Table: Optimizing OpenCores designs by restructuring control logic.

Design name	Design statistics				Baseline			Transformed		
	PI	PO	Reg	FF	Base	BaseOpt	BaseLOpt	Trans	TransOpt	TransLOpt
double_fpu	136	70	132	5222	73425	62006	114	72808	60309	114
mem_ctrl	115	152	59	1138	21064	8798	38	11117	7607	38
nova	85	89	565	6429	559282	219827	118	148228	121603	108
picorv32	102	307	23	701	11599	6819	71	7022	6149	72
reedsolomn	11	10	245	3059	72915	25776	32	31408	22582	31
sudoku	732	731	171	3160	127355	63122	90	97836	59030	64
uart16550	51	37	32	352	4776	2560	18	2942	2374	18
usb	128	121	113	1762	16551	13240	41	14793	12472	41
vga_lcd	89	109	56	830	11682	5321	32	5513	5055	33
wb_dma	217	215	41	796	73642	41074	20	57802	42401	20
Geomean					1.000	1.000	1.000	0.596	0.885	0.959

The following notation is used in the table

PI (**PO**) is the number of bit-level primary inputs (outputs).
Reg is the number of word-level registers in the design.
FF is the number of bit-level flops. (Some of them are part of word-level registers.)
Baseline is the result without MUX restructuring. **Transformed** is the result with MUX restructuring.
Base (**Trans**) are AIG sizes after structural hashing.
BaseOpt (**TransOpt**) is the AIG sizes after applying &dc2 in ABC.
BaseLOpt (**TransLOpt**) is the AIG level count after applying &dc2 in ABC.

Benchmarks

We considered 10 out of 30 designs from the OpenCores repository, for which applying MUX restructuring was most helpful. For the remaining 20 designs, there was no improvement or a very minor improvement (about 1% in area).

Experiments

The optimization was applied to the whole design after flattening the user hierarchy. The results were verified by extracting a combinational AIG before and after the transform and comparing these AIGs using command &cec in ABC.

Conclusions

(1) For the selected designs, unoptimized AIG size is reduced by about 40%.
(2) After AIG rewriting, the average reduction in area is about 11.5%.
(3) After AIG rewriting, the average reduction in AIG level count is about 4%.