# Revised tables for

## *ITEM: If-Then-Else Minimizer for Logic Synthesis*

Kevin Karplus

22 May 1992

Computer Engineering Department
University of California, Santa Cruz
Santa Cruz, CA 95064
Internet: karplus@ce.ucsc.edu

This document contains revised tables for Kevin Karplus's EuroAsic92 talk: *ITEM: If-Then-Else Minimizer for Logic Synthesis*.

Each table gives results for one mapper mapping combinational logic to Xilinx 3000 cells. There are three mappers, and results are given both for mapping unoptimized benchmark files and for optimizing with ITEM first, giving six tables in total.

The numbers reported are the number of 5-input tables, the number of Xilinx cells (CLBs), the number of levels of logic (unit-delay estimate), a weighted-fanout delay estimate, and the number of gate inputs in the network.

The weighted-fanout delay model is from an unpublished paper by Martine Schlag, Pak Chan, and Jackson Kong, and gives an estimate in nanoseconds. It is expected to provide a better estimate of after-routing delay than the unit-delay model, but this has not been verified.

The number of gate inputs is a crude approximation to the number of "pins" in the circuit. It is faulty in two ways: it doesn't count the primary outputs as pins, and it counts inputs shared by two different functions in the same CLB twice. Some minor improvements are needed to ITEM before pins can be counted accurately—these changes should be in the next release (late summer 1992).

The unit delay turns out to be a poor predictor for the fanout-weighted delay. In fact, for this set of 159 examples, any of the area measures (number of lookup tables, number of CLBs, or number of pins) provided a better estimate of the longest weighted-fanout delay. The best predictor was

$$\text{fanout-weighted delay} = 11.552\text{CLBs}^{0.63663}$$

with $R^2 = 0.857$. In contrast, the curve fit for unit delay was

$$\text{fanout-weighted delay} = 38.947\text{unit-delay}^{0.82870}$$

with $R^2 = 0.702$.

## References

[CKTC92] Jason Cong, Andrew Kahng, Peter Trajmar, and Kuang-Chien Chen. Graph based FPGA technology mapping for delay optimization. In *FPGA'92: First International ACM/SIGDA Workshop on Field-Programmable Gate Arrays*, pages 77–82, Berkeley, CA, 16–18 February 1992.

[Kar91] Kevin Karplus. Xmap: a technology mapper for table-lookup field-programmable gate arrays. In *ACM IEEE 28$^{th}$ Design Automation Conference Proceedings*, pages 240–243, San Francisco, CA, 17–21 June 1991.

[SK91] Søren Søe and Kevin Karplus. Logic minimization using two-column rectangle replacement. In *ACM IEEE 28$^{th}$ Design Automation Conference Proceedings*, pages 470–473, San Francisco, CA, 17–21 June 1991.

```
INITIAL
print "Script started " date()
print "The parameters for xtmap are learned from xmap,\n"
print "followed by an exponential sweep of the penalty for delay,\n"
print " to ensure that minimum LutHeight is achieved.\n"
print "The sweep is followed by 5 random changes to the bonuses,\n"
print "to see if there are any easy improvements to make.\n\n"

print "Comparing Xmap, Xcmap, and Xtmap: "
print "(lookup tables, CLBs, unitdelay, pakdelay, pins)\n"
print Filename column(20) Initial column(30) Xcmap
print column(60) Xmap column(90) Xtmap
print column(120) cputime
print "\n"
set lutwidth 5
ALWAYS
set cputime 0
read -blif
print filename()
print column(20)  "(" gatecount() ",," unitdelay() ")"

map xcmap
print column(30)  "(" gatecount() , cellcount() , unitdelay(),
print pakdelay(), number(i(.*)) ")"

map xmap -h 3
print column(60)  "(" gatecount() , cellcount() , unitdelay(),
print pakdelay(), number(i(.*)) ")"

# learn the bonuses from the xmap mapping
set bonuses 0 0 0 0 0 0 0 0 0 -20
map xtmap -learn

# increase delay weight until unitdelay<=lutheight, then do 5 random searches
map xtmap -search -r 5 -a gatecount -d unitdelay
# note: the "optimized" benchmarks were run with cellcount instead of gatecount
# in the first Xtmap mapping---this shouldn't have changed the results by
# much.

# Make sure you choose a minimal delay solution (even if area*delay worse)
map xtmap -search -d unitdelay
print column(90)  "(" gatecount() , cellcount() , unitdelay(),
print pakdelay(), number(i(.*)) ")"
print column(120) cputime()
print "\n"
FINAL
quit
```

Figure 1: Script for producing results on unoptimized benchmarks. The Xtmap results could be improved by using `map xtmap -search -r 5 -d pakdelay`, optimizing for the weighted-fanout delay model rather than number of lookup tables times level of logic.

| | Xcmap | | | | |
|---|---|---|---|---|---|
| benchmark | tables | CLBs | levels | est. delay | pins |
| 5xp1 | 66 | 50 | 5 | 164.8 | 284 |
| 9sym | 193 | 151 | 23 | 357.4 | 721 |
| 9symml | 62 | 51 | 5 | 131.8 | 262 |
| C499 | 100 | 80 | 5 | 100.9 | 397 |
| C880 | 156 | 115 | 8 | 143.55 | 636 |
| alu2 | 196 | 143 | 11 | 330.15 | 725 |
| alu4 | 329 | 248 | 11 | 386.15 | 1261 |
| apex6 | 283 | 228 | 5 | 304.7 | 1257 |
| apex7 | 106 | 76 | 4 | 166.45 | 426 |
| b9 | 51 | 38 | 3 | 88.5 | 218 |
| apex2 | 2830 | 2468 | 134 | 2434.6 | 12517 |
| apex4 | 995 | 903 | 43 | 1580.9 | 4726 |
| des | 1474 | 1134 | 10 | 838.5 | 6188 |
| bw | 28 | 27 | 1 | 103.0 | 143 |
| clip | 263 | 220 | 13 | 502.5 | 1116 |
| count | 31 | 29 | 5 | 88.2 | 166 |
| duke2 | 238 | 200 | 8 | 352.5 | 1043 |
| e64 | 403 | 382 | 16 | 233.8 | 1980 |
| f51m | 29 | 23 | 4 | 92.7 | 119 |
| misex1 | 23 | 16 | 3 | 85.7 | 102 |
| misex2 | 48 | 41 | 3 | 111.4 | 237 |
| rd73 | 235 | 168 | 18 | 500.2 | 903 |
| rd84 | 388 | 262 | 42 | 1072.8 | 1690 |
| rot | 316 | 225 | 10 | 196.8 | 1255 |
| sao2 | 104 | 94 | 8 | 190.5 | 466 |
| vg2 | 222 | 190 | 11 | 316.0 | 978 |
| z4ml | 12 | 11 | 3 | 61.4 | 57 |

Table 1: Results of running Xcmap for mapping to Xilinx 3000 series cells on unoptimized combinational logic benchmark files. Xcmap is a version of the central algorithm of Cong, Kahng, Trajmar, and Chen [CKTC92].

| | Xmap | | | | |
|---|---|---|---|---|---|
| benchmark | tables | CLBs | levels | est. delay | pins |
| 5xp1 | 60 | 52 | 9 | 190.6 | 274 |
| 9sym | 127 | 125 | 37 | 458.2 | 628 |
| 9symml | 56 | 50 | 7 | 143.2 | 252 |
| C499 | 72 | 53 | 6 | 93.55 | 309 |
| C880 | 102 | 90 | 12 | 150.7 | 481 |
| alu2 | 129 | 101 | 21 | 333.15 | 515 |
| alu4 | 251 | 203 | 19 | 307.8 | 1004 |
| apex6 | 223 | 189 | 7 | 267.45 | 1048 |
| apex7 | 77 | 61 | 5 | 154.8 | 351 |
| b9 | 41 | 34 | 4 | 94.6 | 207 |
| apex2 | 2410 | 2274 | 271 | 3466.6 | 11567 |
| apex4 | 1011 | 915 | 46 | 1669.8 | 4735 |
| des | 1262 | 1027 | 13 | 838.5 | 5447 |
| bw | 28 | 27 | 2 | 110.2 | 142 |
| clip | 207 | 193 | 19 | 491.9 | 990 |
| count | 31 | 29 | 5 | 88.2 | 166 |
| duke2 | 218 | 186 | 12 | 361.55 | 985 |
| e64 | 400 | 380 | 16 | 235.2 | 1966 |
| f51m | 26 | 21 | 5 | 93.9 | 110 |
| misex1 | 22 | 17 | 3 | 85.7 | 97 |
| misex2 | 44 | 39 | 3 | 102.4 | 226 |
| rd73 | 155 | 148 | 34 | 561.4 | 755 |
| rd84 | 352 | 334 | 113 | 1733.2 | 1732 |
| rot | 237 | 189 | 19 | 234.2 | 1039 |
| sao2 | 88 | 87 | 9 | 201.7 | 440 |
| vg2 | 187 | 179 | 21 | 352.8 | 915 |
| z4ml | 11 | 10 | 3 | 61.4 | 56 |

Table 2: Results of running Xmap mapping to Xilinx 3000 series cells on unoptimized combinational logic benchmark files. Xmap is a fast greedy algorithm inteded for area, rather than delay optimization [Kar91].

| | Xtmap | | | | | |
|---|---|---|---|---|---|---|
| benchmark | tables | CLBs | levels | est. delay | pins | cpu seconds |
| 5xp1 | 62 | 48 | 5 | 159.9 | 275 | 9.8 |
| 9sym | 133 | 120 | 23 | 348.4 | 600 | 29.6 |
| 9symml | 58 | 47 | 5 | 123.9 | 248 | 7.1 |
| C499 | 72 | 66 | 5 | 101.5 | 339 | 33.1 |
| C880 | 140 | 108 | 8 | 133.2 | 611 | 64.4 |
| alu2 | 146 | 122 | 11 | 307.85 | 631 | 70.2 |
| alu4 | 267 | 220 | 11 | 383.1 | 1114 | 168.9 |
| apex6 | 229 | 190 | 5 | 274.25 | 1080 | 39.1 |
| apex7 | 80 | 66 | 4 | 167.2 | 376 | 18.9 |
| b9 | 44 | 34 | 3 | 91.5 | 215 | 7.0 |
| apex2 | 2758 | 2685 | 134 | 2647.6 | 13550 | 1099.4 |
| apex4 | 994 | 916 | 43 | 1622.9 | 4767 | 436.6 |
| des | 1310 | 1204 | 10 | 851.5 | 6332 | 1002.7 |
| bw | 28 | 27 | 1 | 103.0 | 143 | 7.6 |
| clip | 229 | 201 | 13 | 454.7 | 1050 | 38.6 |
| count | 31 | 29 | 5 | 88.2 | 166 | 2.3 |
| duke2 | 223 | 187 | 8 | 333.5 | 993 | 88.0 |
| e64 | 402 | 380 | 16 | 235.2 | 1975 | 40.2 |
| f51m | 26 | 21 | 4 | 86.7 | 107 | 4.3 |
| misex1 | 21 | 15 | 3 | 85.5 | 89 | 5.1 |
| misex2 | 42 | 35 | 3 | 90.4 | 211 | 5.9 |
| rd73 | 187 | 148 | 18 | 484.7 | 831 | 51.3 |
| rd84 | 290 | 275 | 46 | 806.1 | 1415 | 171.2 |
| rot | 254 | 185 | 10 | 202.0 | 1058 | 54.9 |
| sao2 | 98 | 91 | 8 | 196.5 | 471 | 17.8 |
| vg2 | 215 | 181 | 11 | 299.1 | 956 | 37.9 |
| z4ml | 13 | 11 | 3 | 63.4 | 65 | 2.9 |

Table 3: Results of running Xtmap mapping to Xilinx 3000 series cells on unoptimized combinational logic benchmark files. Xtmap is a new generate-and-test mapper. The cpu time includes running Xcmap and Xmap as well as Xtmap, but most of it is attributable to Xtmap.

```
order -c lutheight
transform -m local -d lutheight -a edges
bcov -m tox -f lutheight bcdelay -s bcvalue bcarea
```

Figure 2: Optimization script used for the optimized benchmarks. First a variable ordering is chosen, then LocalFactor is done, choosing the smallest known equivalent expression using an area-delay product cost function. The area is estimated as the number of non-trivial edges in the if-then-else DAG, and the delay is estimated as the unit-delay resulting from Xcmap. After the transformations, block covering (two-column rectangle replacement [SK91]) is applied to the OR- and XOR-matrices. Column pairs are chosen based on minimum estimated delay, using the number of rows in the rectangle as a tie-breaker.

| | Xcmap optimized | | | | |
|---|---|---|---|---|---|
| benchmark | tables | CLBs | levels | est. delay | pins |
| 5xp1 | 21 | 16 | 3 | 73.2 | 83 |
| 9sym | 8 | 8 | 3 | 61.7 | 49 |
| 9symml | 8 | 8 | 3 | 61.7 | 49 |
| C499 | 94 | 64 | 4 | 82.2 | 373 |
| C880 | 170 | 121 | 7 | 147.8 | 672 |
| alu2 | 90 | 70 | 5 | 177.6 | 366 |
| alu4 | 260 | 195 | 8 | 328.7 | 1011 |
| apex2 | | | | | |
| apex4 | 533 | 414 | 5 | 831.9 | 2170 |
| apex6 | 254 | 214 | 4 | 285.5 | 1187 |
| apex7 | 115 | 79 | 4 | 175.7 | 447 |
| b9 | 42 | 31 | 3 | 79.5 | 198 |
| bw | 28 | 27 | 1 | 103.0 | 143 |
| clip | 59 | 48 | 4 | 125.7 | 252 |
| count | 39 | 33 | 3 | 74.2 | 186 |
| des | 1336 | 841 | 7 | 951.2 | 4765 |
| duke2 | 202 | 151 | 4 | 212.4 | 785 |
| e64 | 157 | 98 | 3 | 137.25 | 578 |
| rot | 316 | 222 | 7 | 178.95 | 1233 |
| f51m | 17 | 14 | 3 | 66.4 | 73 |
| misex1 | 19 | 14 | 2 | 66.3 | 81 |
| misex2 | 51 | 37 | 3 | 94.2 | 206 |
| rd73 | 10 | 9 | 2 | 52.95 | 50 |
| rd84 | 16 | 13 | 3 | 62.5 | 76 |
| sao2 | 52 | 42 | 4 | 110.7 | 212 |
| vg2 | 121 | 85 | 5 | 138.9 | 460 |
| z4ml | 10 | 8 | 2 | 51.7 | 41 |

Table 4: Results of running Xcmap for mapping to Xilinx 3000 series cells on combinational logic benchmark files optimized by ITEM. Xcmap is a version of the central algorithm of Cong, Kahng, Trajmar, and Chen [CKTC92].

| | Xmap optimized | | | | |
|---|---|---|---|---|---|
| benchmark | tables | CLBs | levels | est. de.lay | pins |
| 5xp1 | 19 | 15 | 3 | 67.2 | 79 |
| 9sym | 8 | 8 | 3 | 61.7 | 49 |
| 9symml | 8 | 8 | 3 | 61.7 | 49 |
| C499 | 75 | 69 | 6 | 104.3 | 352 |
| C880 | 118 | 95 | 12 | 153.0 | 531 |
| alu2 | 97 | 74 | 9 | 189.95 | 397 |
| alu4 | 216 | 180 | 13 | 289.8 | 910 |
| apex2 | | | | | |
| apex4 | 689 | 526 | 9 | 1259.95 | 2784 |
| apex6 | 217 | 185 | 8 | 264.95 | 1011 |
| apex7 | 83 | 62 | 7 | 165.45 | 360 |
| b9 | 33 | 26 | 3 | 79.2 | 171 |
| bw | 28 | 27 | 2 | 110.2 | 142 |
| clip | 51 | 44 | 6 | 125.8 | 228 |
| count | 33 | 31 | 4 | 79.95 | 174 |
| des | 1022 | 782 | 12 | 950.7 | 4218 |
| duke2 | 162 | 130 | 8 | 188.6 | 664 |
| e64 | 104 | 78 | 6 | 124.25 | 457 |
| rot | 235 | 183 | 12 | 184.8 | 1011 |
| f51m | 16 | 14 | 3 | 66.4 | 72 |
| misex1 | 17 | 14 | 3 | 72.4 | 78 |
| misex2 | 40 | 30 | 4 | 86.4 | 178 |
| rd73 | 9 | 9 | 2 | 52.95 | 49 |
| rd84 | 14 | 13 | 3 | 62.5 | 74 |
| sao2 | 47 | 39 | 7 | 123.5 | 201 |
| vg2 | 84 | 71 | 7 | 124.1 | 374 |
| z4ml | 7 | 7 | 2 | 51.7 | 36 |

Table 5: Results of running Xmap mapping to Xilinx 3000 series cells on combinational logic benchmark files optimized by ITEM. Xmap is a fast greedy algorithm inteded for area, rather than delay optimization [Kar91].

| | Xtmap | | | | | |
|---|---|---|---|---|---|---|
| benchmark | tables | CLBs | levels | est. de.lay | pins | cpu seconds |
| 5xp1 | 20 | 15 | 3 | 73.5 | 88 | 5.7 |
| 9sym | 8 | 8 | 3 | 61.7 | 49 | 39.0 |
| 9symml | 8 | 8 | 3 | 61.7 | 49 | 8.8 |
| C499 | 86 | 75 | 4 | 95.2 | 394 | 539.8 |
| C880 | 138 | 113 | 7 | 134.55 | 619 | 93.3 |
| alu2 | 84 | 68 | 5 | 168.35 | 353 | 41.0 |
| alu4 | 222 | 182 | 8 | 303.05 | 934 | 193.4 |
| apex2 | | | | | | |
| apex4 | 524 | 406 | 5 | 821.1 | 2142 | 892.4 |
| apex6 | 216 | 195 | 4 | 277.0 | 1087 | 60.9 |
| apex7 | 82 | 64 | 4 | 161.95 | 372 | 27.3 |
| b9 | 38 | 28 | 3 | 79.7 | 188 | 5.4 |
| bw | 28 | 27 | 1 | 103.0 | 143 | 10.1 |
| clip | 47 | 39 | 4 | 113.7 | 212 | 30.9 |
| count | 33 | 28 | 3 | 67.45 | 178 | 4.9 |
| des | 1115 | 771 | 7 | 951.2 | 4387 | 1111.6 |
| duke2 | 176 | 145 | 4 | 190.2 | 755 | 100.1 |
| e64 | 157 | 98 | 3 | 137.25 | 578 | 132.4 |
| rot | 263 | 201 | 7 | 163.2 | 1125 | 175.1 |
| f51m | 17 | 13 | 3 | 66.4 | 69 | 2.7 |
| misex1 | 19 | 14 | 2 | 69.3 | 83 | 4.4 |
| misex2 | 39 | 33 | 3 | 86.3 | 185 | 6.7 |
| rd73 | 9 | 8 | 2 | 52.95 | 48 | 19.6 |
| rd84 | 14 | 12 | 3 | 64.7 | 73 | 70.1 |
| sao2 | 43 | 38 | 4 | 113.7 | 206 | 16.9 |
| vg2 | 91 | 72 | 5 | 132.9 | 388 | 56.1 |
| z4ml | 6 | 5 | 2 | 51.8 | 32 | 2.4 |

Table 6: Results of running Xtmap mapping to Xilinx 3000 series cells on combinational logic benchmark files optimized by ITEM. Xtmap is a new generate-and-test mapper. The cpu time includes optimization time, Xcmap time, and Xmap time as well as Xtmap, but most of it is attributable to the optimization and to Xtmap.