. The error

checking has been beefed up too -- I think every possible error in a

placement file will be caught now and the user will get an informative

error message. In particular, the "non-consecutive subblock" error that

tripped up Russ and Ivan will be caught right away.

- Moved all hash table logic out of read\_netlist.c and into a new module,

hash.c. I can use this general hash table stuff all over now.

- Changed defaults to be 30 routing iterations and detailed routing.

- Changed VPR to use a graph-based model of the routing resources available.

This allows VPR to perform either global or detailed routing now. Also

added new graphics options to view the routing resources available.

Took the routing checker out of route.c and put it in check\_route.c.

Also moved some code from route.c to stats.c.

- Converted some defined constants to enumerated types so I can see the

symbolic names in gdb.

- Cleaned up some of the placer code. Centralized allocation of the temporary

placement data structures. Changed the standard deviation computation

so that the old\_costs array wasn't needed anymore.

Version 3.98a

- Made a minor change to the graphics package. Descenders (e.g. "g")

are now considered in the vertical text centering routines.

- Changed the placer inner loop so that the frequency with which recompute

cost is called is variable. Recomputing the cost from time to time

from scratch stops round offs from accumulating. The change didn't make

VPR significantly faster.

- Made -inner\_num a float to allow small inner nums to be tried.

- Code cleanup. Converted the annealing schedule parameters from

global variables to a structure that is passed around. Converted the

graphics state from global variables to static variables in draw.c set

via a call to set\_graphics\_state.

- Added -verify\_binary\_search option. Makes the router try routings

with channel widths of best-2 and best-3 (in addition to the best-1

tried by the binary search). If one of these additional routings

succeeds, the router keeps trying routings until two in a row (e.g.

width = 9 and width = 8) fail. The binary search usually finds the

min. channel width, but due to router flukiness it will occassionally

happen that a width = 19 routing fails, but width = 18 succeeds and

so on.

Version 3.98

- Changed initial routing channel width guess to clb\_size (# of pins

on a clb). Should give slightly faster run times for big clbs.

- Fixed a bug where VPR would wrongly think it had an error and

terminate itself if two outputs were made logically equivalent.

Bug noticed by George Varghese of UC Berkeley.

- Altered VPR to use the new netlist format of blifmap (cluster-based).

In this process I fixed a minor netlist parsing bug where VPR would

refuse to accept a netlist with a comment between a .clb and a .pinlist

line. The new information on what's in a cluster is in the .subblock

lines following each .clb line (after the .pinlist line).

Version 3.23

NB: May want to add check in read\_place for subblock numbers that have

gaps (not consecutive starting at zero). Russ Tessier's suggestion.

(I did this on March 22, 1997 -- Version 3.99).

- Updated graphics again -- made all internal functions static so that

graphics.c is a true component without name space conflicts with user

programs. Also moved the toggle\_nets function to draw.c, as this

cleaned up the code dependencies a bit.

- Corrected RISA reference in the source code to point to ICCAD (pointed

out by Russ Tessier of MIT).

- Rewrote large parts of graphics package. It should now be faster for

zoomed in graphics, as it preclips them based on their bounding box.

This will also make zoomed-in postscript much smaller. The PostScript

driver has been rewritten; output file sizes are about 50% smaller now

if all graphics are on-screen. When zoomed-in the clipping makes an

even bigger difference. Also added the ability to specify the font size

for text and the line width for geometry (I demand-load the fonts

for maximum efficiency while hiding details like font loading from the

calling program). Finally, I removed the global drawscreen function

and replaced it with a callback function to clean up the code a bit.

Version 3.22

- Changed some code in util.c that HP's C compiler didn't like.

Turned on all warning options in gcc and fixed all warnings,

unused variables, etc. Also slightly altered the header structure

of the program to better check function definitions against

function declarations in util.c and graphics.c. Someday I should

make all local functions static and put declarations for all functions

called across modules in header files. Added an ARCH\_TYPE flag

to the makefile so that machine dependent code (random number

generator right now) is altered merely by changing the makefile.

This version should now work fine on both SUN and HP.

- Fixed a minor error message bug Guy found.

- When there is only 1 location for a clb (nx = ny = 1, with 1 clb)

the placer goes into an infinite loop trying to move the block.

Added a check for this so it now just refuses to place the circuit.

- Added an option to specify FPGA size (-nx and -ny) on the command line

for Guy. ny can now be bigger than nx, so I had to change a couple

of things in the range limiter routines.

- Added -aspect\_ratio option to allow simulation of nonsquare FPGAs.

Cleaned up a few bits of ugly code I found while checking that I always

used nx and ny properly. Had to change the range limiter routine to

make nonsquare FPGAs work properly.

Version 3.21

- Added delta channel width description function. Makes only one channel

extra wide (or narrow).

- Changed channel width code so that io channel widths are no longer

tied to the maximum channel width in the rest of the array.

- Added -fixed\_pins option to lock the pins in their initial random

positions.

Version 3.20

- Added nonlinear congestion option to the placer. Can either replace

the circuit for every channel width or just place once to a suggested

width.

- Changed the router so that it only resets the pathcost element of

channel segments that were reached in the last routing phase. This only

sped up the router by 5% -- rather disappointing.

- Added code to recompute the cost (quickly) after each placement temp.

is finished so the check\_cost checks won't fail due to round off

accumulation.

Version 3.13

- Fixed bug in netlist parser that went into an infinite loop if a global

net was not the first item in it's hash table entry.

- Added -place\_cost\_exp option to set the exponent to which we take the

average number of tracks per channel factor in the cost function. Setting

this option to be greater than 1 allows more sever penalization of narrow

channels during placement.

- Made Pathfinder algorithm with init\_pres\_fac = 0.5 and acc\_fac\_mult

= 0.2 the default for the router. Gives about 9% better results than

the old router options.

- Changed the -pin\_block\_update\_type option to -block\_update\_type and

allowed it to take a value of "pathfinder". This invokes a true

Pathfinder algorithm, where some costs are updated immediately and

others are updated after all nets are routed. This Pathfinder

algorithm is new this version.

- Fixed a minor bug in the placer cost function that led to incorrect

bounding boxes (and hence costs) of nets that had more than one pin

connected to the same block. The initial bounding box set up

overcounted the number of blocks on the bb edges. This bug caused

an error message in Mike's big run of vpr on all mcnc circuits.

Version 3.12

- Added a new option, -pin\_block\_update\_type, to allow selection of a

mixed Pathfinder, immediate-update algorithm where pin costs are updated

immediately and channel costs are updated in the Pathfinder manner.

- Now computes and prints out routed wirelength statistics (requested by

Mike).

- Fixed a minor bug in the routine that counted the number of bends in

the routed circuit. The bug counted an extra bend in rare cases

(specifically when wires came out of an OPIN into more than one channel),

causing the bend numbers to be about 1% higher than they should have

been.

Version 3.11

- Added a few more routing options. Set a couple of defaults to what

I've determined to be better values. I compute statistics on how

many bends there are per net. This version is going to Jonathan.

Version 3.10

- Changed the bounding box update method so that large nets have their

bounding boxes incrementally updated whenever possible. This has

sped up the bounding box calculation for bigkey\_mod by a

factor of about 34. For other netlists with fewer high fanout nets

the improvement is smaller, but still significant.

- Changed the cost function calculation so that as much data as possible

is precomputed. This and the bounding box change together speed

up placement of small circuits (e64) by 2x, and large circuits (alu4)

by 15x.

- Added more options to allow more control over the router.

Code maintenance changes:

Restructured the options parser to make it more modular. Added checks

to determine that the parameters supplied for each option are of the

correct (integer, float or string) type.

Version 3.02

- Changed the cost function's for the router. Pins that are not

oversubscribed now have a cost of 0., not 1. The costs of all pins

should be 1 less than in version 3.01. Channels are penalized for

overuse via formulae like cost = 1. + (chan\_occ - chan\_width) \* fac

rather than the old cost = 1. + (chan\_occ / chan\_width - 1.) \* fac.

This new cost penalizes absolute rather than relative overuse, and

should be faster to compute and lead to larger penalties (regardless

of channel width) with smaller penalty factors, fac. It's magnitude

should be more in line with that of the pin overuse cost.

The one drawback is that the relative overuse cost considered using

11 tracks in a 10 track channel to be less severe than using 2 tracks

in a 1 track channel, which is probably the way it should be.

Version 3.01

- Allocated small chunks of memory with calls to my\_small\_malloc instead

of my\_malloc. My\_small\_malloc is my custom memory handler that

avoids the 8-byte or so storage overhead of malloc, at the cost of not

keeping enough information to free the data structures.

- Removed the temporary code from the routing serial number generator,

routing print routine, and channel cost and occupancy dumping routine

that allowed direct comparison with Version 2.22a's results. All

information is now printed with the true (new) pin numbering scheme.

- Removed temporary code that set CLB's opin cost to 0. Now it is 1;

it shouldn't make much difference.

Version 3.00

This version evolved from Version 2.22, not 2.22a. The changes are:

- Input netlist format changed from blif to that produced by blifmap (.net).

- Architecture description file is changed to allow the definition of more

arbitrary clbs, with general classes of logically equivalent pins.

- The entire program has been altered to place and route general clbs rather

than just LUTs with one output.

- The program allows global nets to be flagged via .global statements in the

input netlist file. These nets are ignored in all subsequent place and

route steps by the program, but the blocks they must connect are listed

in the final routing output.

- The routing output now goes to a file, which is specified on the command

line.

- Added two new command line options -route\_only and -place\_only. If

-route\_only is specified, an existing placement is read in from the

place\_file and is routed. If -place\_only is specified, the program

quits after writing out the final placement of a circuit (no routing is

done).

Bug Fixes and minor alterations:

- Changed the chan\_y.occ vs. chan\_width\_y check in check\_routing. It

had the wrong subscript (this would have been bad!).

- Changed the placement output file to include subblock information for IOs.

- Increased the input buffer size for all parsers and added a check for

lines so long that they overflow the input buffer (motivated by J.P.).

- Changed binary search for minimum necessary track width so that tiny

circuits that succeed with a width\_factor of 0 are properly handled

(motivated by J.P.).

- Fixed minor mistake drawing to the OPINs of IO blocks.

- Partially fixed the inaccurate drawing problem caused by X Windows'

apparent inability to handle coordinates outside the range -15000 to

15000. I clip point by point to the -15000 to 15000 range. This means

drawrect, fillrect and drawtext will always work properly. Drawline

(fillpoly) will always work for horizontal or vertical lines (polygon edges)

but will be incorrect for extremely zoomed diagonal ones. Drawarc will

also be incorrect at extreme zooms. The PostScript output always looks

correct. The only complete fix for this problem is for me to write my

own clipping routines. This is a lot of code, and will hurt performance

since I'll be duplicating X Windows' own checks. I don't think it's

worthwhile to overcome a problem that only shows up when people zoom

in on the graphics absurdly (like a bored Steve, who found the inaccurate

drawing).

- Routing serial number generator now only considers the pin numbers of

traceback segments that are of type OPIN or IPIN.

Version 2.22a

Made some minor changes to Version 2.22 to allow this program's output to

be compared to that of Version 3.0 in order to be sure the programs work

the same way. Fixed a nasty bug in check\_routing where the chan\_width\_y

not exceeded check was incorrect. Changed the serial number generation

to avoid taking the pin numbers of things that were not IPINS or OPINs.

Version 2.22

Same as version 2.21 with a few changes to make util.c and util.h a bit

more modular.

Version 2.21

Updated the router graphics, and fixed the unsaved routing bug, and

the improper abort sequence when the PostScript file can't be opened.

This version is fully functional, but can handle only clbs with 1 LUT

in them.

Version 2.20

This version correctly routes nets and updates the cost functions. It

is a complete place and route tool. The router graphics have now been

implemented, and all graphics allow a text message and selection of clbs.