

Lab 3: Designing a power efficient architecture for 4-Bit Adders

- a report by

**Adithya Anand | A59010781
Gandhar Deshpande | A59005457**

**CE, Department of
Electrical And Computer Engineering**

**Prepared in the partial fulfillment of
ECE 260A, in conjunction with taking the final comprehensive
examination.**



University of California, San Diego

Quarter 1, 2021 - 22

CONTENTS

1. Reference Architecture
 - 1.1 Summary
 - 1.2 Netlist Diagram
 - 1.3 Metrics
2. Proposed Architecture
 - 2.1 Design choices over Reference Architecture
 - 2.2 Netlist Diagram
 - 2.3 Metrics
 - 2.4 What did not work as expected
 - 2.5 Future improvements in design

1. Reference Architecture

1.1 Summary

We've taken the reference architecture to be a cascaded ripple carry adder.

Implemented in System Verilog, we've introduced a gate-level implementation of a cascade of full adders. The number of full adders is parameterised and introduced by the test bench. It runs the following rubric:

1. ar, br, cr, dr are the four states of the variables.
 - a. In each clock cycle, $a \rightarrow ar$, $ar \rightarrow br$, $br \rightarrow cr$, $cr \rightarrow dr$.
2. $sum = s3$ where $s3 = s2 + dr$, $s2 = s1 + cr$, $s1 = ar + br$
 - a. $cin[i] = cout[i-1]$ (**for each stage**)
 - b. $s1 = ar[i] \wedge br[i] \wedge cin[i]$ at any clock cycle.
 - i. $cout[i] = ((ar[i] \wedge br[i]) \& cin[i]) \vee (ar[i] \& br[i])$
 - c. $s2 = s1[i] \wedge cr[i] \wedge cin[i]$ at any clock cycle
 - i. $cout[i] = ((s1[i] \wedge cr[i]) \& cin[i]) \vee (s1[i] \& cr[i])$
 - d. $s3 = s2[i] \wedge dr[i] \wedge cin[i]$ at any clock cycle
 - i. $cout[i] = ((s2[i] \wedge dr[i]) \& cin[i]) \vee (s2[i] \& dr[i])$

1.2 Netlist Diagram

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
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------

1.3 Simulation Metrics

<u>Metrics</u>	<u>Reference Architecture</u>
Worst negative slack in ps (setup analysis)	-0.37
Total power consumed	1.96 mW
Leakage power	29.15 uW
Area of combinational logic	2242.44
# of combinational cells	961
Total area	2846.52

2. Proposed Architecture

2.1 Design choices over Reference Architecture

We considered the following architectures to implement the adder.

1. Ripple-carry adder in a tree-based architecture.
2. A modified Ripple carry adder with a carry-lookahead adder
3. Behavioral model of an adder

Ripple Carry Adder

We implemented the ripple carry adder in a tree-based architecture. This does the addition in the same manner as the cascaded reference architecture, but it modifies the way the adders are placed, to give a better performance. The adders are now placed in a manner as shown below:

```
s1 = ar + br;  
s2 = cr + dr;  
sum = s1 + s2;
```

This gives some improvement in terms of latency as well as power and area numbers.

Carry lookahead Adder

We decided against the carry-lookahead adder for the reasons listed below:

1. The combinational logic for generating and propagating the carry of each stage would be too costly with respect to power and area consumption.
2. Carry lookahead adder was optimised only if the number of bits was precomputed. If we parameterize the number of bits, we have a general expression for generating the carry, which does not improve the efficiency as much. The general expression we wrote has been given below:

```
assign propagate[i] = ar[i] ^ br[i];  
assign generate[i] = ar[i] & br[i];  
assign carry[i+1] = generate[i] | (propagate[i] & carry[i]);  
assign sum[i] = propagate[i] ^ carry[i];
```

Behavioral model of an Adder

We decided to make complete use of System Verilog's powers, and use a behavioral model of an adder.

1. This would allow the DC Shell to convert our Verilog code into the most optimal RTL schematic due to the number of wires and registers being sparse.
2. We found that the DC Shell does optimization based on resource utilization and power.
3. This reduced the latency overhead compared to the cascaded model, and in turn, the negative slack associated with the adder.
4. On RTL synthesis and inspection of power and area consumption, the behavioural model trumped the reference architecture in all aspects.
5. While the timing is better improved with the ripple carry adder, it comes at a large expense of high resources, which is why we have chosen this model with a little higher latency but much better power and area utilization.

2.2 Netlist Diagram

2.3 Simulation Metrics

<u>Metrics</u>	<u>Ripple carry with tree</u>
Worst negative slack in ps (setup analysis)	-0.18
Total power consumed	1.84 mW
Leakage power	25.19 uW
Area of combinational logic	1820.52
# of combinational cells	826
Total area	2457.36

<u>Metrics</u>	<u>Carry-Lookahead Model</u>
Worst negative slack in ps (setup analysis)	-0.17
Total power consumed	1.88 mW
Leakage power	26.76 uW
Area of combinational logic	1877.4
# of combinational cells	814
Total area	2516.4

<u>Metrics</u>	<u>Behavioral Model</u>
Worst negative slack in ps (setup analysis)	-0.25
Total power consumed	1.61 mW
Leakage power	15.72 uW
Area of combinational logic	925.56
# of combinational cells	358
Total area	1519.2

2.4. What did not work as expected

1. Initially, we tried implementing a carry skip adder. The implementation required choosing between our initial carryIn and each stage's carryOut by multiplexing the inputs at each stage. Our implementation was faulted, and we shifted our focus on the carry-lookahead adder. Upon review, we found that our multiplexer's select was using a single wire which was getting updated over all stages, and we should have used an array instead.
2. The behavioral model optimized the design in terms of area, power and number of combinational units. However, it did not optimize the architecture in terms of latency as much as the tree structure of ripple carry adders.
3. We also considered using carry-select adders; however, carry-select adders have a larger area as compared to carry-lookahead adders, and would not provide as much an improvement in terms of timing due to the lesser number of bits used.

2.5 Future improvements in design

For future improvements, we can create a pipelined version of this adder. With pipelining, the number of combinational blocks between DFFs would reduce. This allows us to increase the frequency and also gives one output every clock cycle. Another option can be to use the transpose architecture of the filter which would create a directly pipelined version by putting the combinational blocks between the DFFs.

Contributions:

Gandhar: Carry Lookahead implementation, cascaded implementation

Adithya: Carry-skip implementation, behavioral implementation

Report and analysis : Both equal contributions.