



Advanced VLSI Design

EE 7325

Comparison of a 32-bit Behavioral Multiplier and Booth-2 Algorithm Multiplier

Mark Ripley Sears (mrs171030)

Navya Sri Sreeram (nxs161131)

Pin Pitch	0.26um
Design #1: Behavioral Multiplier	7,658 Cells
Design #2: Booth-2 Algorithm	23,598 Cells

08/06/2018

32-bit Multiplier Design

In this project, two multiplier designs were synthesized and the tradeoffs are compared.

1) Design #1: Behavioral Multiplier (X*Y)

A default synthesis tool inferred 32x32 bit multiplier design was synthesized and simulated.

2) Design #2: Booth-2 Algorithm Multiplier with CLA adder

A 32-bit multiplier design that uses the Booth-2 algorithm to reduce the number of partial products was synthesized and simulated. The verilog implementation consists of three stages:

1. Compute 16 partial products (using Booth-2 reduces partial products by half)
2. Compression of partial products into two 64-bit rows.
3. Final fast 64-bit carry look-ahead adder to compute the final product.

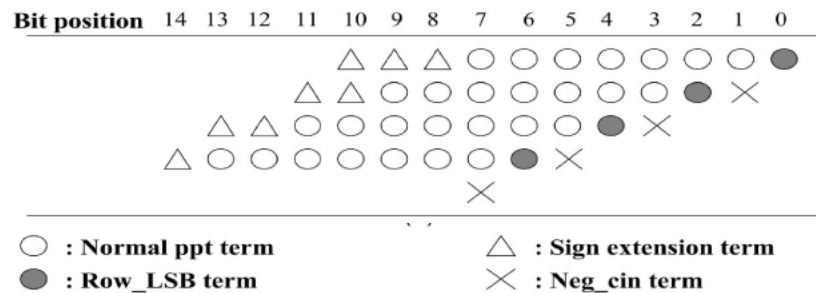
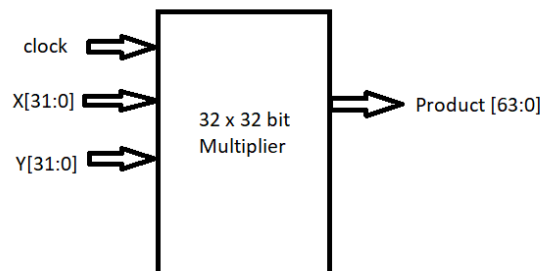


Figure 1: Partial Products for 8-bit Multiplier using booth algorithm

Multiplier Block Diagram

Both the designs have 65 input pins and 64 output pins.



Verilog Code Description

The full Verilog code used for both designs can be found in the Appendix. For Design #1, the verilog is simple and entirely behavioral, letting the synthesizer do the hard work of multiplication. Design #2 uses a hierarchical approach to specify and generate the logic. Modules are written to generate the partial products, compressors, and CLA adder. The top level multiplier module instantiates many copies of these modules. A parameter "WIDTH" was used to specify the number of bits in the multiplier, and generate-for loops were used to instantiate many modules.

Partial Products:

A 32-bit Booth-2 multiplier will have 16 partial products (PP). To compute each PP a module "booth2PP_32b" was created. It uses the multiplicand and 3 of the multiplier bits to correctly compute a single partial product. Signals "neg", "single", and "double" are determined by the 3 multiplier bits based on the Booth-2 method. Then, the partial product is modified based on these signals:

- If neither the "single" or "double" is true, then the PP is all 0's.
- If the "double" signal is true, then the PP is shifted left.
- If the "neg" signal is true, then the PP is inverted and a 1 is added.

Each partial product is 33 bits. The extra bit accounts for shifting left. Since there are 16 partial products for a 32-bit multiplier, our design instantiates 16 of the Booth-2 modules. Using Booth-2 is beneficial because it halves the number of PPs and reduces the size of the compressors.

Compressors:

A "compressor" module was created to compress each PP column into only 2 rows of bits. This is simply a large tree of full adders. For our design, we used a 17 bit compressor for each column of PPs. The 17th bit is required for certain cases of the multiplier. For $n = 17$, each module contains 15 full adders, and has 14 pass-carries in addition to the outputs "Sum" and " C_{drop} ". Since the product will have 64 bits, our design instantiates 64 of these compressors. The sign extend bits are a function of the MSB of a PP and are added as inputs to certain compressors. The extra "add 1" for computing the 2's complement of a PP is also added as an input to certain compressors. A carefully crafted generate-for loop is used to instantiate all the compressors.

Admittedly, using 17-bit compressors for all columns is inefficient because many columns will always have 0's as their inputs, so some of the full adders will always be adding 0. This is actually a trade off between design complexity and area/power consumption. To get the minimum area, more effort would need to be put into organizing these compressors and avoid adding all the 0's.

Carry-Lookahead Adder

The final part of the multiplier is the fast adder. A carry-lookahead tree module is implemented to precompute all the carries to speed this up. For a 64-bit adder, the lookahead trees get quite large and are only tractable in verilog through careful use of generate loops. In an attempt to minimize overall delay at the expense of area and power, 16 lookahead trees were used. The leftmost bit of each tree being the 16 MSBs. For 64-bits, each tree consisted of 6 layers ($2^6 = 64$).

Each lookahead tree consists of nodes which represent group “Generates” and “Propagates”. A verilog module was created to instantiate each tree, which computes these Generates and Propagates.

Logical Formulas for computing Generates and Propagates

	Generate	Propagate
Single bit	$G^0_0 = A \& B$	$P^0_0 = A + B$
Group	$G^1_0 = G^0_1 + (P^0_1 \& G^0_0)$	$P^1_0 = P^0_1 \& P^0_0$

The verilog module for the carry-lookahead adder instantiated 16 trees and uses them to compute each carry without the need for a ripple carry. For example, the 16 MSB carries can be found using the top level group generate of each tree:

$$\text{Carry \#63} = G^6_0$$

(since there is no carry-in for this group, the propagate term is not needed).

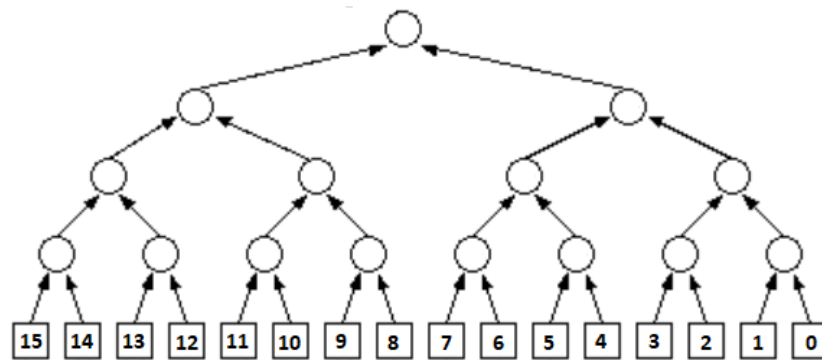


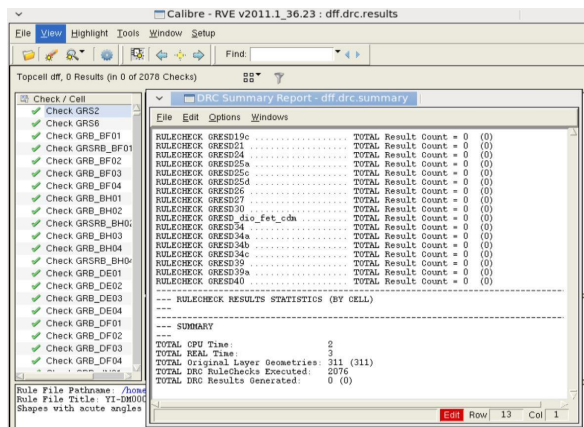
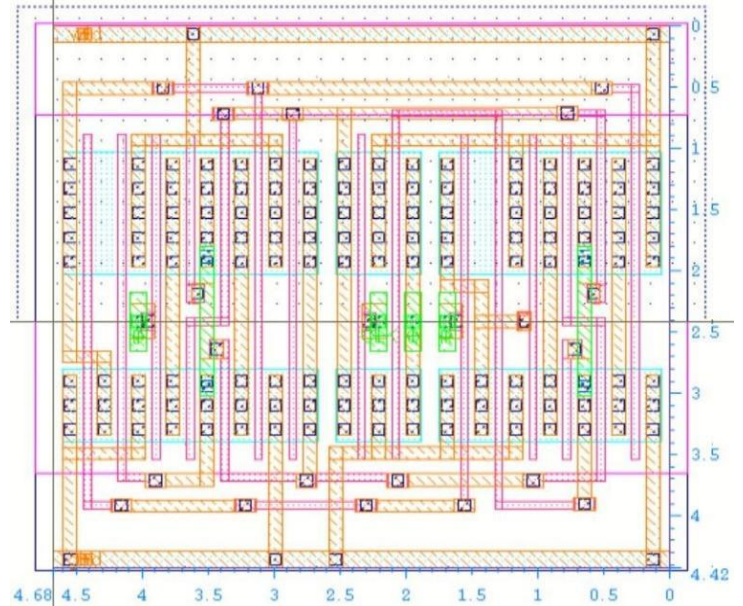
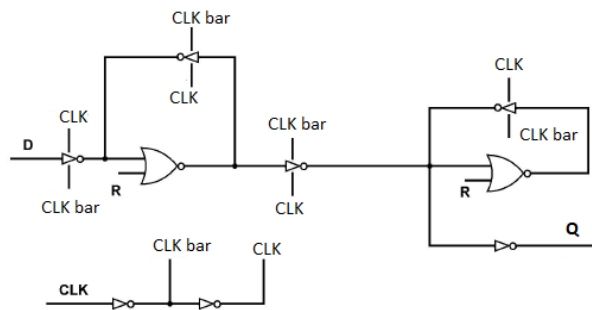
Figure 2: Example of 16-bit Carry Lookahead tree



The precomputed carries feed into 64 full adders. The output of this 64-bit adder is the final product, which gets latched into a register and completes the multiplier.

Cell Library : Layout, DRC and LVS Reports

The standard cell library was created using the 65nm process. Each of our cells have a pin pitch of 0.26um and a height of 4.42um. Schematic and layout views were generated for each cell, and DRC, LVS, and PEX were run.


DFF Design (Positive edged FF with Reset Pin)



Layout Cell / Type	Source Cell	Nets	Instances	Ports
 dff 	dff	16L, 16S	13L, 13S	6L, 6S

Cell dff Summary (Clean)

CELL COMPARISON RESULTS (TOP LEVEL)



LAYOUT CELL NAME: dff

SOURCE CELL NAME: dff

INITIAL NUMBERS OF OBJECTS

	Layout	Source	Component Type
	-----	-----	-----
Ports:	6	6	
Nets:	18	18	
Instances:	15	15	MN (4 pins)
	15	15	MP (4 pins)
Total Inst:	30	30	

DFF Measured Timing characteristics

	Transition to 0	Transition to 1
T_{su_dd}	20ps	56ps
T_{hold}	-55ps	-19ps
$T_{clk \rightarrow Q}$	251ps	276ps



Figure 3: D-Flip Flop Measurement for $T_{su_dd}(0)$

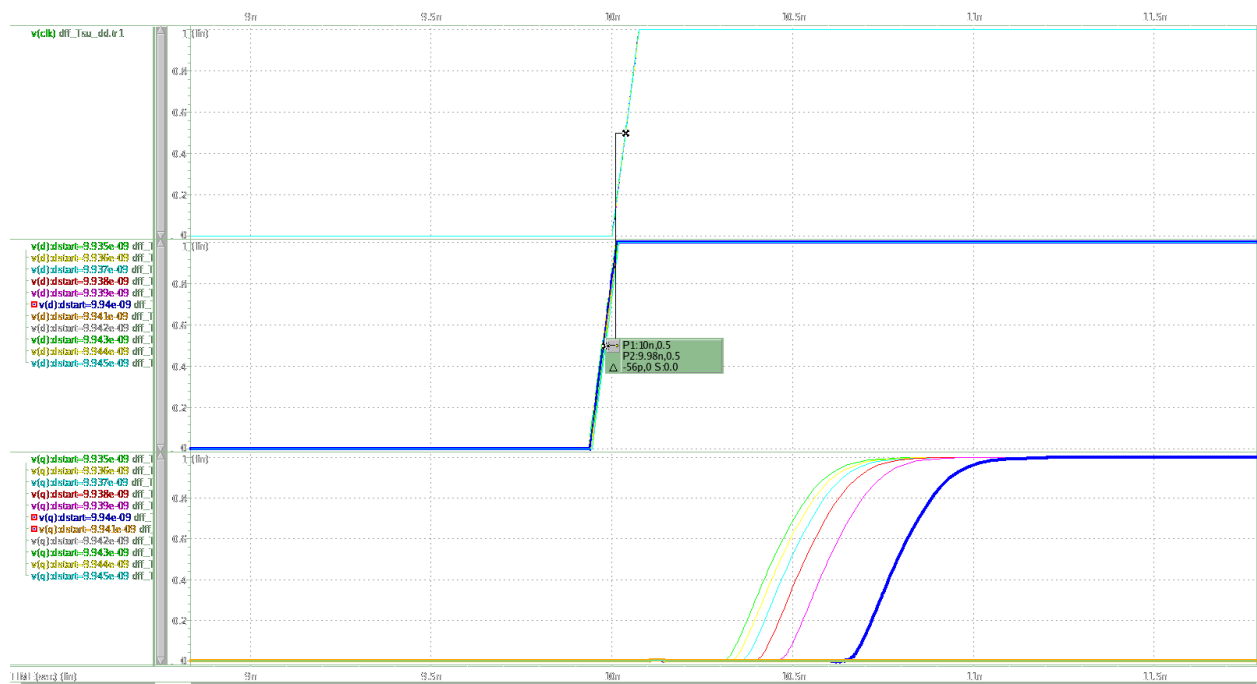
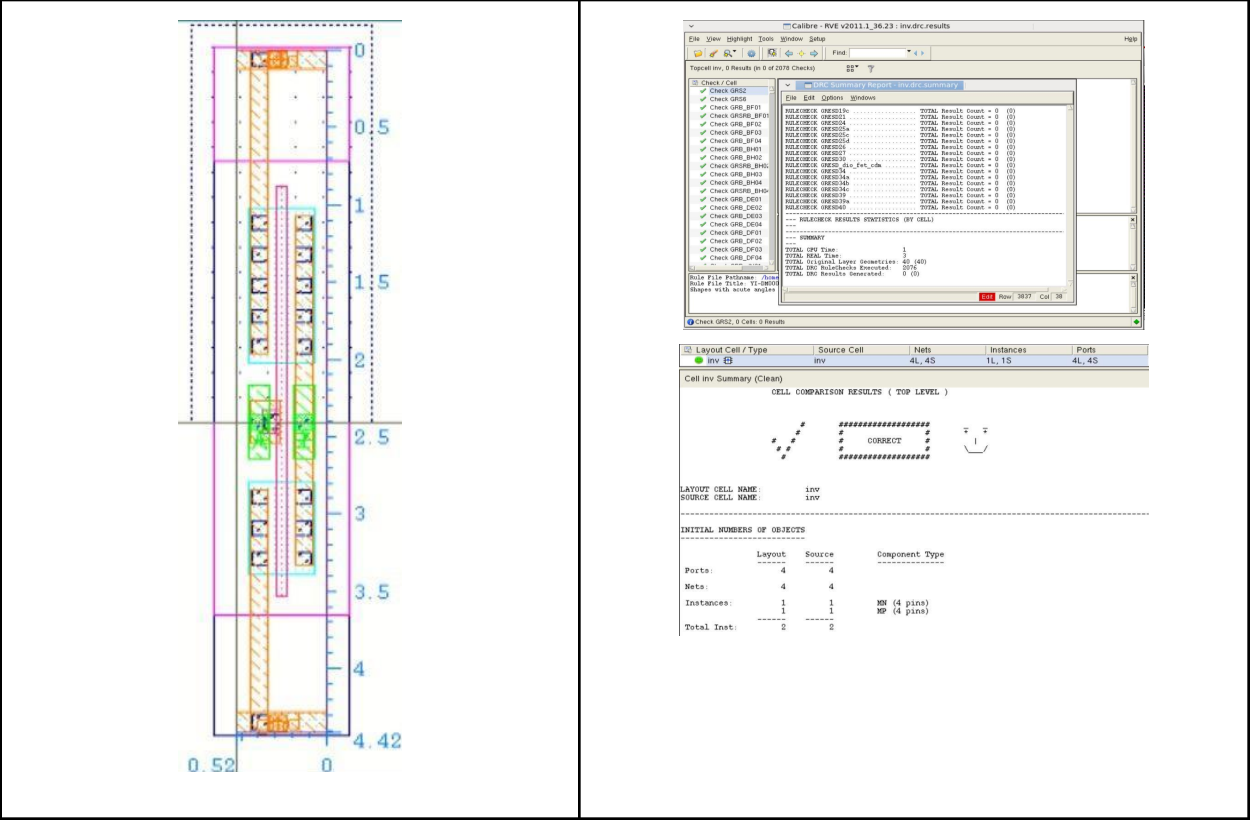
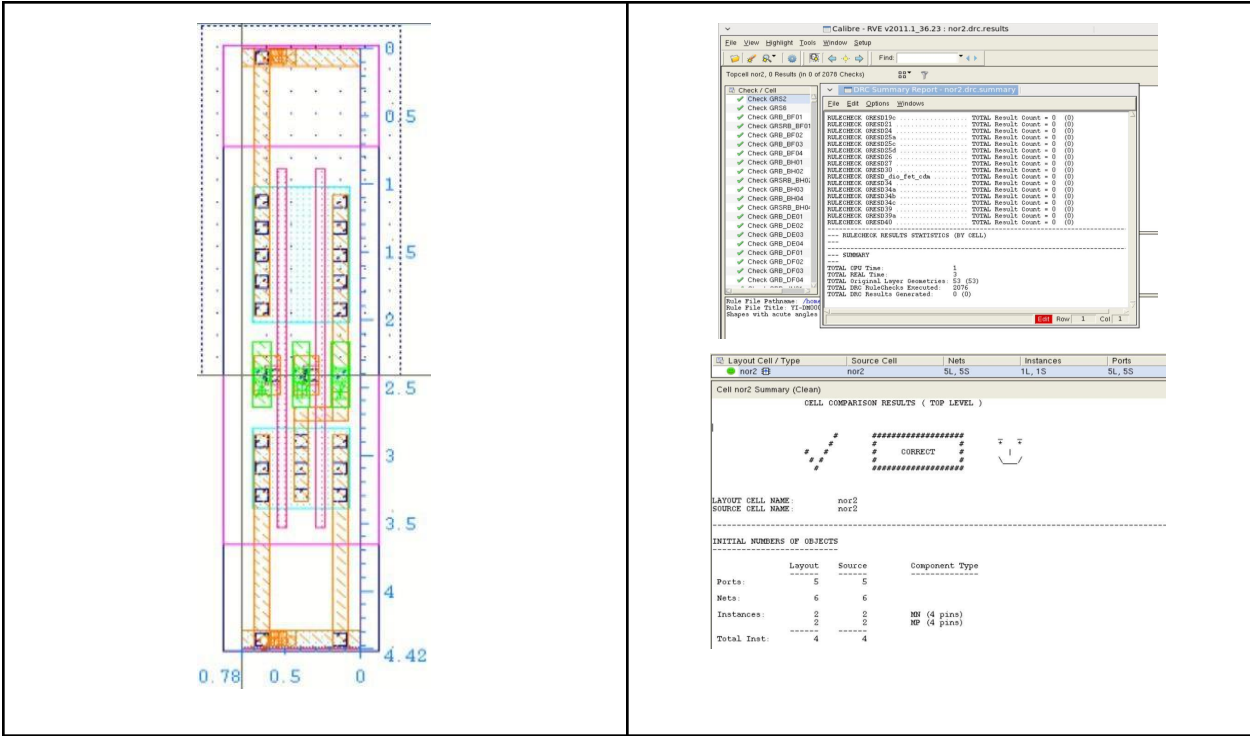


Figure 4: D-Flip Flop Measurement for $T_{su_dd}(1)$

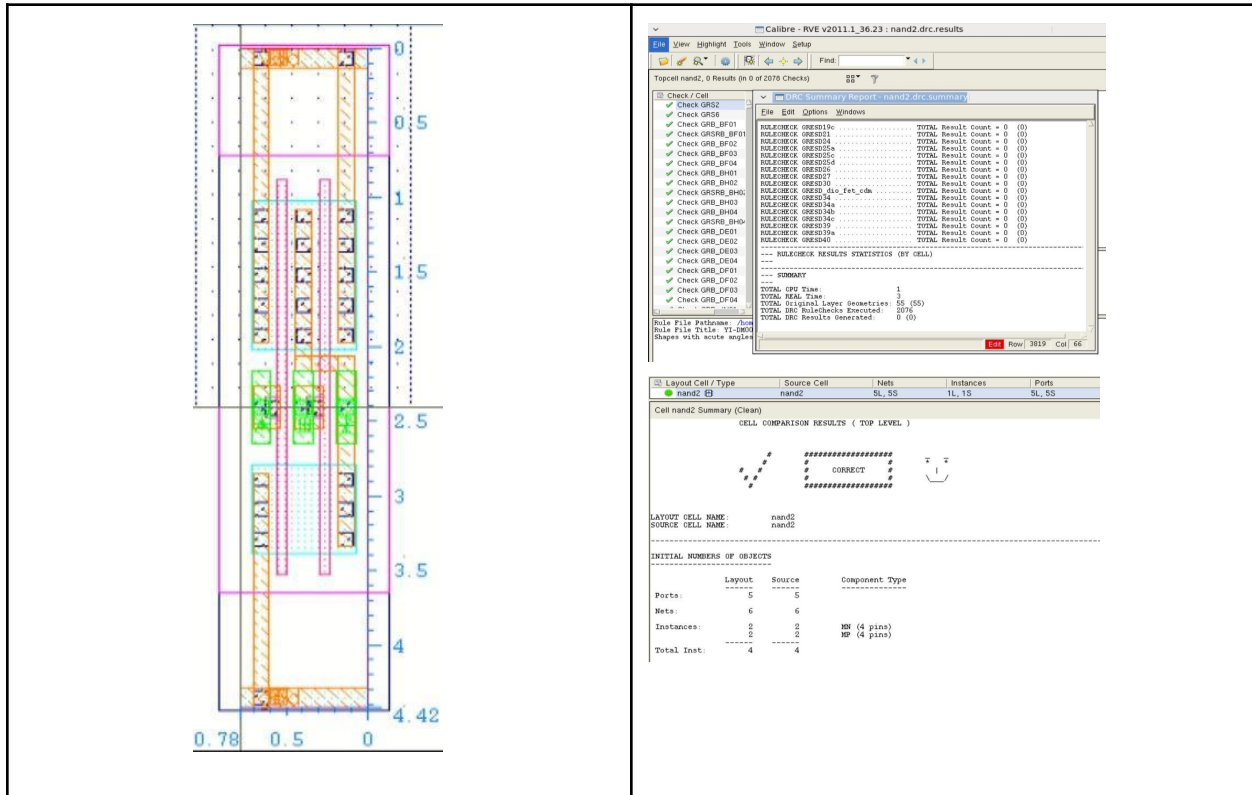
inverter



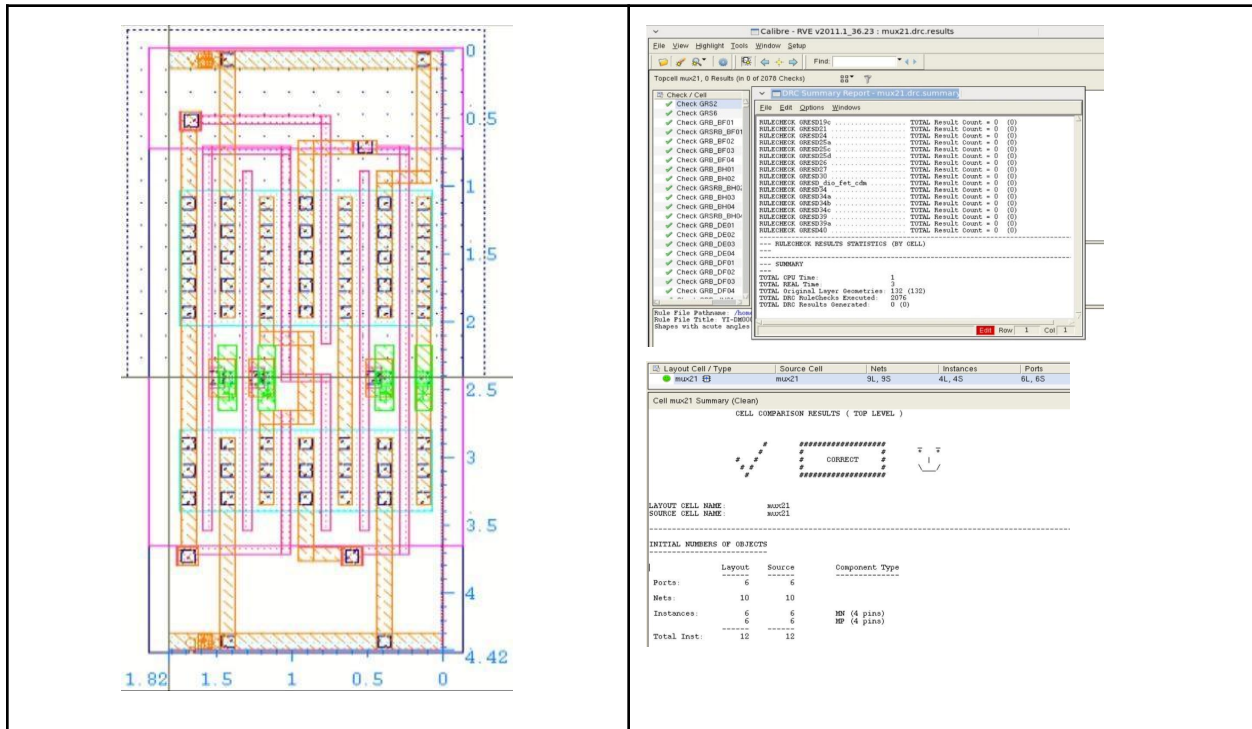
nor2



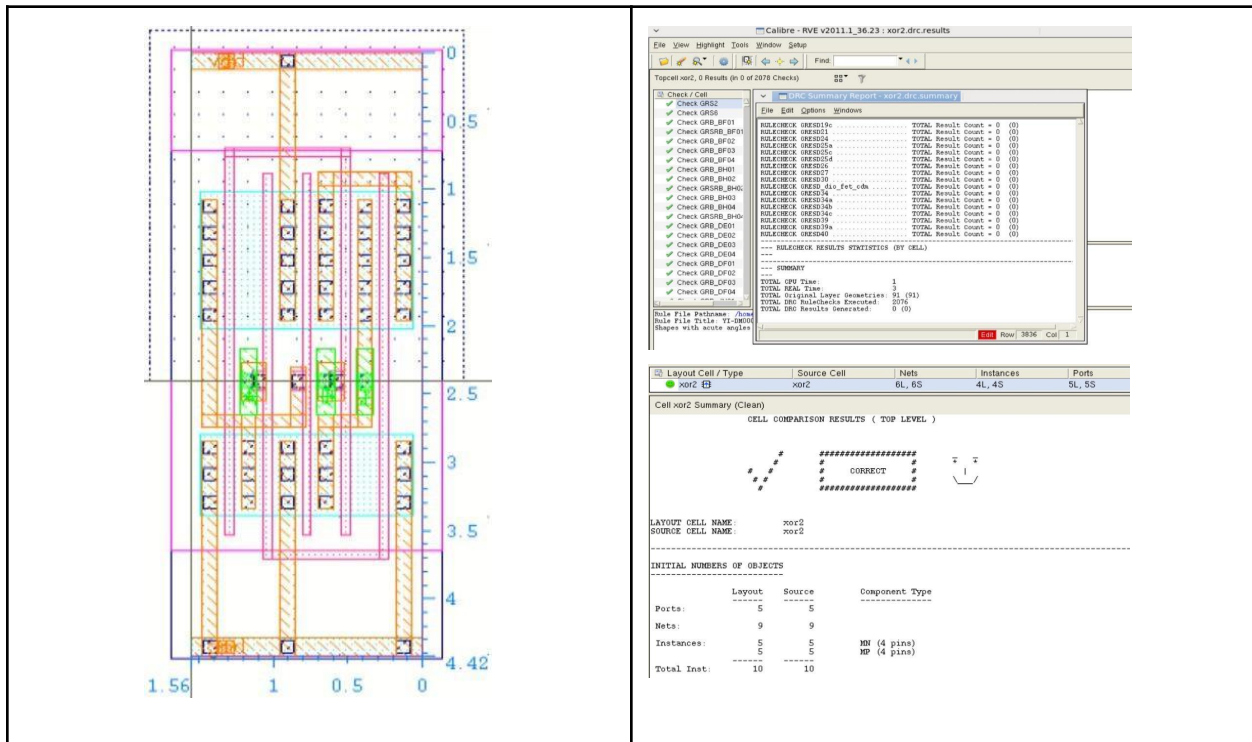
nand2



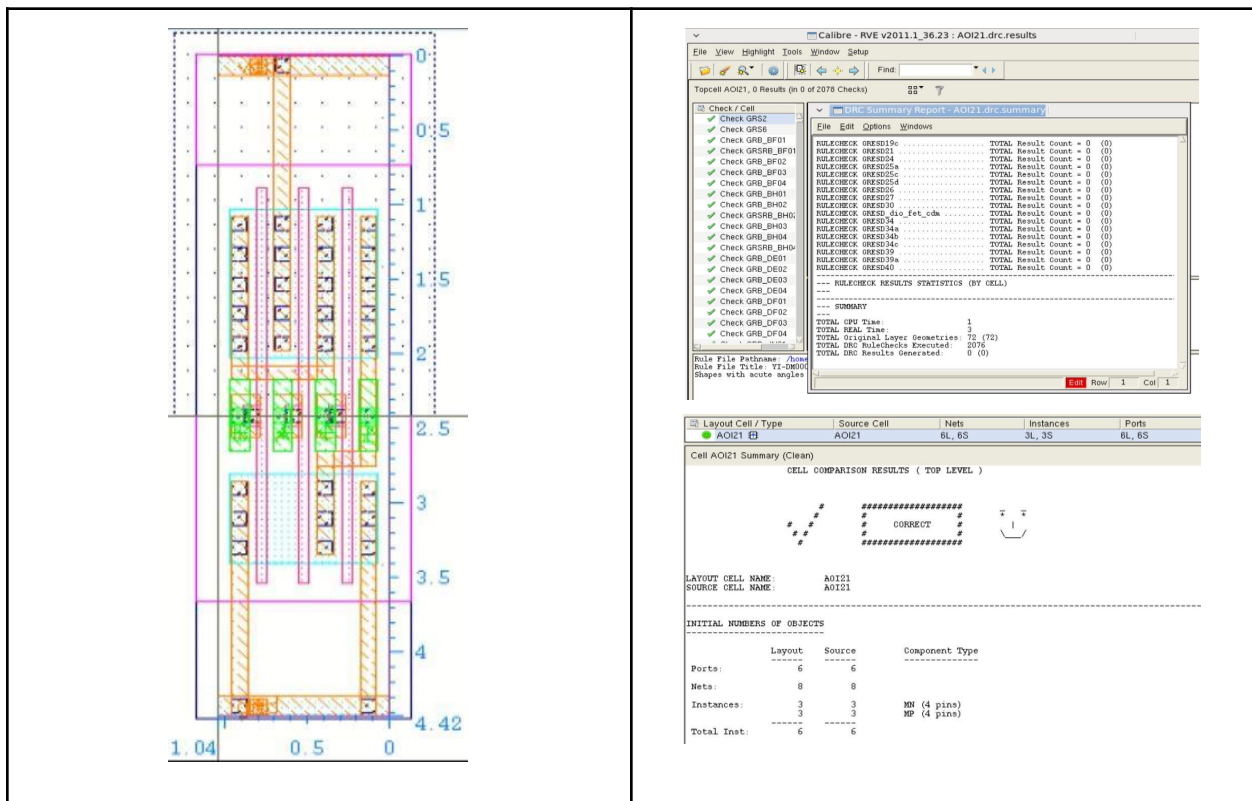
mux21



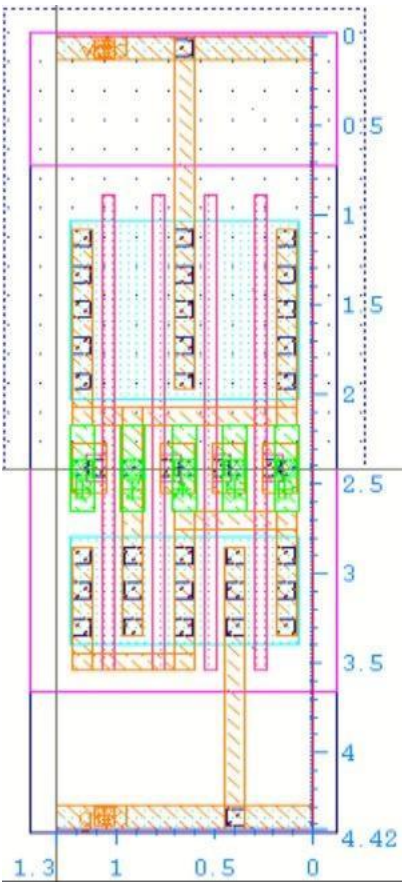
Xor



AOI21



OAI22



Calibre - RVE v2011.1.36.23 : OAI22.drc.results

File View Highlight Tools Window Setup

Find

Topcell OAI22, 0 Results (in 0 of 2078 Checks)

Check / Cell

- ✓ Check GR52
- ✓ Check GR52
- ✓ Check GR5B_BF01
- ✓ Check GR5B_BF02
- ✓ Check GR5B_BF03
- ✓ Check GR5B_BF04
- ✓ Check GR5B_BH01
- ✓ Check GR5B_BH02
- ✓ Check GR5B_BH03
- ✓ Check GR5B_BH04
- ✓ Check GR5B_BH05
- ✓ Check GR5B_DE01
- ✓ Check GR5B_DE02
- ✓ Check GR5B_DE03
- ✓ Check GR5B_DE04
- ✓ Check GR5B_DF01
- ✓ Check GR5B_DF02
- ✓ Check GR5B_DF03
- ✓ Check GR5B_DF04

Rule File Pathname: /home/...
Rule File Title: VI-DE000
Shape with acute angles

DRG Summary Report - OAI22.drc.summary

File Edit Options Windows

Rule	Result Count	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423	1424	1425	1426	1427	1428	1429	1430	1431</
------	--------------	---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	--------

Library Characterization

After the standard cell library is completed and layouts are extracted, the library must be characterized with the Siliconsmart tool. Doing this generates a library with timing and power information about all the cells. The script runs simulations on each cell for various input slopes on all input pins and creates an output profile array from the results. After characterization, the library can be used by other tools, like Primetime, to accurately determine cell timings and power consumption with given input slopes.

This characterization process was a bit difficult in the 65nm process, since scripts had to be modified in order to work properly. The modified “siliconsmart.pl” script is included in the appendix. Here are some things we learned during the process:

1. The input, output and gnd pins are set as inout pins in the inst files by default. The siliconsmart.pl had to be modified for assigning the inputs as input pins, outputs as output pins and power pins (VDD and GND) as supply pins. The script had to be changed for dff to assign input D, clock and reset pins as input pins. Output Q as output pin and power pins (VDD and GND) as supply pins.
2. The technology file sourced in the configure.tcl script should be gf65. The name of configure.tcl has to remain the same. The Siliconsmart ACE tool does not recognize the configure script if the name changes.
3. The pin names are converted to uppercase by the Siliconsmart ACE tool. It is important to change the pin labels in the layout to uppercase so that they match with the standard library file.

Design Synthesis

With our standard cell library complete, the verilog code is synthesized using the Design Vision tool into a structural netlist of individual logic gates from our cell library. Doing this allows the place and route tool to realize the logic using only standard gates.

Design #1: Behavioral Multiplier	7658 Cells
Design #2: Booth-2 Algorithm	23,598 Cells

Verilog Waveform Verification of Synthesized Designs

After synthesis, the structural netlist functionality is checked using the same verilog testbench. Below are waveforms showing that the synthesized designs function correctly and give the same outputs.

1) Behavioral Multiplier (X*Y)

_32x32/Y	50	50		5		2		0		4294967		4012967		2012558		888888		-555555		3685896		3...
_32x32/X	-3	-3		-6		3		63		4294967		31568		43333		888888		777786		3458678		3...
_32x32/clk	1																					
_32x32/P_behav	x			-150		-30		6		0		18446741531089		126681342256		87210175814		790121876544		-432102901230		1...
_32x32/P_behav_syn	x			-150		-30		6		0		18446741531089		126681342256		87210175814		790121876544		-432102901230		1...

2) Booth-2 Algorithm Multiplier

_32/Y	50	50		5		2		0		4294967		4012967		2012558		888888		-555555		3685896		3...
_32/X	-3	-3		-6		3		63		4294967		31568		43333		888888		777786		3458678		3...
_32/clk	-1																					
_32/P	x			-150		-30		6		0		18446741531089		126681342256		87210175814		790121876544		-432102901230		1...
_32/P_syn	x			-150		-30		6		0		18446741531089		126681342256		87210175814		790121876544		-432102901230		1...

Primitime was used for timing and power analysis of both synthesized designs. Primitime uses the information from a standard cell library along with the synthesized structural netlist to simulate critical path timings and power consumption for the design. In order for the script to work properly, Primitime needs a path between two clocked DFFs or else the paths would be unconstrained. In the verilog code, we had to ensure there were registers on both the input and output bits.

Below are the results of the primetime script. The data arrival time reported is the worst case path for any two flip flops, so after this time all the output bits should be correct.

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
clock network register	1.623e-04 -9.797e-05	0.0000	0.0000	1.623e-04	(9.14%)	i
combinational sequential memory io_pad black_box	1.635e-04 0.0000 0.0000 0.0000 0.0000	1.366e-03 0.0000 0.0000 0.0000 0.0000	2.604e-07 0.0000 0.0000 0.0000 0.0000	1.268e-03 3.446e-04 0.0000 0.0000 0.0000	(71.44%) (19.42%) (0.00%) (0.00%) (0.00%)	
Net Switching Power	= 1.546e-03	(87.11%)					
Cell Internal Power	= 2.278e-04	(12.83%)					
Cell Leakage Power	= 9.768e-07	(0.06%)					
Total Power	= 1.775e-03	(100.00%)					

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
clock network register	1.623e-04 -9.805e-05	0.0000	0.0000	1.623e-04	(6.72%)	i
combinational sequential memory io_pad black_box	4.094e-04 0.0000 0.0000 0.0000 0.0000	1.373e-03 5.686e-04 0.0000 0.0000 0.0000	2.604e-07 7.684e-07 0.0000 0.0000 0.0000	1.275e-03 9.788e-04 0.0000 0.0000 0.0000	(52.78%) (40.50%) (0.00%) (0.00%) (0.00%)	
Net Switching Power	= 1.942e-03	(80.36%)					
Cell Internal Power	= 4.736e-04	(19.60%)					
Cell Leakage Power	= 1.029e-06	(0.04%)					
Total Power	= 2.416e-03	(100.00%)					

Figure 5: PrimeTime Results for Behavioral Design #1 (Left) and Booth-2 Algorithm Design #2 (Right)

	Critical Path data arrival time	Total Power Consumption
Design #1	13.61ns	1.775mW
Design #2	10.39ns	2.416mW

As expected, Design #2 with Booth-2 and CLA adder is faster than the purely behavioral Design #1. Design #2 is about 24% faster, but uses 36% more total power. This seems like a reasonable tradeoff. It is interesting to compare where the power is used. Both designs use the same amount of power for the clock network and registers, the difference is only in the combinational logic. Design #2 uses 2.84 times more power in the combinational logic, which is mostly due to the large carry lookahead trees.

Placement and Routing

The Library Exchange Format file (LEF) should be extracted from virtuoso and a header file should be added to this. The encounter can read only the LEF file to understand the cell library created in virtuoso. The LEF header file that was added includes design rules like minimum width, minimum area and pitch size of metal layers and the LEF file includes abstract information of the cells in the cell library.

- Edited the via rules and pitch size in the LEF to use 65nm sizes.
- Edited pitch size and cell height in the header LEF.
- Load the LEF file and synthesized verilog file in Encounter.
- Specified the floor plan with Aspect Ratio as 1, core utilization as 0.7 and core margin as 10.
- Placed the standard cells (medium congestion effort) and filler cells in rows with a row spacing of 1.04um.
- Connected Tie-hi nets to global vdd and Tie-lo nets to global ground.
- Placed the Power Rings (width of ring: 0.78 and spacing between the rings: 1.3) and Power Stripes for every row.
- Route->Nanoroute for the routing.
- pincover.tcl was run in the terminal.
- The file was saved as Design Exchange Format file (DEF)
- Because Encounter does not export the via definitions to the DEF, a perl (addvias.pl) script was used for adding the vias as per the editing done in the LEF file.
- The DEF file is imported into virtuoso.
- A cadence SKILL script called "pins.il" was written and run from the CIW terminal to add the input and output pin labels to the layout view. This was needed so that the layout and schematic had the same number of ports in order to pass LVS. The Calibre tool performs the LVS test based on the pin labels rather than the pins placed, so having pin labels for all the input and output pins is important.

Layout, DRC and LVS Reports of the Final Design

- In order to pass DRC, had to update via object sizes to 65nm rules in the .lef and .def (by modifying addvias.pl)
- To pass LVS on Design #2, had to replace some wire names in the synthesized netlist because hspice is case sensitive, but Calibre is not.

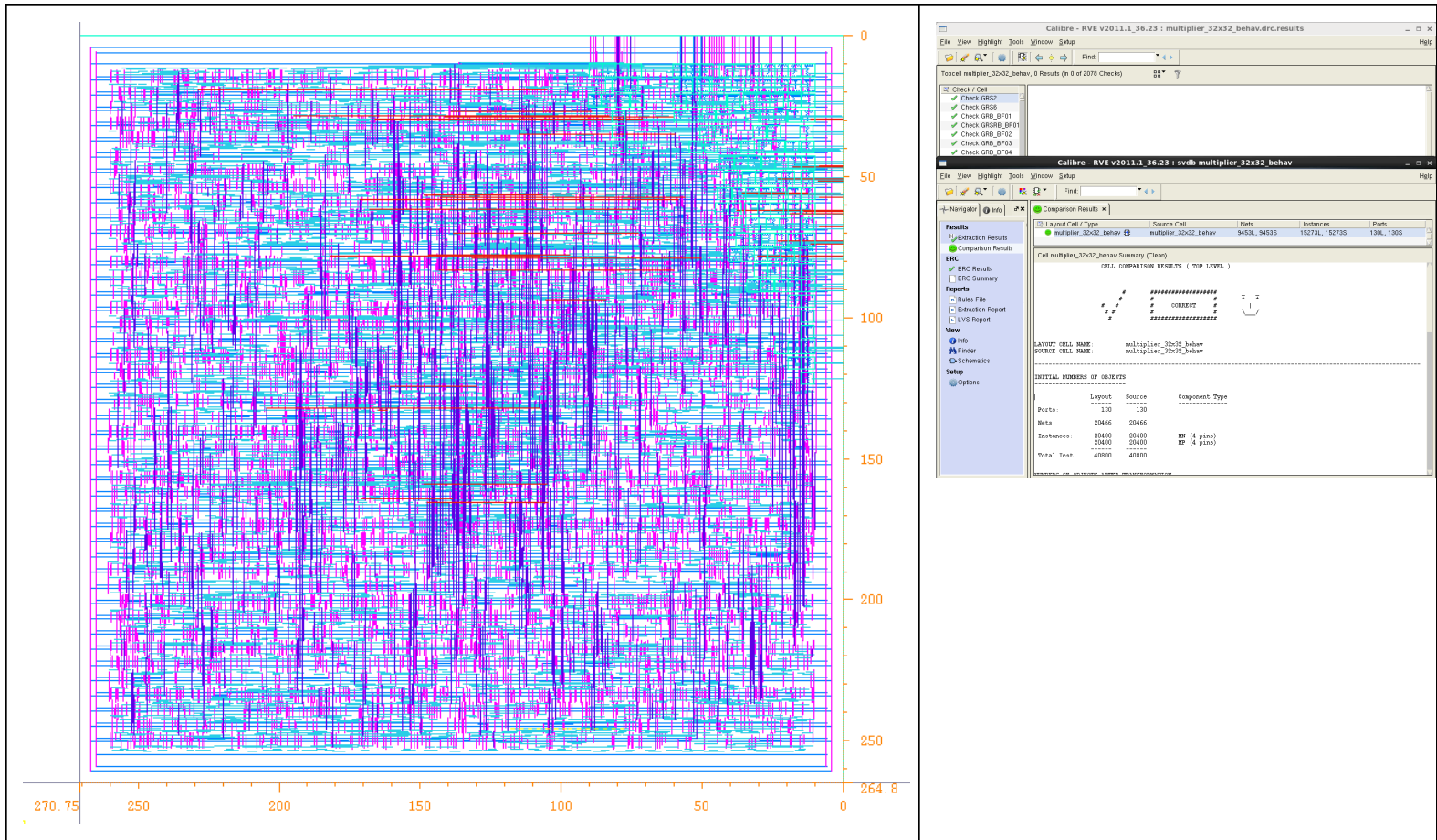


Figure 6: Final Layout, DRC report and LVS report for Design #1 Behavioral Multiplier

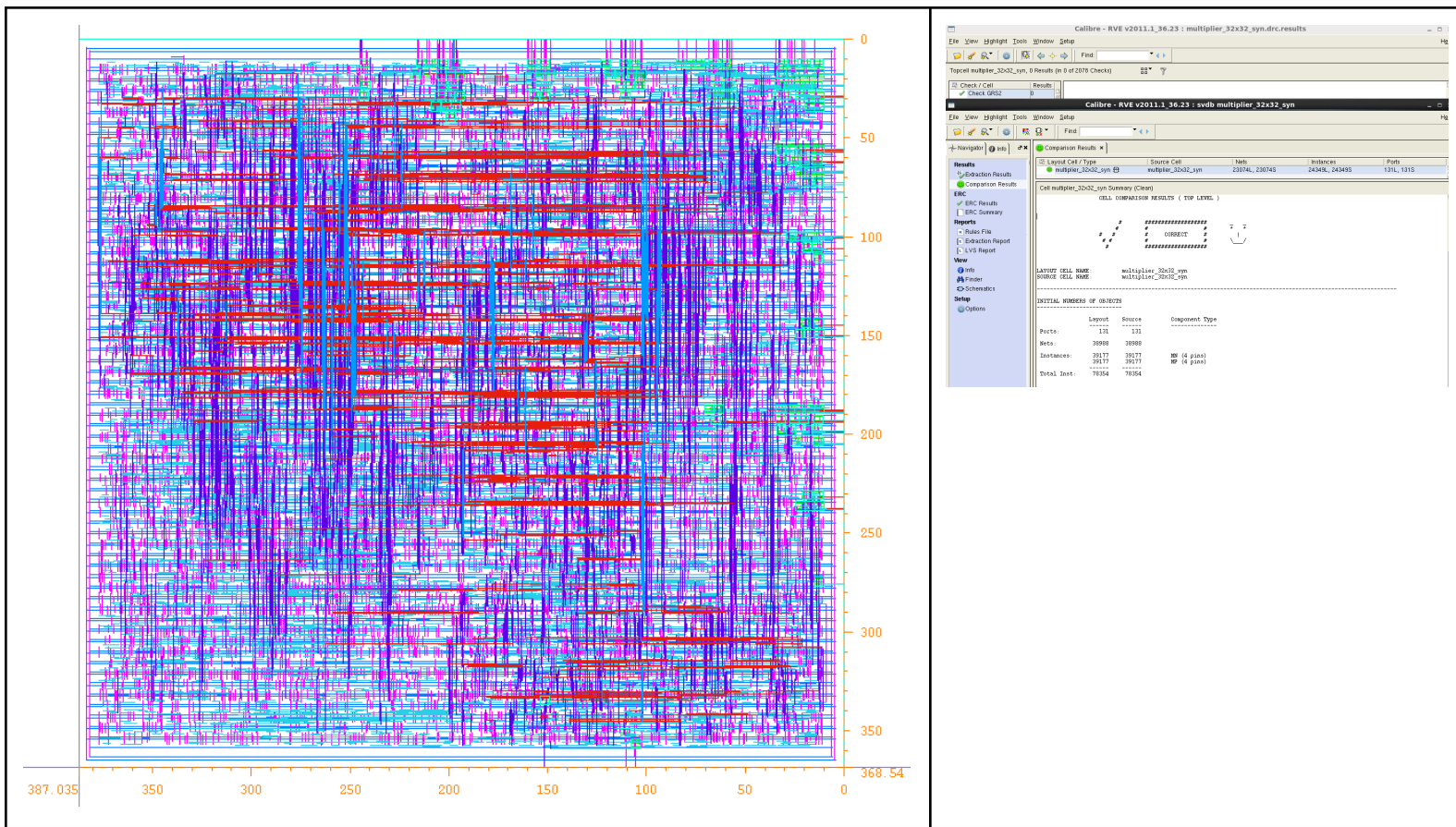


Figure 7: Final Layout, DRC report and LVS report for Design #2 Booth-2 Algorithm Multiplier



“Amoeba” view of Design #2, showing how each module was placed into the layout. The final adder with lookahead trees takes up nearly $\frac{1}{4}$ of the total area.



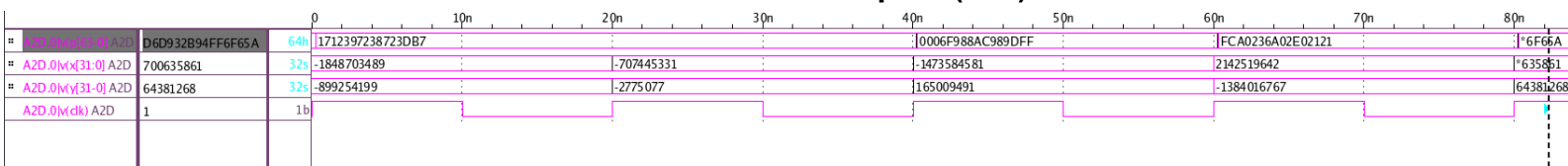
“Amoeba” view detail, showing numerous compressors and Booth-2 PP modules

Design Verification in Hspice

After the chip layout is complete, the layout is extracted to a spice netlist and the final functionality of the chip is verified using Hspice. The Hspice test file used is included in the Appendix: “multiplier_verification.sp”. Output pins were loaded with 25fF. Each input pin was controlled by using the PWL hspice function.

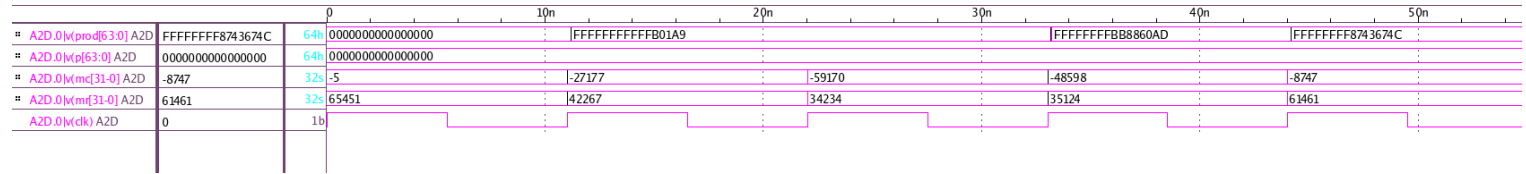
Below are waveforms showing the correct functionality of both multiplier designs. For the 32-bit inputs, large numbers were chosen to exercise most of the bits. The product is reported in hexadecimal because WaveView does not display 64-bit signed decimal numbers correctly. On the first clock rising edge, the inputs are latched into the input registers. On the next clock rising edge, the outputs are latched into the output register, at the same time new inputs are latched into the input register. In this way, there is 1 clock cycle of delay before the output product becomes correct.

Behavioral Multiplier (X*Y)



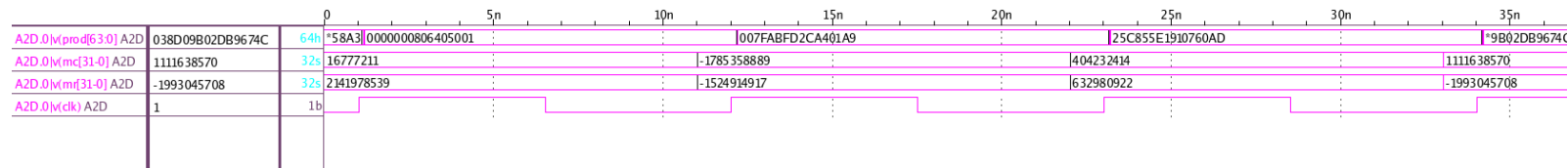
Clock Edge #	Time Index (ns)	Input Y (multiplicand)	Input X (multiplier)	Output Product (Hexadecimal)	Output Product (Decimal)	Correct Output?
1	20	-899,254,199	-1,848,703,489	1712 3972 3872 3DB7	1,662,454,375,189,200,311	Y
2	40	-2,775,077	-707,445,331	0006 F988 AC98 9DFF	1,963,215, 266,815	Y
3	60	165,009,491	-1,473,584,581	FCA0 236A 02E0 2121	-243,155,441,656,258,271	Y
4	80	-1,384,016,767	2,142,519,642	D6D9 32B9 4FF6 F65A	-2,965,283,108,154,837,414	Y

Booth-2 Algorithm Multiplier



Clock Edge #	Time Index (ns)	Input MC (multiplicand)	Input MR (multiplier)	Output Product (Hexadecimal)	Output Product (Decimal)	Correct Output?
1	25	-5	65,451	FFFF FFFF FFFB 01A9	-327,255	Y
2	45	-59,170	34,234	FFFF FFFF 8743 674C	-2,025,625,780	Y

Design #2 was also simulated using a clock of 11ns to show functionality at this increased speed.



Clock Edge #	Time Index (ns)	Input MC (multiplicand)	Input MR (multiplier)	Output Product (Hexadecimal)	Output Product (Decimal)	Correct Output?
1	12	16,777,211	2,141,978,539	007F ABFD 2CA4 01A9	35,936,425,906,274,729	Y
2	23	-1,785,358,889	-1,524,914,917	25C8 55E1 9107 60AD	2,722,520,402,034,647,213	Y
3	34	404,232,414	632,980,922	038D 09B0 2DB9 674C	255,871,406,116,005,708	Y

32-bit Multiplier Design Trade-Offs

The Booth-2 Algorithm Multiplier which uses a Carry Lookahead Adder for the final 64-bit addition is faster as compared to the Behavioral Multiplier Design. The additional hardware increases the speed by 24%. However, it increases the Total Power Consumption by 36%. The area for Design #2 is twice that of Design #1. This meets our expectation that increasing speed by using a CLA adder is very costly in terms of power and area, however it is likely still a good trade off for many applications with modern technology processes.

The Booth-2 Algorithm Multiplier uses 15 Full Adders for each column in the “compressor” stage to reduce 16 Partial Products to two rows. The 64 columns use a total of 960 Full Adders. Many of these Full Adders add zeroes. The design can further be improved by using fewer adders in the process. This would reduce the area and the power consumption, but would require significant effort to fine tune the design.

Below is a table summarizing the two designs. It is easy to see that the increased speed of Design #2 is paid for with area and power.

	Number of Cells	Total Area (sq. um)	Total Power Consumption (mW)	Data Arrival Time (ns)
Design #1 (Behavioral)	7,658	71,694.6	1.775	13.61
Design #2 (Booth-2 Algorithm)	23,598	142,637.9	2.416	10.39