PLACE-AND-ROUTE DESCRIPTION and EXAMPLES

William Bricken
January 2003

OVERVIEW

The encoding strategy and techniques for CoMesh place-and-route are described, along with six example circuits that have been placed-and-routed. No optimizations are included. The place-and-route output is included as an appendix, along with the ILOC circuit forms.

DESCRIPTION

The placed-and-routed layouts in the appendix look like this:

```
CIRCUIT NAME
=============================================================================
=============================================================BLOCK-1
TIER-1     TIER-2     TIER-3     TIER-4     TIER-5     REGISTER-TIER   |
 cell       cell       cell       cell       cell       register       |
 cell                                                                  V
=============================================================BLOCK-2
TIER-1     TIER-2     TIER-3     TIER-4     TIER-5     REGISTER-TIER   |
 cell       cell       cell       cell       cell       register       |
 cell                                                                  V
=============================================================================
=============================================================================
```

Cells are stacked in tiers, reading vertically. A normal block has 16 cells per tier. Blocks are collections of 5 logic-tiers plus one register tier, reading horizontally, for a total of 80 cells and 16 registers. Multiple blocks are stacked vertically and separated by wide lines.

TIMING

For 300 MHz, the register tier in each block clocks out signals every 3.3 ns. 2.2 ns are allocated for within block computation, and 1.1 for between block communication. Given no pipelining, a function that uses N blocks would divide the effective MHz by N. If input signals are available each clock cycle, the pipelined function would require N units of 3.3 ns to set up the first output, and then would operate at 300 MHz as long as input was available.

CELLS

Each cell is identified by a cell-function and a numerical cell-id, separated by a colon:

<cell-function>:<cell-id> example: (b (2)):4

The cell-function is encoded in parens forms. The ids in the cell-function are the cell inputs. The cell-id is the cell output.

The cell-function encodes
- the inputs to the cell as letters and numbers
  - letters are primary inputs
  - numbers are inputs from other cells
- the settings of the cell configuration SRAMs as parens
- the output of the cell as the numerical cell-id
  - cell-ids with asterisks are primary outputs
  - cell-ids with at-signs are feedback register outputs

Briefly wrt SRAM encoding using parens, any label is a positive input if not directly within a parens container, and a negative input if within a parens container. The configuration of parens specify the logic settings.

Cell-functions with special labels are special configurations:

a=b	an NXOR function
(a=b)	an XOR function
c<a>b	a MUX function
(a b c d)	a wide function (here with four inputs)
---	an empty or unused cell

NXOR, XOR, and MUX special cells require two adjacent normal cells. Wide cells nominally require one less normal cells than the number of their inputs. Wide cell routing has not yet been finalized, and may be more efficient (specifically a two-input tree across multiple tiers). Empty cells can be filled with other functions.

TIERS

In this version of place-and-route, each tier has an unlimited number of cells. The width of a tier can be determined by counting cells, with special cells requiring more than one normal cell. A cell can be placed on any tier preceding the tier that uses its output. In order to reduce congestion on particular tiers, cells can thus be moved forward until the output of the cell is needed by the immediately following tier.

Normally a tier has 16 cells. A wide tier can be spread across several blocks, all processing in parallel. Thus wide tiers add area but not necessarily time.

REGISTERS

The sixth tier is a register tier, encoded as

<register-input>:<register-output> example: 14:@3

Register inputs are always single cell-ids. Register outputs are encoded as to whether they feedback into the block, or pass a signal through to the next block.

There are three types of register encoding:
The at-sign register "@id" has its output fed back within the block or within another block used by the same function.
The "out" register is a primary output.
The arrow register "cell-id->block-id" passes its input to a following block.

At-sign registers are normal registers used to feedback signal values. Feedback values occur on the next clock cycle, and may be input to any block used by a function. The output of an at-sign register is encoded to include the identification of the block that uses the output. Encoding uses arrows to identify the direction of feedback/feedforward. All blocks that use a particular at-sign register output are encoded using arrows.

Arrow registers are normal registers used primarily for pipelining between blocks. The output of an arrow register encodes which blocks use the output of the register during the next clock cycle. Backward arrows encode feedback to earlier blocks, forward arrows encode feedforward to later blocks. For pipelining, all registers must synchronize, which means that register forward outputs must pass through every block in lock-step, and cannot skip blocks. Thus some arrow registers merely pass signals through a block to another block later in the processing sequence.

The difference between an at-sign register and an arrow register is that the at-sign register is an explicit register within the design of the circuit and is mandatory to coordinate functional timing of feedback loops. An arrow register is not an essential part of a timed feedback loop, and is required only for CoMesh pipeline timing between blocks.

ROUTING

The cell-input of each cell-function must occur as an output of a prior cell if it is a number. Letters represent primary inputs that are assumed to be available to all tiers. The register tier encodes routing between blocks.

A normal block has limited wiring between tiers, and thus a limited number of cell-ids can enter the next tier as input. As well, switching of cell outputs to next tier inputs is constrained. Registers have limited routing available between blocks. There routing refinements are not included herein.

EXAMPLES

Six examples are attached, three combinational and three sequential. They range from very simple to complex, and are intended to illustrate all the features described above. For comprehension, the large examples do not have a large gate count.

Simple Examples

CON1 is a small combinational logic circuit that fits within four tiers of one block, using less than 18% of the cell resources within a block. It has 7 inputs {a,b,c,d,e,f,g} and 2 outputs {10*,11*}; and consists of 17 2-input ASIC gates, including two MUX gates. The pyramid shape, wide at the first tier and narrowing to the last tier is very common. Outputs {1,3,4,12} from tier-1 are used immediately in tier-2. Output 2 from tier-1 is not used until tier-4, so the corresponding cell could be moved forward to balance cell usage across tiers. Tier-5 is not used at all; it could be moved to any other tier location.

LION is a simple sequential FSM with 2 inputs, 1 output, and 2 registers. It consists of 13 2-input ASIC gates. On each clock cycle, the registers feed back their values into tier-1 and tier-2. LION uses about 17% of the block, and does not use any cells in tier-4 and tier-5.

Two Block Examples that can be Optimized

CM85A is the 4-bit magnitude comparator with 11 inputs and 3 outputs. It consists of 36 2-input ASIC gates. CM85A uses six logic tiers in two blocks. Tier-1 is twelve normal cells wide, while the tier-2 uses only one cell. It is straightforward to combine tier-2 with tier-3, in order to fit CM85A into one block. As presented, CM85A uses 43 cells in two blocks, about 45% of a single block. Seven arrow registers are used to transfer signals from block-1 to block-2.

S208 is an FSM with 11 inputs, 2 outputs, and 5 registers. It consists of 46 2-input ASIC gates. S208 occupies seven logic tiers, with tier-6 and tier-7 easily compressed into tier-5 to fit the function into a single block. S208 begins very wide, with the at-sign registers feeding back extensively into tier-1. To alleviate this congestion, cells {7,27,34} can be moved forward to tier-5; cells {9,11} can be moved to tier-3; and cells {29,32} can be moved to tier-2.

Large Multiblock Examples

COUNT is a combinational counter with 35 inputs and 16 outputs, and consists of 128 2-input ASIC gates. It requires four blocks and is difficult to optimize because there is an NXOR and a MUX in nearly every tier; these cause excessive fan-out. Optimizing COUNT to three blocks is relatively easy, it becomes very wide when optimized to two blocks. Very few signals are passed between blocks. COUNT uses 93 cells, the equivalent of about one entire block. Note the similarity of cell functions in each tier.

MULT16A is an 8-bit multiplier with 17 inputs, 1 output, and 16 registers. It consists of 183 2-input ASIC gates. MULT16A spreads over ten blocks and is quite narrow. Several block outputs are passed through many other blocks until they are needed; these are indicated by an arrow register pointing both directions. Registers form tight loops that make block compression difficult without changing the architecture of the multiplier. Note that the last register in block-10 is @3, which feeds back into every other prior block. This makes the function very difficult to compress. Multipliers like this are implemented as specialized embedded ASICs in most FPGA architectures. Our solution is to construct a multiplier that is ideal for the CoMesh architecture and provide it as a pre-programmed "soft" macro.

APPENDIX: Placed-and-routed Example Circuits

SIMPLE CIRCUITS provide layout format examples.

CON1

```
================================================================================
================================================================================
(b d):1     (12 (d)):9  (5 (b)):8  ((2 (13))):11*  ---   10*:out
(b e):2     (4 (g)):7   7<a>6:13   ((8 9)):10*     ---   11*:out
(a (f)):3   (1 (e)):6   ---        ---             ---   ---
((b)(e)):4  (e 3):5     ---        ---             ---   ---
b<c>a:12    ---         ---        ---             ---   ---
================================================================================
================================================================================
```

LION

```
================================================================================
================================================================================
(b @2):4        ((5 6 @2)):12*  ((3 5)):8   ---   ---   12*:out
(a (@1)):5      (10 (@2)):7     ((7 11)):9  ---   ---   9:@2
(b (@1)):6      (a 4):3         ---         ---   ---   8:@1
(a b @1):10     ---             ---         ---   ---   ---
(b (a)(@1)):11  ---             ---         ---   ---   ---
================================================================================
================================================================================
```

TWO BLOCK CIRCUITS provide examples of inter-block routing, and optimization possibilities.

CM85A

g=h:20	((j)(20)):5	(3 4 (5)):16	((16)(19)):10	(1 2 (10)):15*	15*:out
c=d:19	---	((4)(5)):9	(c (d)(16)):13	((2)(10)):7	7:->02
(h (g)(j)):12	---	((3)(5)):8	(d (c)(16)):14	((1)(10)):6	6:->02
(g (h)(j)):11	---	---	---	---	13:->02
(f (e)):4	---	---	---	---	14:->02
(e (f)):3	---	---	---	---	9:->02
(b (a)):2	---	---	---	---	8:->02
(a (b)):1	---	---	---	---	12:->02
((k 7 9 12 14)):18*	---	---	---	---	17*:out
((i 6 8 11 13)):17*	---	---	---	---	18*:out

S208

(@4 (j)):7	(b 12 22 (a)):31	(b 15):6	(9 11 14):28	(28 (@4)):20	34*:out
((f)(@1)):9	(30 (@2)):21	(13 @5):14	(b 17 (a)):25	---	27:@4
((h)(@3)):10	(22 (@3)):19	(19 29):16	(b 16 (a)):24	---	31:@5
((i)(@5)):11	(18 33):15	(21 32):17	---	---	6:@1
(@4 @5):12	(10 26):13	---	---	---	25:@2
(@4 (@1)):18	---	---	---	---	24:@3
((@4)(@5)):22	---	---	---	---	20:->02
(@3 (g)(@2)):26	---	---	---	---	---
((b (a)(@4))):27	---	---	---	---	---
(@3 (@4)(@5)):29	---	---	---	---	---
((@3)(@4)(@5)):30	---	---	---	---	---
(@2 (@3)(@4)(@5)):32	---	---	---	---	---
((a)(@2)(@3)(@4)(@5)):33	---	---	---	---	---
(@1 @2 @3 @4 @5):34*	---	---	---	---	---
(k 7 20):23	(23 (a)):8*	---	---	---	8*:out

MULTIBLOCK CIRCUITS provide examples of inter-block routing, and of larger functions that cannot be compressed to a single parallel collection of blocks.

```
COUNT
=============================================================================================
=============================================================================================
af=ah:32          (ae (1)):6        ak<ai>37:49       (ac (5)):4        am<ai>35:51       18*:out
(af ah):1         ae=1:37           ad=6:36           ((ag (49))):17*   ab=4:34           17*:out
---               aj<ai>32:48       ((ag (48))):16*   ac=5:35           ((ag (50))):18*   16*:out
---               ---               (ad (6)):5        al<ai>36:50       (ab (4)):3        51:->02
---               ---               ---               ---               ---               34:->02
---               ---               ---               ---               ---               3:->02
=============================================================================================
an<ai>34:52       (bh (2)):15       aq<ai>47:54       (bd (14)):13      au<ai>45:56       23*:out
aa=3:33           ((ag (52))):20*   bf=15:46          ((ag (54))):22*   bb=13:44          22*:out
((ag (51))):19*   bh=2:47           ((ag (53))):21*   bd=14:45          ((ag (55))):23*   21*:out
(aa (3)):2        ao<ai>33:53       (bf (15)):14      as<ai>46:55       (bb (13)):12      20*:out
---               ---               ---               ---               ---               19*:out
---               ---               ---               ---               ---               56:->03
---               ---               ---               ---               ---               44:->03
---               ---               ---               ---               ---               12:->03
=============================================================================================
aw<ai>44:57       (ax (11)):10      ba<ai>42:59       (at (9)):8        be<ai>40:61       28*:out
az=12:43          ((ag (57))):25*   av=10:41          ((ag (59))):27*   ar=8:39           27*:out
((ag (56))):24*   ax=11:42          ((ag (58))):26*   at=9:40           ((ag (60))):28*   26*:out
(az (12)):11      ay<ai>43:58       (av (10)):9       bc<ai>41:60       (ar (8)):7        25*:out
---               ---               ---               ---               ---               24*:out
---               ---               ---               ---               ---               61:->04
---               ---               ---               ---               ---               39:->04
=============================================================================================
bg<ai>39:62       ((ag (62))):30*   ((ag (63))):31*   ---               ---               31*:out
ap=7:38           bi<ai>38:63       ---               ---               ---               30*:out
((ag (61))):29*   ---               ---               ---               ---               29*:out
=============================================================================================
=============================================================================================
```

MULT16A 29

((a)(q)):17	(32)(@5)):154	(31 154):34	(34 153 @6):172	(100 156):68	38*:out
((b)(q)):18	((17)(@3)):152	(35 154):38*	(69 (31)):100	(37 (@6)):138	152:->10
((c)(q)):19	(32 @5):35	(154 @6):69	(34 153):37	---	33:->09
((d)(q)):20	(17 @3):33	((31)(154)):153	---	---	17:->09
((e)(q)):21	---	((154)(@6)):156	---	---	18:->09
((f)(q)):22	---	---	---	---	19:->08
((g)(q)):23	---	---	---	---	20:->08
((h)(q)):24	---	---	---	---	21:->07
((i)(q)):25	---	---	---	---	22:->06-07
((j)(q)):26	---	---	---	---	23:->06
((k)(q)):27	---	---	---	---	24:->05
((l)(q)):28	---	---	---	---	25:->05
((m)(q)):29	---	---	---	---	26:->04
((n)(q)):30	---	---	---	---	27:->03-04
((o)(q)):31	---	---	---	---	28:->03
((p)(q)):32	---	---	---	---	68:->02
---	---	---	---	---	138:->02
---	---	---	---	---	172:->02
---	---	---	---	---	29:->02
---	---	---	---	---	30:->02

(@7 (68)):71	(96 102 @7):185	(101 140):55	((137 185)):170	(95 104):53	171:01<-@5
(30 (68)):96	(71 (30)):140	(54 (@7)):137	(55 (29)):104	(72 (29)):141	170:01<-@6
(68 (@7)):101	(96 102):54	---	(55 (@8)):103	(95 104 @8):184	152:<-02->
(68 (30)):102	---	---	(29 (55)):95	---	33:<-02->
((138 172)):171	---	---	(@8 (55)):72	---	17:<-02->
---	---	---	---	---	18:<-02->
---	---	---	---	---	19:<-02->
---	---	---	---	---	20:<-02->
---	---	---	---	---	21:<-02->
---	---	---	---	---	22:<-02->
---	---	---	---	---	22:<-02->
---	---	---	---	---	23:<-02->
---	---	---	---	---	24:<-02->
---	---	---	---	---	25:<-02->
---	---	---	---	---	26:<-02->
---	---	---	---	---	27:<-02->
---	---	---	---	---	27:<-02->
---	---	---	---	---	28:<-02->
---	---	---	---	---	53:->03
---	---	---	---	---	141:->03
---	---	---	---	---	184:->03
---	---	---	---	---	103:->03

(103 141):56	((136 184)):169	(94 106):52	(52 (@9)):135	(@10 (57)):74	169:02<-@7
(53 (@8)):136	(56 (28)):106	(73 (28)):142	(105 142):57	(27 (57)):93	168:02<-@8
---	(56 (@9)):105	(94 106 @9):183	---	(57 (@10)):107	152:<-03->
---	(28 (56)):94	---	---	(57 (27)):108	33:<-03->
---	(@9 (56)):73	---	---	((135 183)):168	17:<-03->
---	---	---	---	---	18:<-03->
---	---	---	---	---	19:<-03->
---	---	---	---	---	20:<-03->
---	---	---	---	---	21:<-03->
---	---	---	---	---	22:<-03->
---	---	---	---	---	22:<-03->
---	---	---	---	---	23:<-03->
---	---	---	---	---	24:<-03->
---	---	---	---	---	25:<-03->
---	---	---	---	---	26:<-03->
---	---	---	---	---	27:<-03->
---	---	---	---	---	74:->04
---	---	---	---	---	93:->04

```
---             ---               ---               ---             ---               107:->04
---             ---               ---               ---             ---               108:->04
================================================================================================
(93 108):51       (51 (@10)):134  (@11 (58)):75     (92 110 @11):181  (109 144):59     167:03<-@9
(74 (27)):143     (107 143):58    (26 (58)):92      (75 (26)):144     (50 (@11)):133   152:<-04->
(93 108 @10):182  ---             (58 (@11)):109    (92 110):50       ---              33:<-04->
---               ---             (58 (26)):110     ---               ---              17:<-04->
---               ---             ((134 182)):167   ---               ---              18:<-04->
---               ---             ---               ---               ---              19:<-04->
---               ---             ---               ---               ---              20:<-04->
---               ---             ---               ---               ---              21:<-04->
---               ---             ---               ---               ---              22:<-04->
---               ---             ---               ---               ---              22:<-04->
---               ---             ---               ---               ---              23:<-04->
---               ---             ---               ---               ---              24:<-04->
---               ---             ---               ---               ---              25:<-04->
---               ---             ---               ---               ---              59:->05
---               ---             ---               ---               ---              133:->05
---               ---             ---               ---               ---              181:->05
================================================================================================
(@12 (59)):76   (91 112 @12):180  (111 145):60    ((132 180)):165 (90 114):48       166:03-04<-@10
(25 (59)):91    (76 (25)):145     (49 (@12)):132  (60 (24)):114   (77 (24)):146     165:04<-@11
(59 (@12)):111  (91 112):49       ---             (60 (@13)):113  (90 114 @13):179  152:<-05->
(59 (25)):112   ---               ---             (24 (60)):90    ---               33:<-05->
((133 181)):166 ---               ---             (@13 (60)):77   ---               17:<-05->
---             ---               ---             ---             ---               18:<-05->
---             ---               ---             ---             ---               19:<-05->
---             ---               ---             ---             ---               20:<-05->
---             ---               ---             ---             ---               21:<-05->
---             ---               ---             ---             ---               22:<-05->
---             ---               ---             ---             ---               22:<-05->
---             ---               ---             ---             ---               23:<-05->
---             ---               ---             ---             ---               48:->06
---             ---               ---             ---             ---               146:->06
---             ---               ---             ---             ---               179:->06
---             ---               ---             ---             ---               113:->06
================================================================================================
(113 146):61    ((131 179)):164   (89 116):47       (47 (@14)):130  (@15 (62)):79    164:05<-@12
(48 (@13)):131  (61 (23)):116     (78 (23)):147     (115 147):62    (22 (62)):88     163:05<-@13
---             (61 (@14)):115    (89 116 @14):178  ---             (62 (@15)):117   152:<-06->
---             (23 (61)):89      ---               ---             (62 (22)):118    33:<-06->
---             (@14 (61)):78     ---               ---             ((130 178)):163  17:<-06->
---             ---               ---               ---             ---              18:<-06->
---             ---               ---               ---             ---              19:<-06->
---             ---               ---               ---             ---              20:<-06->
---             ---               ---               ---             ---              21:<-06->
---             ---               ---               ---             ---              22:<-06->
---             ---               ---               ---             ---              79:->07
---             ---               ---               ---             ---              88:->07
---             ---               ---               ---             ---              117:->07
---             ---               ---               ---             ---              118:->07
================================================================================================
(88 118):46       (46 (@15)):129  (@16 (63)):80    (87 120 @16):176  (119 149):64    162:06<-@14
(79 (22)):148     (117 148):63    (21 (63)):87     (80 (21)):149     (45 (@16)):128  152:<-07->
(88 118 @15):177  ---             (63 (@16)):119   (87 120):45       ---             33:<-07->
---               ---             (63 (21)):120    ---               ---             17:<-07->
---               ---             ((129 177)):162  ---               ---             18:<-07->
---               ---             ---              ---               ---             19:<-07->
---               ---             ---              ---               ---             20:<-07->
---               ---             ---              ---               ---             64:->08
---               ---             ---              ---               ---             128:->08
---               ---             ---              ---               ---             176:->08
================================================================================================
(@4 (64)):81    (86 122 @4):175  (121 150):65   ((127 175)):160 (85 124):43      161:06-07<-@15
(20 (64)):86    (81 (20)):150    (44 (@4)):127  (65 (19)):124   (70 (19)):139    160:07<-@16
```

```
(64 (@4)):121    (86 122):44      ---           (65 (@2)):123   (85 124 @2):174  152:<-08->
(64 (20)):122    ---              ---           (19 (65)):85    ---              33:<-08->
((128 176)):161  ---              ---           (@2 (65)):70    ---              17:<-08->
---              ---              ---           ---             ---              18:<-08->
---              ---              ---           ---             ---              43:->09
---              ---              ---           ---             ---              139:->09
---              ---              ---           ---             ---              174:->09
---              ---              ---           ---             ---              123:->09
================================================================================================
(123 139):66   ((126 174)):159   (84 99):42        (42 (@1)):125   (33 67):36        159:08<-@4
(43 (@2)):126  (66 (18)):99      (82 (18)):151     (98 151):67     (17 (67)):83      158:08<-@2
---            (66 (@1)):98      (84 99 @1):173    ---             (67 (17)):97      152:<-09->
---            (18 (66)):84      ---               ---             ((125 173)):158   36:->10
---            (@1 (66)):82      ---               ---             ---               83:->10
================================================================================================
(83 97):41      ((41)(@3)):155  (39 155):40  ---      ---      157:01-02-03-04-05-06-07-08-09<-@3
((36 152)):157  (41 @3):39      ---          ---      ---      40:09<-@1
================================================================================================
================================================================================================
```

ILOC INTERNAL FORMS

ILOC internal form encodes functionality as labeled rows consisting of an id and a parens form. The number after the + sign in the id shows the tier that the parens form is placed in. The parens forms have been configured to fit into cells exactly, with the parens containers specifying the settings of the SRAM switches in the configuration memory of a cell. Id labels indicate wires that route signals between cells on different tiers. Special cells have special labels. At the top of each circuit is some information, and the ids of the primary inputs and outputs of the circuit. The information field consists of

```
(name process input-file output-file=nil process-file=not-known process-time date)
```

```
((con1 comesh-place-and-route pun37 nil code-not-known 0.037 (2003 1 10 1 34 42))
 ((main)
  ((a unk) (b unk) (c unk) (d unk) (e unk) (f unk) (g unk))
  ((oa ~10) (ob ~11))
  ((~1+01  (b d)   )
   (~2+01  (b e)   )
   (~3+01  (a (f))   )
   (~4+01  ((b) (e))   )
   (<12>+01  ((b c) ((a) (c)))   )
   (~5+02  (e ~3)   )
   (~6+02  (~1 (e))   )
   (~7+02  (~4 (g))   )
   (~9+02  (<12> (d))   )
   (~8+03  (~5 (b))   )
   (<13>+03  ((a ~7) (~6 (a)))   )
   (~10+04  ((~8 ~9))   )
   (~11+04  ((~2 (<13>)))   )  )))
```

```
((lion comesh-place-and-route pun37 nil code-not-known 0.071 (2003 1 10 1 35 0))
 ((main)
  ((a unk) (b unk) (@clok unk))
  ((oa ~12))
  ((~4+01  (b !2)   )
   (~5+01  (a (!1))   )
   (~6+01  (b (!1))   )
   (~10+01  (a b !1)   )
   (~11+01  (b (a) (!1))   )
   (~3+02  (a ~4)   )
   (~7+02  (~10 (!2))   )
   (~12+02  ((~5 ~6 !2))   )
   (~8+03  ((~3 ~5))   )
   (~9+03  ((~7 ~11))   )
   (!1+06  (@clok (~8))   )
   (!2+06  (@clok (~9))   )  )))
```

```
((cm85a comesh-place-and-route pun37 nil code-not-known 0.167 (2003 1 10 1 34 43))
 ((main)
  ((a unk)(b unk)(c unk)(d unk)(e unk)(f unk)(g unk)(h unk)(i unk)(j unk) (k unk))
  ((oa ~17) (ob ~15) (oc ~18))
  ((~1+01  (a (b))   )
   (~2+01  (b (a))   )
   (~3+01  (e (f))   )
   (~4+01  (f (e))   )
   (~11+01  (g (h) (j))   )
   (~12+01  (h (g) (j))   )
   (=19=+01  (((c d) ((c) (d))))   )
   (=20=+01  (((g h) ((g) (h))))   )
   (~5+02  ((j) (=20=))   )
   (~8+03  ((~3) (~5))   )
   (~9+03  ((~4) (~5))   )
   (~16+03  (~3 ~4 (~5))   )
   (~10+04  ((~16) (=19=))   )
   (~13+04  (c (d) (~16))   )
   (~14+04  (d (c) (~16))   )
   (~6+05  ((~1) (~10))   )
   (~7+05  ((~2) (~10))   )
   (~15+05  (~1 ~2 (~10))   )
   (~17+07  ((i ~6 ~8 ~11 ~13))   )
   (~18+07  ((k ~7 ~9 ~12 ~14))   )  )))


((s208 comesh-place-and-route pun37 nil code-not-known 0.313 (2003 1 10 1 35 2))
 ((main)
  ((a unk)(b unk)(c unk)(d unk)(e unk)(f unk)(g unk)(h unk)(i unk)
   (j unk)(k unk)(@clok unk))
  ((oa ~34) (ob ~8))
  ((~7+01  (!4 (j))   )
   (~9+01  ((f) (!1))   )
   (~10+01  ((h) (!3))   )
   (~11+01  ((i) (!5))   )
   (~12+01  (!4 !5)   )
   (~18+01  (!4 (!1))   )
   (~22+01  ((!4) (!5))   )
   (~26+01  (!3 (g) (!2))   )
   (~27+01  ((b (a) (!4)))   )
   (~29+01  (!3 (!4) (!5))   )
   (~30+01  ((!3) (!4) (!5))   )
   (~32+01  (!2 (!3) (!4) (!5))   )
   (~33+01  ((a) (!2) (!3) (!4) (!5))   )
   (~34+01  (!1 !2 !3 !4 !5)   )
   (~13+02  (~10 ~26)   )
   (~15+02  (~18 ~33)   )
   (~19+02  (~22 (!3))   )
   (~21+02  (~30 (!2))   )
   (~31+02  (b ~12 ~22 (a))   )
   (~6+03  (b ~15)   )
   (~14+03  (~13 !5)   )
   (~16+03  (~19 ~29)   )
   (~17+03  (~21 ~32)   )
   (~24+04  (b ~16 (a))   )
   (~25+04  (b ~17 (a))   )
   (~28+04  (~9 ~11 ~14)   )
   (~20+05  (~28 (!4))   )
   (!4+06  (@clok (~27))   )
   (!5+06  (@clok (~31))   )
   (!1+06  (@clok (~6))   )
   (!2+06  (@clok (~25))   )
   (!3+06  (@clok (~24))   )
   (~23+07  (k ~7 ~20)   )
   (~8+08  (~23 (a))   )  )))
```

```
((count comesh-place-and-route pun37 nil code-not-known 0.476 (2003 1 10 1 34 46))
 ((main)  ((aa unk)(ab unk)(ac unk)(ad unk)(ae unk)(af unk)(ag unk)(ah unk)(ai unk)
           (aj unk)(ak unk)(al unk)(am unk)(an unk)(ao unk)(ap unk)(aq unk)(ar unk)
           (as unk)(at unk)(au unk)(av unk)(aw unk)(ax unk)(ay unk)(az unk)(ba unk)
           (bb unk)(bc unk)(bd unk)(be unk)(bf unk)(bg unk)(bh unk)(bi unk))
  ((oa ~31) (ob ~30) (oc ~29) (od ~28) (oe ~27) (of ~26) (og ~25) (oh ~24)
   (oi ~23) (oj ~22) (ok ~21) (ol ~20) (om ~19) (on ~18) (oo ~17) (op ~16))
  ((~1+01  (af ah)   )
   (=32=+01  (((af ah) ((af) (ah))))   )
   (~6+02  (ae (~1))   )
   (=37=+02  (((ae ~1) ((ae) (~1))))   )
   (<48>+02  ((ai aj) ((ai) (=32=)))   )
   (~5+03  (ad (~6))   )
   (~16+03  ((ag (<48>)))   )
   (=36=+03  (((ad ~6) ((ad) (~6))))   )
   (<49>+03  ((ai ak) (=37= (ai)))   )
   (~4+04  (ac (~5))   )
   (~17+04  ((ag (<49>)))   )
   (=35=+04  (((ac ~5) ((ac) (~5))))   )
   (<50>+04  ((ai al) (=36= (ai)))   )
   (~3+05  (ab (~4))   )
   (~18+05  ((ag (<50>)))   )
   (=34=+05  (((ab ~4) ((ab) (~4))))   )
   (<51>+05  ((ai am) (=35= (ai)))   )
   (~2+07  (aa (~3))   )
   (~19+07  ((ag (<51>)))   )
   (=33=+07  (((aa ~3) ((aa) (~3))))   )
   (<52>+07  ((ai an) (=34= (ai)))   )
   (~15+08  (bh (~2))   )
   (~20+08  ((ag (<52>)))   )
   (=47=+08  (((bh ~2) ((bh) (~2))))   )
   (<53>+08  ((ai ao) (=33= (ai)))   )
   (~14+09  (bf (~15))   )
   (~21+09  ((ag (<53>)))   )
   (=46=+09  (((bf ~15) ((bf) (~15))))   )
   (<54>+09  ((ai aq) (=47= (ai)))   )
   (~13+10  (bd (~14))   )
   (~22+10  ((ag (<54>)))   )
   (=45=+10  (((bd ~14) ((bd) (~14))))   )
   (<55>+10  ((ai as) (=46= (ai)))   )
   (~12+11  (bb (~13))   )
   (~23+11  ((ag (<55>)))   )
   (=44=+11  (((bb ~13) ((bb) (~13))))   )
   (<56>+11  ((ai au) (=45= (ai)))   )
   (~11+13  (az (~12))   )
   (~24+13  ((ag (<56>)))   )
   (=43=+13  (((az ~12) ((az) (~12))))   )
   (<57>+13  ((ai aw) (=44= (ai)))   )
   (~10+14  (ax (~11))   )
   (~25+14  ((ag (<57>)))   )
   (=42=+14  (((ax ~11) ((ax) (~11))))   )
   (<58>+14  ((ai ay) (=43= (ai)))   )
   (~9+15  (av (~10))   )
   (~26+15  ((ag (<58>)))   )
   (=41=+15  (((av ~10) ((av) (~10))))   )
   (<59>+15  ((ai ba) (=42= (ai)))   )
   (~8+16  (at (~9))   )
   (~27+16  ((ag (<59>)))   )
   (=40=+16  (((at ~9) ((at) (~9))))   )
   (<60>+16  ((ai bc) (=41= (ai)))   )
   (~7+17  (ar (~8))   )
   (~28+17  ((ag (<60>)))   )
   (=39=+17  (((ar ~8) ((ar) (~8))))   )
   (<61>+17  ((ai be) (=40= (ai)))   )
   (~29+19  ((ag (<61>)))   )
```

```
   (=38=+19  (((ap ~7) ((ap) (~7))))   )
   (<62>+19  ((ai bg) (=39= (ai)))   )
   (~30+20  ((ag (<62>)))   )
   (<63>+20  ((ai bi) (=38= (ai)))   )
   (~31+21  ((ag (<63>)))   )  )))

((mult16a comesh-place-and-route pun37 nil code-not-known 6.426 (2003 1 10 1 35 39))
 ((main) ((a unk) (b unk) (c unk) (d unk) (e unk) (f unk) (g unk) (h unk) (i unk)
          (j unk) (k unk) (l unk) (m unk) (n unk) (o unk) (p unk) (q unk) (@clok unk))
  ((oa ~38))
  ((~17+01  ((a) (q))   )
   (~18+01  ((b) (q))   )
   (~19+01  ((c) (q))   )
   (~20+01  ((d) (q))   )
   (~21+01  ((e) (q))   )
   (~22+01  ((f) (q))   )
   (~23+01  ((g) (q))   )
   (~24+01  ((h) (q))   )
   (~25+01  ((i) (q))   )
   (~26+01  ((j) (q))   )
   (~27+01  ((k) (q))   )
   (~28+01  ((l) (q))   )
   (~29+01  ((m) (q))   )
   (~30+01  ((n) (q))   )
   (~31+01  ((o) (q))   )
   (~32+01  ((p) (q))   )
   (~33+02  (~17 !3)   )
   (~35+02  (~32 !5)   )
   (~152+02  ((~17) (!3))   )
   (~154+02  ((~32) (!5))   )
   (~34+03  (~31 ~154)   )
   (~38+03  (~35 ~154)   )
   (~69+03  (~154 !6)   )
   (~153+03  ((~31) (~154))   )
   (~156+03  ((~154) (!6))   )
   (~37+04  (~34 ~153)   )
   (~100+04  (~69 (~31))   )
   (~172+04  (~34 ~153 !6)   )
   (~68+05  (~100 ~156)   )
   (~138+05  (~37 (!6))   )
   (~71+07  (!7 (~68))   )
   (~96+07  (~30 (~68))   )
   (~101+07  (~68 (!7))   )
   (~102+07  (~68 (~30))   )
   (~171+07  ((~138 ~172))   )
   (~54+08  (~96 ~102)   )
   (~140+08  (~71 (~30))   )
   (~185+08  (~96 ~102 !7)   )
   (~55+09  (~101 ~140)   )
   (~137+09  (~54 (!7))   )
   (~72+10  (!8 (~55))   )
   (~95+10  (~29 (~55))   )
   (~103+10  (~55 (!8))   )
   (~104+10  (~55 (~29))   )
   (~170+10  ((~137 ~185))   )
   (~53+11  (~95 ~104)   )
   (~141+11  (~72 (~29))   )
   (~184+11  (~95 ~104 !8)   )
   (!5+12  (@clok (~171))   )
   (!6+12  (@clok (~170))   )
   (~56+13  (~103 ~141)   )
   (~136+13  (~53 (!8))   )
   (~73+14  (!9 (~56))   )
   (~94+14  (~28 (~56))   )
   (~105+14  (~56 (!9))   )
```

```
(~106+14  (~56 (~28))   )
(~169+14  ((~136 ~184))   )
(~52+15  (~94 ~106)   )
(~142+15  (~73 (~28))   )
(~183+15  (~94 ~106 !9)   )
(~57+16  (~105 ~142)   )
(~135+16  (~52 (!9))   )
(~74+17  (!10 (~57))   )
(~93+17  (~27 (~57))   )
(~107+17  (~57 (!10))   )
(~108+17  (~57 (~27))   )
(~168+17  ((~135 ~183))   )
(!7+18  (@clok (~169))   )
(!8+18  (@clok (~168))   )
(~51+19  (~93 ~108)   )
(~143+19  (~74 (~27))   )
(~182+19  (~93 ~108 !10)   )
(~58+20  (~107 ~143)   )
(~134+20  (~51 (!10))   )
(~75+21  (!11 (~58))   )
(~92+21  (~26 (~58))   )
(~109+21  (~58 (!11))   )
(~110+21  (~58 (~26))   )
(~167+21  ((~134 ~182))   )
(~50+22  (~92 ~110)   )
(~144+22  (~75 (~26))   )
(~181+22  (~92 ~110 !11)   )
(~59+23  (~109 ~144)   )
(~133+23  (~50 (!11))   )
(!9+24  (@clok (~167))   )
(~76+25  (!12 (~59))   )
(~91+25  (~25 (~59))   )
(~111+25  (~59 (!12))   )
(~112+25  (~59 (~25))   )
(~166+25  ((~133 ~181))   )
(~49+26  (~91 ~112)   )
(~145+26  (~76 (~25))   )
(~180+26  (~91 ~112 !12)   )
(~60+27  (~111 ~145)   )
(~132+27  (~49 (!12))   )
(~77+28  (!13 (~60))   )
(~90+28  (~24 (~60))   )
(~113+28  (~60 (!13))   )
(~114+28  (~60 (~24))   )
(~165+28  ((~132 ~180))   )
(~48+29  (~90 ~114)   )
(~146+29  (~77 (~24))   )
(~179+29  (~90 ~114 !13)   )
(!10+30  (@clok (~166))   )
(!11+30  (@clok (~165))   )
(~61+31  (~113 ~146)   )
(~131+31  (~48 (!13))   )
(~78+32  (!14 (~61))   )
(~89+32  (~23 (~61))   )
(~115+32  (~61 (!14))   )
(~116+32  (~61 (~23))   )
(~164+32  ((~131 ~179))   )
(~47+33  (~89 ~116)   )
(~147+33  (~78 (~23))   )
(~178+33  (~89 ~116 !14)   )
(~62+34  (~115 ~147)   )
(~130+34  (~47 (!14))   )
(~79+35  (!15 (~62))   )
(~88+35  (~22 (~62))   )
(~117+35  (~62 (!15))   )
```

```
(~118+35  (~62 (~22))   )
(~163+35  ((~130 ~178))   )
(!12+36  (@clok (~164))   )
(!13+36  (@clok (~163))   )
(~46+37  (~88 ~118)   )
(~148+37  (~79 (~22))   )
(~177+37  (~88 ~118 !15)   )
(~63+38  (~117 ~148)   )
(~129+38  (~46 (!15))   )
(~80+39  (!16 (~63))   )
(~87+39  (~21 (~63))   )
(~119+39  (~63 (!16))   )
(~120+39  (~63 (~21))   )
(~162+39  ((~129 ~177))   )
(~45+40  (~87 ~120)   )
(~149+40  (~80 (~21))   )
(~176+40  (~87 ~120 !16)   )
(~64+41  (~119 ~149)   )
(~128+41  (~45 (!16))   )
(!14+42  (@clok (~162))   )
(~81+43  (!4 (~64))   )
(~86+43  (~20 (~64))   )
(~121+43  (~64 (!4))   )
(~122+43  (~64 (~20))   )
(~161+43  ((~128 ~176))   )
(~44+44  (~86 ~122)   )
(~150+44  (~81 (~20))   )
(~175+44  (~86 ~122 !4)   )
(~65+45  (~121 ~150)   )
(~127+45  (~44 (!4))   )
(~70+46  (!2 (~65))   )
(~85+46  (~19 (~65))   )
(~123+46  (~65 (!2))   )
(~124+46  (~65 (~19))   )
(~160+46  ((~127 ~175))   )
(~43+47  (~85 ~124)   )
(~139+47  (~70 (~19))   )
(~174+47  (~85 ~124 !2)   )
(!15+48  (@clok (~161))   )
(!16+48  (@clok (~160))   )
(~66+49  (~123 ~139)   )
(~126+49  (~43 (!2))   )
(~82+50  (!1 (~66))   )
(~84+50  (~18 (~66))   )
(~98+50  (~66 (!1))   )
(~99+50  (~66 (~18))   )
(~159+50  ((~126 ~174))   )
(~42+51  (~84 ~99)   )
(~151+51  (~82 (~18))   )
(~173+51  (~84 ~99 !1)   )
(~67+52  (~98 ~151)   )
(~125+52  (~42 (!1))   )
(~36+53  (~33 ~67)   )
(~83+53  (~17 (~67))   )
(~97+53  (~67 (~17))   )
(~158+53  ((~125 ~173))   )
(!4+54  (@clok (~159))   )
(!2+54  (@clok (~158))   )
(~41+55  (~83 ~97)   )
(~157+55  ((~36 ~152))   )
(~39+56  (~41 !3)   )
(~155+56  ((~41) (!3))   )
(~40+57  (~39 ~155)   )
(!3+60  (@clok (~157))   )
(!1+60  (@clok (~40))   )  )))
```