

ESE345 Project Report

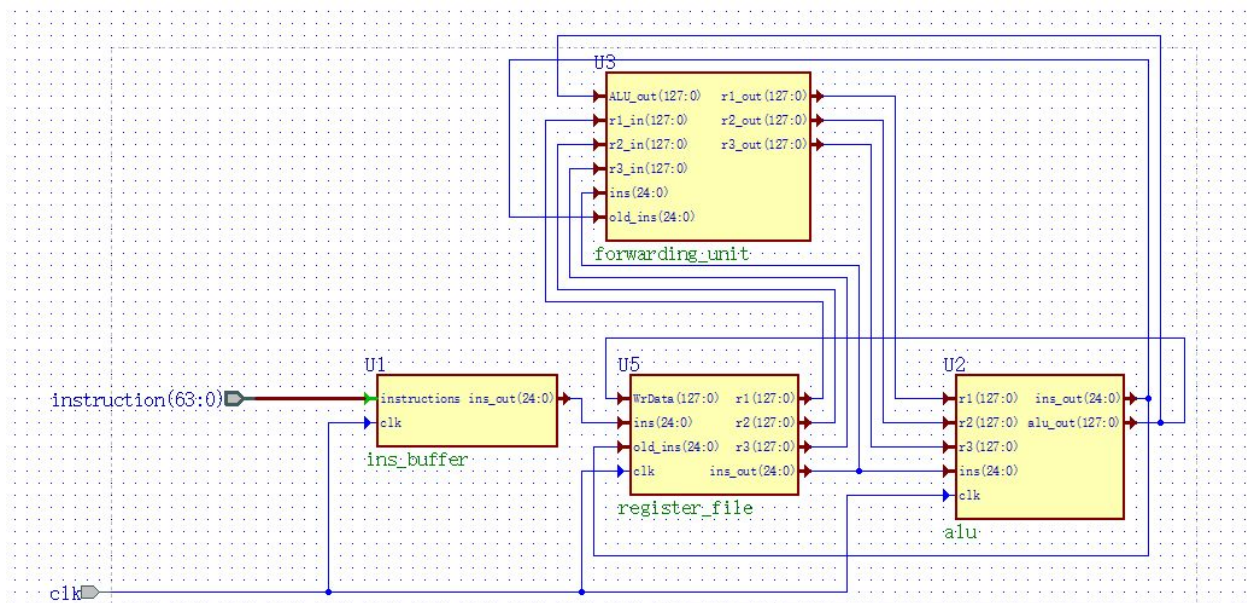
Fengwei Zhang #111252554

Gavin Zhou #111586947

1. Project Goal

This objective of the project is to learn the use of VHDL/Verilog hardware description language and modern CAD tools for the hierarchical gate-level and dataflow/RTL design of the 4-stage pipelined multimedia unit with a reduced set of multimedia instructions similar to those in the Sony Cell SPU and Intel SSE architectures.

2. Multimedia unit block diagram



3. Design procedure

3.1. ALU

In this module, there are five input which are 128 bit vector `r1`, `r2`, `r3` and 25 bit instruction and `clk`, and two outputs 25 bit `ins` and calculated 128 bit `alu_out`. Case statement based on instruction's 25 to 15 bit is used to tell which instruction is in this

cycle. The reason why there is a `ins_out` is like transferring the control signal to register file to tell register which register have to write into.

For the load instruction, first give `r1`'s value to the `alu_out` and then load immediate to the `alu_out`.

For the R4 type instruction, `temp1` and `temp2` which are short 32 bit vector variable and `temp3` and `temp4` which are long 64 bit vector variable are used to get the temporary product of `rs2` and `rs3` and sum of product of `rs2` and `rs3` and `rs1`. There might be saturation when add and subtract. For the add saturation case, there are two cases, one is positive number plus positive number to get a positive saturation and negative number subtract negative numbers to get a negative saturation. For the subtract saturation case, there are still two cases which are positive number subtract negative numbers to get a positive saturation and negative number subtract positive number to get a negative saturation.

For the R3 type instruction, there are some or, xor, rotate, left shift and right shift instruction which can use pre-defined function or, xor, ror, sll and srl. Saturation cases are like the R4 type instruction, while in the MSGN instruction there is only one saturation case which is the for the same number of bit maximum negative number is bigger than the maximum positive number.

3.2. Instruction Buffer

Declare a 64 capacity array to store 64 instructions. Every PC count 1 more then fetch next instruction.

3.3. Register File

There are write enable control signal to determine should write into register. The `old_ins` is transferred from alu to determine in last cycle which register have to write into.

3.4. Forwarding Unit

All instructions takes 4 clock cycles to complete, which will cause data hazard if an instruction uses a register that is recently modified in the earlier 2 clock cycles. The forwarding unit will detect this situation and delivers the most recent data of that register to the input of ALU to avoid data hazard.

3.5. Pipeline Unit

Pipeline unit is a structural unit in the project. It works as a bridge between each unit. Signals or Variables will be declared to store the data and instructions output

from one unit and send them to the next unit's inputs. These signals can be seen as the pipeline registers, they have the same function.

4. Testbenches

Imported two new libraries `STD.textio.all` and `ieee.std_logic_textio.all` to use built-in function to read txt file generated by assembly to binary instruction converter program written in Java. And still write the `alu_out` 128 bit data into txt file.

5. Conclusions

The final design was tested and verified for various input data and different op-codes using different inputs in the testbench. All the instructions functions as expected and the results generated match the theoretical results. So it was determined that the final design is a success.

Test Instruction List:

li r1,0,0000000000000001

li r1,1,0000000000000010

li r1,2,0000000000000011

li r1,3,0000000000000100

li r1,4,0000000000000101

li r1,5,0000000000000110

li r1,6,0000000000000111

li r1,7,0000000000001000

li r2,0,0000000000000001

li r2,1,0000000000000010

li r2,2,0000000000000011

li r2,3,0000000000000100

li r2,4,0000000000000101

li r2,5,0000000000000110

li r2,6,0000000000000111

li r2,7,0000000000001000

li r3,0,0000000000000001

li r3,1,0000000000000010

li r3,2,0000000000000011

li r3,3,0000000000000100

li r3,4,0000000000000101

li r3,5,0000000000000110

li r3,6,0000000000000111

li r3,7,0000000000001000

IAL r4,r1,r2,r3

IAH r5,r1,r2,r3

ISL r6,r1,r2,r3

ISH r7,r1,r2,r3

LAL r8,r1,r2,r3

LAH r9,r1,r2,r3

LSL r10,r1,r2,r3

LSH r11,r1,r2,r3

A r12,r1,r2

AH r13,r1,r2

AHS r14,r1,r2

AND r15,r1,r2

BCW r16,r1,r2

CLZ r17,r1,r2

MAX r18,r1,r2

MIN r19,r1,r2

MSGN r20,r1,r2

MPYU r21,r1,r2

OR r22,r1,r2

POPCNTH r23,r1,r2

ROT r24,r1,r2

ROTW r25,r1,r2

SHLHI r26,r1,r2

SFH r27,r1,r2

SFW r28,r1,r2

SFHS r29,r1,r2

XOR r30,r1,r2

Opcode after transformed:

00000000000000000100001

00010000000000000100001

001000000000000001100001

0011000000000000010000001

0100000000000000010100001

0101000000000000011000001

0110000000000000011100001

01110000000000000100000001

00000000000000000100010

000100000000000001000010

001000000000000001100010

0011000000000000010000010

0100000000000000010100010

0101000000000000011000010

0110000000000000011100010

01110000000000000100000010

00000000000000000100011

000100000000000001000011

001000000000000001100011

0011000000000000010000011

0100000000000000010100011

0101000000000000011000011

0110000000000000011100011

0111000000000000100000011
1000000011000100000100100
1000100011000100000100101
1001000011000100000100110
1001100011000100000100111
1010000011000100000101000
1010100011000100000101001
1011000011000100000101010
1011100011000100000101011
1100000001000100000101100
1100000010000100000101101
1100000011000100000101110
1100000100000100000101111
1100000101000100000110000
1100000110000100000110001
1100000111000100000110010
1100001000000100000110011
1100001001000100000110100
1100001010000100000110101
1100001011000100000110110
1100001100000100000110111
1100001101000100000111000
1100001110000100000111001
1100001111000100000111010

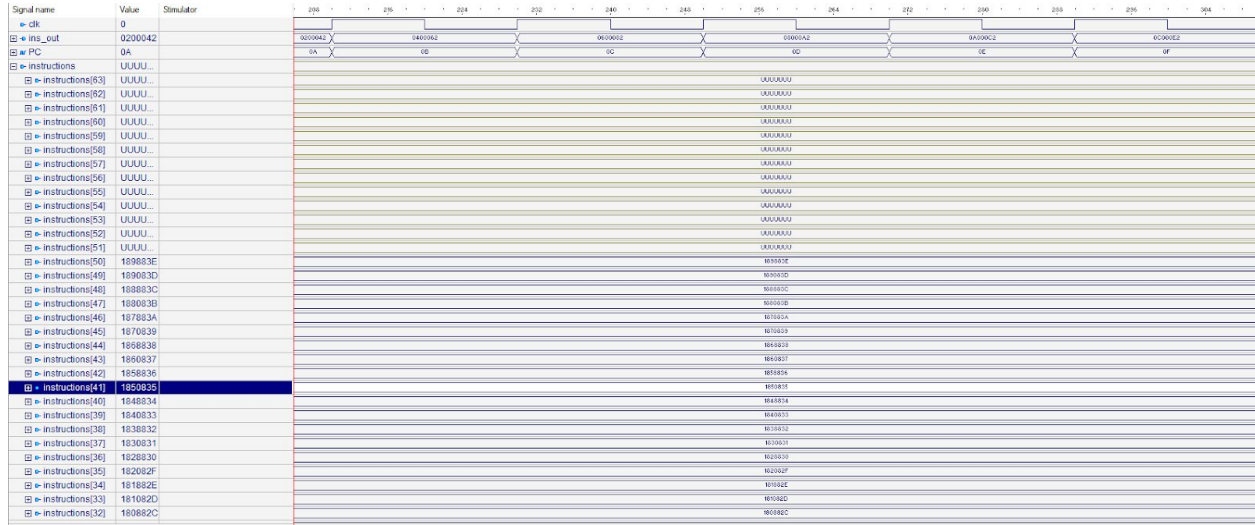
1100010000000100000111011

1100010001000100000111100

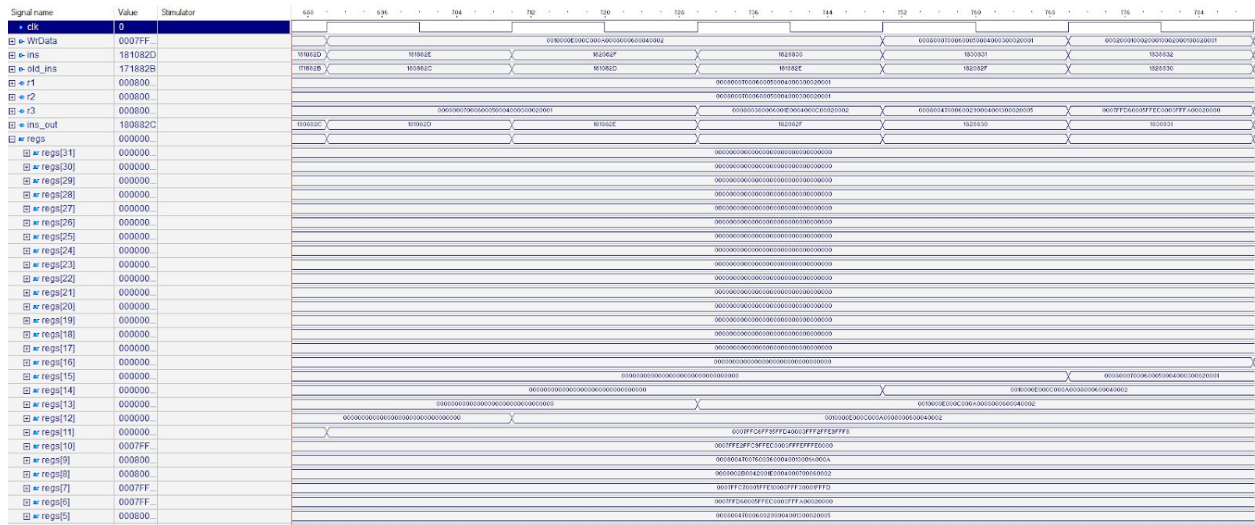
1100010010000100000111101

1100010011000100000111110

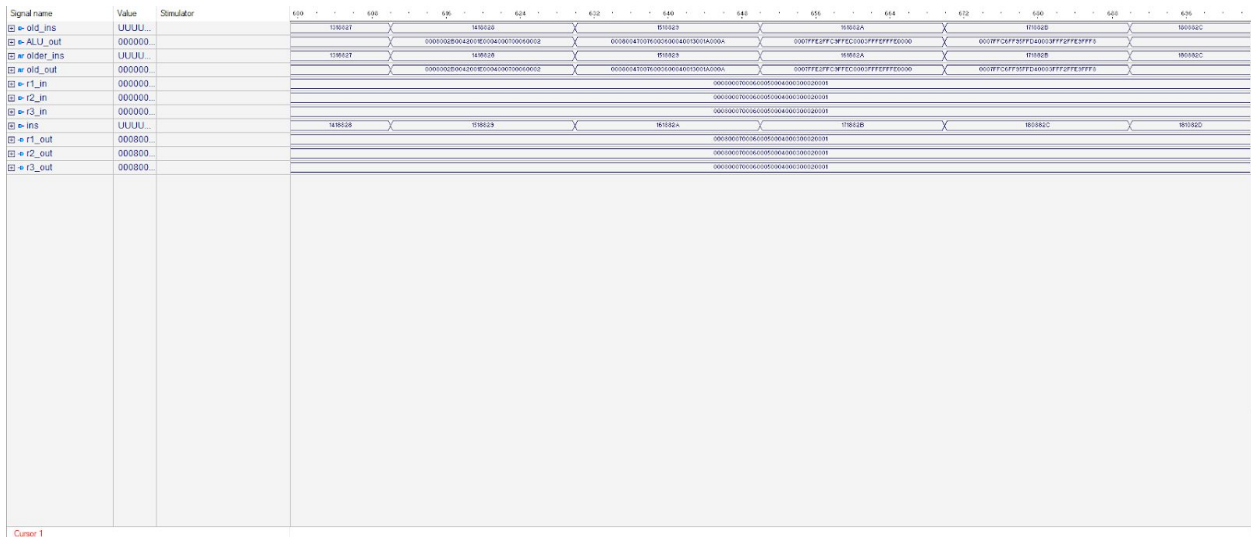
Waveform of Instruction Buffer during working(some instruction register is not used)



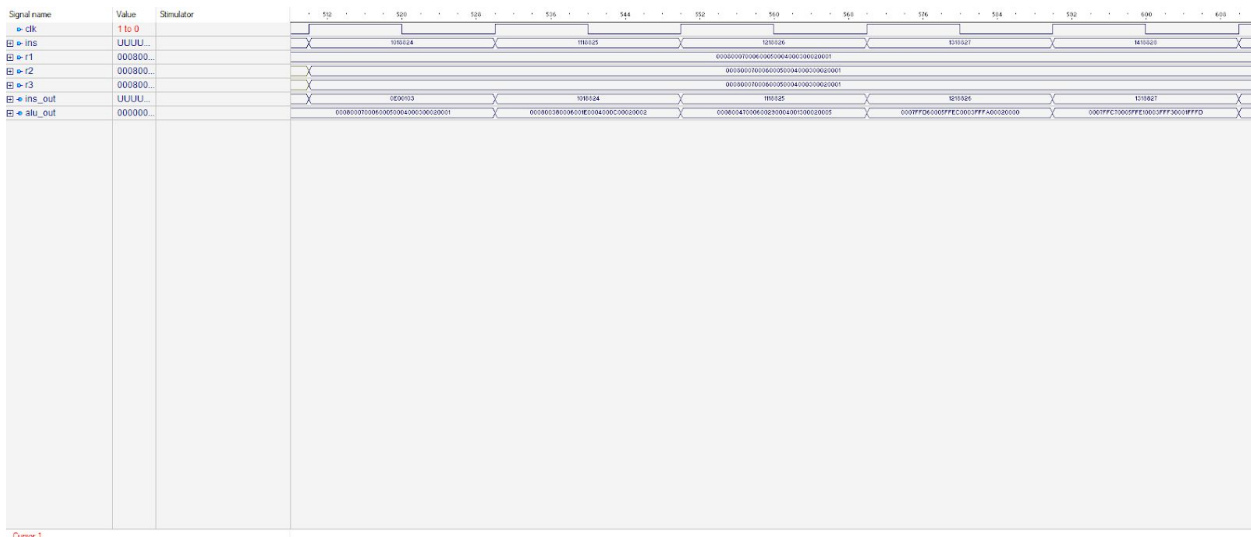
Waveform of Register File during working



Waveform of forwarding unit during working



Waveform of ALU during working



Waveform of final results in the registers

Signal name	Value	Simulator	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7	reg8	reg9	reg10	reg11	reg12	reg13	reg14	reg15	reg16	reg17	reg18	reg19	reg20	reg21	reg22	reg23	reg24	reg25	reg26	reg27	reg28	reg29	reg30	reg31	reg32	reg33	reg34	reg35	reg36	reg37	reg38	reg39	reg40	reg41	reg42	reg43	reg44	reg45	reg46	reg47	reg48	reg49	reg50	reg51	reg52	reg53	reg54	reg55	reg56	reg57	reg58	reg59	reg60	reg61	reg62	reg63	reg64	reg65	reg66	reg67	reg68	reg69	reg70	reg71	reg72	reg73	reg74	reg75	reg76	reg77	reg78	reg79	reg80	reg81	reg82	reg83	reg84	reg85	reg86	reg87	reg88	reg89	reg90	reg91	reg92	reg93	reg94	reg95	reg96	reg97	reg98	reg99	reg100	reg101	reg102	reg103	reg104	reg105	reg106	reg107	reg108	reg109	reg110	reg111	reg112	reg113	reg114	reg115	reg116	reg117	reg118	reg119	reg120	reg121	reg122	reg123	reg124	reg125	reg126	reg127	reg128	reg129	reg130	reg131	reg132	reg133	reg134	reg135	reg136	reg137	reg138	reg139	reg140	reg141	reg142	reg143	reg144	reg145	reg146	reg147	reg148	reg149	reg150	reg151	reg152	reg153	reg154	reg155	reg156	reg157	reg158	reg159	reg160	reg161	reg162	reg163	reg164	reg165	reg166	reg167	reg168	reg169	reg170	reg171	reg172	reg173	reg174	reg175	reg176	reg177	reg178	reg179	reg180	reg181	reg182	reg183	reg184	reg185	reg186	reg187	reg188	reg189	reg190	reg191	reg192	reg193	reg194	reg195	reg196	reg197	reg198	reg199	reg200	reg201	reg202	reg203	reg204	reg205	reg206	reg207	reg208	reg209	reg210	reg211	reg212	reg213	reg214	reg215	reg216	reg217	reg218	reg219	reg220	reg221	reg222	reg223	reg224	reg225	reg226	reg227	reg228	reg229	reg230	reg231	reg232	reg233	reg234	reg235	reg236	reg237	reg238	reg239	reg240	reg241	reg242	reg243	reg244	reg245	reg246	reg247	reg248	reg249	reg250	reg251	reg252	reg253	reg254	reg255	reg256	reg257	reg258	reg259	reg260	reg261	reg262	reg263	reg264	reg265	reg266	reg267	reg268	reg269	reg270	reg271	reg272	reg273	reg274	reg275	reg276	reg277	reg278	reg279	reg280	reg281	reg282	reg283	reg284	reg285	reg286	reg287	reg288	reg289	reg290	reg291	reg292	reg293	reg294	reg295	reg296	reg297	reg298	reg299	reg300	reg301	reg302	reg303	reg304	reg305	reg306	reg307	reg308	reg309	reg310	reg311	reg312	reg313	reg314	reg315	reg316	reg317	reg318	reg319	reg320	reg321	reg322	reg323	reg324	reg325	reg326	reg327	reg328	reg329	reg330	reg331	reg332	reg333	reg334	reg335	reg336	reg337	reg338	reg339	reg340	reg341	reg342	reg343	reg344	reg345	reg346	reg347	reg348	reg349	reg350	reg351	reg352	reg353	reg354	reg355	reg356	reg357	reg358	reg359	reg360	reg361	reg362	reg363	reg364	reg365	reg366	reg367	reg368	reg369	reg370	reg371	reg372	reg373	reg374	reg375	reg376	reg377	reg378	reg379	reg380	reg381	reg382	reg383	reg384	reg385	reg386	reg387	reg388	reg389	reg390	reg391	reg392	reg393	reg394	reg395	reg396	reg397	reg398	reg399	reg400	reg401	reg402	reg403	reg404	reg405	reg406	reg407	reg408	reg409	reg410	reg411	reg412	reg413	reg414	reg415	reg416	reg417	reg418	reg419	reg420	reg421	reg422	reg423	reg424	reg425	reg426	reg427	reg428	reg429	reg430	reg431	reg432	reg433	reg434	reg435	reg436	reg437	reg438	reg439	reg440	reg441	reg442	reg443	reg444	reg445	reg446	reg447	reg448	reg449	reg450	reg451	reg452	reg453	reg454	reg455	reg456	reg457	reg458	reg459	reg460	reg461	reg462	reg463	reg464	reg465	reg466	reg467	reg468	reg469	reg470	reg471	reg472	reg473	reg474	reg475	reg476	reg477	reg478	reg479	reg480	reg481	reg482	reg483	reg484	reg485	reg486	reg487	reg488	reg489	reg490	reg491	reg492	reg493	reg494	reg495	reg496	reg497	reg498	reg499	reg500	reg501	reg502	reg503	reg504	reg505	reg506	reg507	reg508	reg509	reg510	reg511	reg512	reg513	reg514	reg515	reg516	reg517	reg518	reg519	reg520	reg521	reg522	reg523	reg524	reg525	reg526	reg527	reg528	reg529	reg530	reg531	reg532	reg533	reg534	reg535	reg536	reg537	reg538	reg539	reg540	reg541	reg542	reg543	reg544	reg545	reg546	reg547	reg548	reg549	reg550	reg551	reg552	reg553	reg554	reg555	reg556	reg557	reg558	reg559	reg560	reg561	reg562	reg563	reg564	reg565	reg566	reg567	reg568	reg569	reg570	reg571	reg572	reg573	reg574	reg575	reg576	reg577	reg578	reg579	reg580	reg581	reg582	reg583	reg584	reg585	reg586	reg587	reg588	reg589	reg590	reg591	reg592	reg593	reg594	reg595	reg596	reg597	reg598	reg599	reg600	reg601	reg602	reg603	reg604	reg605	reg606	reg607	reg608	reg609	reg610	reg611	reg612	reg613	reg614	reg615	reg616	reg617	reg618	reg619	reg620	reg621	reg622	reg623	reg624	reg625	reg626	reg627	reg628	reg629	reg630	reg631	reg632	reg633	reg634	reg635	reg636	reg637	reg638	reg639	reg640	reg641	reg642	reg643	reg644	reg645	reg646	reg647	reg648	reg649	reg650	reg651	reg652	reg653	reg654	reg655	reg656	reg657	reg658	reg659	reg660	reg661	reg662	reg663	reg664	reg665	reg666	reg667	reg668	reg669	reg670	reg671	reg672	reg673	reg674	reg675	reg676	reg677	reg678	reg679	reg680	reg681	reg682	reg683	reg684	reg685	reg686	reg687	reg688	reg689	reg690	reg691	reg692	reg693	reg694	reg695	reg696	reg697	reg698	reg699	reg700	reg701	reg702	reg703	reg704	reg705	reg706	reg707	reg708	reg709	reg710	reg711	reg712	reg713	reg714	reg715	reg716	reg717	reg718	reg719	reg720	reg721	reg722	reg723	reg724	reg725	reg726	reg727	reg728	reg729	reg730	reg731	reg732	reg733	reg734	reg735	reg736	reg737	reg738	reg739	reg740	reg741	reg742	reg743	reg744	reg745	reg746	reg747	reg748	reg749	reg750	reg751	reg752	reg753	reg754	reg755	reg756	reg757	reg758	reg759	reg760	reg761	reg762	reg763	reg764	reg765	reg766	reg767	reg768	reg769	reg770	reg771	reg772	reg773	reg774	reg775	reg776	reg777	reg778	reg779	reg780	reg781	reg782	reg783	reg784	reg785	reg786	reg787	reg788	reg789	reg790	reg791	reg792	reg793	reg794	reg795	reg796	reg797	reg798	reg799	reg800	reg801	reg802	reg803	reg804	reg805	reg806	reg807	reg808	reg809	reg810	reg811	reg812	reg813	reg814	reg815	reg816	reg817	reg818	reg819	reg820	reg821	reg822	reg823	reg824	reg825	reg826	reg827	reg828	reg829	reg830	reg831	reg832	reg833	reg834	reg835	reg836	reg837	reg838	reg839	reg840	reg841	reg842	reg843	reg844	reg845	reg846	reg847	reg848	reg849	reg850	reg851	reg852	reg853	reg854	reg855	reg856	reg857	reg858	reg859	reg860	reg861	reg862	reg863	reg864	reg865	reg866	reg867	reg868	reg869	reg870	reg871	reg872	reg873	reg874	reg875	reg876	reg877	reg878	reg879	reg880	reg881	reg882	reg883	reg884	reg885	reg886	reg887	reg888	reg889	reg890	reg891	reg892	reg893	reg894	reg895	reg896	reg897	reg898	reg899	reg900	reg901	reg902	reg903	reg904	reg905	reg906	reg907	reg908	reg909	reg910	reg911	reg912	reg913	reg914	reg915	reg916	reg917	reg918	reg919	reg920	reg921	reg922	reg923	reg924	reg925	reg926	reg927	reg928	reg929	reg930	reg931	reg932	reg933	reg934	reg935	reg936	reg937	reg938	reg939	reg940	reg941	reg942	reg943	reg944	reg945	reg946	reg947	reg948	reg949	reg950	reg951	reg952	reg953	reg954	reg955	reg956	reg957	reg958	reg959	reg960	reg961	reg962	reg963	reg964	reg965	reg966	reg967	reg968	reg969	reg970	reg971	reg972	reg973	reg974	reg975	reg976	reg977	reg978	reg979	reg980	reg981	reg982	reg983	reg984	reg985	reg986	reg987	reg988	reg989	reg990	reg991	reg992	reg993	reg994	reg995	reg996	reg997	reg998	reg999	reg1000	reg1001	reg1002	reg1003	reg1004	reg1005	reg1006	reg1007	reg1008	reg1009	reg1010	reg1011	reg1012	reg1013	reg1014	reg1015	reg1016	reg1017	reg1018	reg1019	reg1020	reg1021	reg1022	reg1023	reg1024	reg1025	reg1026	reg1027	reg1028	reg1029	reg1030	reg1031	reg1032	reg1033	reg1034	reg1035	reg1036	reg1037	reg1038	reg1039	reg1040	reg1041	reg1042	reg1043	reg1044	reg1045	reg1046	reg1047	reg1048	reg1049	reg1050	reg1051	reg1052	reg1053	reg1054	reg1055	reg1056	reg1057	reg1058	reg1059	reg1060	reg1061	reg1062	reg1063	reg1064	reg1065	reg1066	reg1067	reg1068	reg1069	reg1070	reg1071	reg1072	reg1073	reg1074	reg1075	reg1076	reg1077	reg1078	reg1079	reg1080	reg1081	reg1082	reg1083	reg1084	reg1085	reg1086	reg1087	reg1088	reg1089	reg1090	reg1091	reg1092	reg1093	reg1094	reg1095	reg1096	reg1097	reg1098	reg1099	reg1100	reg1101	reg1102	reg1103	reg1104	reg1105	reg1106	reg1107	reg1108	reg1109	reg1110	reg1111	reg1112	reg1113	reg1114	reg1115	reg1116	reg1117	reg1118	reg1119	reg1120	reg1121	reg1122	reg1123	reg1124	reg1125	reg1126	reg1127	reg1128	reg1129	reg1130	reg1131	reg1132	reg1133	reg1134	reg1135	reg1136	reg1137	reg1138	reg1139	reg1140	reg1141	reg1142	reg1143	reg1144	reg1145	reg1146	reg1147	reg1148	reg1149	reg1150	reg1151	reg1152	reg1153	reg1154	reg1155	reg1156	reg1157	reg1158	reg1159	reg1160	reg1161	reg1162	reg1163	reg1164	reg1165	reg1166	reg1167	reg1168	reg1169	reg1170	reg1171	reg1172	reg1173	reg1174	reg1175	reg1176	reg1177	reg1178	reg1179	reg1180	reg1181	reg1182	reg1183	reg1184	reg1185	reg1186	reg1187	reg1188	reg1189	reg1190	reg1191	reg1192	reg1193	reg1194	reg1195	reg1196	reg1197	reg1198	reg1199	reg1200	reg1201	reg1202	reg1203	reg1204	reg1205	reg1206	reg1207	reg1208	reg1209	reg1210	reg1211	reg1212	reg1213	reg1214	reg1215	reg1216	reg1217	reg1218	reg1219	reg1220	reg1221	reg1222	reg1223	reg1224	reg1225	reg1226	reg1227	reg1228	reg1229	reg1230	reg1231	reg1232	reg1233	reg1234	reg1235	reg1236	reg1237	reg1238	reg1239	reg1240	reg1241	reg1242	reg1243	reg1244	reg1245	reg1246	reg1247	reg1248	reg1249	reg1250	reg1251	reg1252	reg1253	reg1254	reg1255	reg1256	reg1257	reg1258	reg1259	reg1260	reg1261	reg1262	reg1263	reg1264	reg1265	reg1266	reg1267	reg1268	reg1269	reg1270	reg1271	reg1272	reg1273	reg1274	reg1275	reg1276	reg1277	reg1278	reg1279	reg1280	reg1281	reg1282	reg1283	reg1284	reg1285	reg1286	reg1287	reg1288	reg1289	reg1290	reg1291	reg1292	reg1293	reg1294	reg1295	reg1296	reg1297	reg1298	reg1299	reg1300	reg1301	reg1302	reg1303	reg1304	reg1305	reg1306	reg1307	reg1308	reg1309	reg1310	reg1311	reg1312	reg1313	reg1314	reg1315	reg1316	reg1317	reg1318	reg1319	reg1320	reg1321	reg1322	reg1323	reg1324	reg1325	reg1326	reg1327	reg1328	reg1329	reg1330	reg1331	reg1332	reg1333	reg1334	reg1335	reg1336	reg1337	reg1338	reg1339	reg1340	reg1341	reg1342	reg1343	reg1344	reg1345	reg1346
-------------	-------	-----------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------

Final result in register:

00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
0800380018000A00040001800080002
0E001000280030006000800080010000
80040003800300028002000180010000
00010003000200020001000200010001
00080007000600050004000300020001
000000310000001900000000900000001
00080007000600050004000300020001
00080007000600050004000300020001
00080007000600050004000300020001
00080007000600050004000300020001
0000000C0000000D0000000D0000000E
00020001000200010002000100020001
00080007000600050004000300020001
0010000E000C000A0008000600040002
0010000E000C000A0008000600040002
0010000E000C000A0008000600040002
0007FFC6FF35FFD40003FFF2FFE3FFF8
0007FFE2FFC3FFEC0003FFFEFFFE0000
000800470076003600040013001A000A
0008002B0042001E0004000700060002
0007FFC70005FFE10003FFF30001FFFD
0007FFD60005FFEC0003FFFA00020000
00080047000600290004001300020005
000800380006001E0004000C00020002
00080007000600050004000300020001
00080007000600050004000300020001
00080007000600050004000300020001
00000000000000000000000000000000

Expected Results:

000800380006001E0004000C00020002

00080047000600290004001300020005

0007FFD60005FFEC0003FFFA00020000

0007FFC70005FFE10003FFF30001FFFD

0008002B0042001E0004000700060002

000800470076003600040013001A000A

0007FFE2FFC9FFEC0003FFFEFFFE0000

0007FFC6FF95FFD40003FFF2FFE9FFF8

0010000E000C000A0008000600040002

0010000E000C000A0008000600040002

0010000E000C000A0008000600040002

00080007000600050004000300020001

00020001000200010002000100020001

0000000C0000000D0000000D0000000E

00080007000600050004000300020001

00080007000600050004000300020001

00080007000600050004000300020001

00000031000000190000000900000001

00080007000600050004000300020001

00010003000200020001000200010001

80040003800300028002000180010000

0E001000280030006000800080010000

08000380800000A00040001800080002

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

```

1
2  library ieee;
3  use ieee.std_logic_1164.all;
4  use ieee.numeric_std.all;
5  --use ieee.std_logic_arith.all;
6
7  entity alu is
8      port(
9          r1: in std_logic_vector (127 downto 0);
10         r2: in std_logic_vector (127 downto 0);
11         r3: in std_logic_vector (127 downto 0);
12         ins: in std_logic_vector (24 downto 0);
13         clk : in std_logic;
14         ins_out: out std_logic_vector (24 downto 0);
15         alu_out: out std_logic_vector (127 downto 0)
16     );
17 end alu;
18
19 architecture behavioral of alu is
20 begin
21     process(r1, r2, r3, ins, clk)
22         variable temp1, temp2: std_logic_vector (31 downto 0);
23         variable temp3, temp4: std_logic_vector (63 downto 0);
24         variable zero_count : integer;
25         variable one_count : integer;
26         variable bits_of_shift : integer;
27     begin
28         if rising_edge(clk) then
29
30             ins_out<=ins;
31
32             -----4.1 load imm instruction-----
33             if ins(24) = '0' then
34                 if ins(23 downto 21) = "000" then
35                     alu_out <= r1;
36                     alu_out(15 downto 0) <= ins(20 downto 5);
37                 elsif ins(23 downto 21) = "001" then
38                     alu_out <= r1;
39                     alu_out(31 downto 16) <= ins(20 downto 5);
40                 elsif ins(23 downto 21) = "010" then
41                     alu_out <= r1;
42                     alu_out(47 downto 32) <= ins(20 downto 5);
43                 elsif ins(23 downto 21) = "011" then
44                     alu_out <= r1;
45                     alu_out(63 downto 48) <= ins(20 downto 5);
46                 elsif ins(23 downto 21) = "100" then
47                     alu_out <= r1;
48                     alu_out(79 downto 64) <= ins(20 downto 5);
49                 elsif ins(23 downto 21) = "101" then
50                     alu_out <= r1;
51                     alu_out(95 downto 80) <= ins(20 downto 5);
52                 elsif ins(23 downto 21) = "110" then
53                     alu_out <= r1;
54                     alu_out(111 downto 96) <= ins(20 downto 5);
55                 elsif ins(23 downto 21) = "111" then
56                     alu_out <= r1;
57                     alu_out(127 downto 112) <= ins(20 downto 5);
58                 end if;
59
60             -----4.2 R4 instruction-----
61             -----saturation??-----
62             elsif ins(24 downto 23) = "10" then
63
64                 case ins(22 downto 20) is
65                     -----Signed Integer Multiply-Add Low with
66                     Saturation-----

```



```

66         when "000" =>
67             temp1:=std_logic_vector (signed(r2(111 downto 96))* signed(r3(111
        downto 96)));
68             temp2:=std_logic_vector (signed(temp1) + signed(r1(127 downto 96)));
69             if temp2(31)='1' and temp1(31)='0' and r1(127)='0' then
70                 alu_out(127 downto 96)<=(others=>'1');
71                 alu_out(127)<='0';
72             elsif temp2(31)='0' and temp1(31)='1' and r1(127)='1' then
73                 alu_out(127 downto 96)<=(others=>'0');
74                 alu_out(127)<='1';
75             else
76                 alu_out(127 downto 96)<=temp2;
77             end if;
78
79             temp1:=std_logic_vector (signed(r2(79 downto 64))* signed(r3(79
        downto 64)));
80             temp2:=std_logic_vector (signed(temp1) + signed(r1(95 downto 64)));
81             if temp2(31)='1' and temp1(31)='0' and r1(95)='0' then
82                 alu_out(95 downto 64)<=(others=>'1');
83                 alu_out(95)<='0';
84             elsif temp2(31)='0' and temp1(31)='1' and r1(95)='1' then
85                 alu_out(95 downto 64)<=(others=>'0');
86                 alu_out(95)<='1';
87             else
88                 alu_out(95 downto 64)<=temp2;
89             end if;
90
91             temp1:=std_logic_vector (signed(r2(47 downto 32))* signed(r3(47
        downto 32)));
92             temp2:=std_logic_vector (signed(temp1) + signed(r1(63 downto 32)));
93             if temp2(31)='1' and temp1(31)='0' and r1(63)='0' then
94                 alu_out(63 downto 32)<=(others=>'1');
95                 alu_out(63)<='0';
96             elsif temp2(31)='0' and temp1(31)='1' and r1(63)='1' then
97                 alu_out(63 downto 32)<=(others=>'0');
98                 alu_out(63)<='1';
99             else
100                 alu_out(63 downto 32)<=temp2;
101             end if;
102
103             temp1:=std_logic_vector (signed(r2(15 downto 0))* signed(r3(15 downto
        0)));
104             temp2:=std_logic_vector (signed(temp1) + signed(r1(31 downto 0)));
105             if temp2(31)='1' and temp1(31)='0' and r1(31)='0' then
106                 alu_out(31 downto 0)<=(others=>'1');
107                 alu_out(31)<='0';
108             elsif temp2(31)='0' and temp1(31)='1' and r1(31)='1' then
109                 alu_out(31 downto 0)<=(others=>'0');
110                 alu_out(31)<='1';
111             else
112                 alu_out(31 downto 0)<=temp2;
113             end if;
114             -----Signed Integer Multiply-Add High with
        Saturation-----
115         when "001" =>
116             temp1:=std_logic_vector (signed(r2(127 downto 112))* signed(r3(127
        downto 112)));
117             temp2:=std_logic_vector (signed(temp1) + signed(r1(127 downto 96)));
118             if temp2(31)='1' and temp1(31)='0' and r1(127)='0' then
119                 alu_out(127 downto 96)<=(others=>'1');
120                 alu_out(127)<='0';
121             elsif temp2(31)='0' and temp1(31)='1' and r1(127)='1' then
122                 alu_out(127 downto 96)<=(others=>'0');
123                 alu_out(127)<='1';
124             else
125                 alu_out(127 downto 96)<=temp2;

```

```

126         end if;
127
128         temp1:=std_logic_vector (signed(r2(95 downto 80))* signed(r3(95
downto 80)));
129
130         temp2:=std_logic_vector (signed(temp1) + signed(r1(95 downto 64)));
131         if temp2(31)='1' and temp1(31)='0' and r1(95)='0' then
132             alu_out(95 downto 64)<=(others=>'1');
133             alu_out(95)<='0';
134         elsif temp2(31)='0' and temp1(31)='1' and r1(95)='1' then
135             alu_out(95 downto 64)<=(others=>'0');
136             alu_out(95)<='1';
137         else
138             alu_out(95 downto 64)<=temp2;
139         end if;
140
141         temp1:=std_logic_vector (signed(r2(63 downto 48))* signed(r3(63
downto 48)));
142
143         temp2:=std_logic_vector (signed(temp1) + signed(r1(63 downto 32)));
144         if temp2(31)='1' and temp1(31)='0' and r1(63)='0' then
145             alu_out(63 downto 32)<=(others=>'1');
146             alu_out(63)<='0';
147         elsif temp2(31)='0' and temp1(31)='1' and r1(63)='1' then
148             alu_out(63 downto 32)<=(others=>'0');
149             alu_out(63)<='1';
150         else
151             alu_out(63 downto 32)<=temp2;
152         end if;
153
154         temp1:=std_logic_vector (signed(r2(31 downto 16))* signed(r3(31
downto 16)));
155
156         temp2:=std_logic_vector (signed(temp1) + signed(r1(31 downto 0)));
157         if temp2(31)='1' and temp1(31)='0' and r1(31)='0' then
158             alu_out(31 downto 0)<=(others=>'1');
159             alu_out(31)<='0';
160         elsif temp2(31)='0' and temp1(31)='1' and r1(31)='1' then
161             alu_out(31 downto 0)<=(others=>'0');
162             alu_out(31)<='1';
163         else
164             alu_out(31 downto 0)<=temp2;
165         end if;
166
167         -----Signed Integer Multiply-Subtract Low with
Saturation-----
168         when "010" =>
169             temp1:=std_logic_vector (signed(r2(111 downto 96))*signed(r3(111
downto 96)));
170
171             temp2:=std_logic_vector (signed(r1(127 downto 96))-signed(temp1));
172             if temp2(31)='1' and temp1(31)='1' and r1(127)='0' then
173                 alu_out(127 downto 96)<=(others=>'1');
174                 alu_out(127)<='0';
175             elsif temp2(31)='0' and temp1(31)='0' and r1(127)='1' then
176                 alu_out(127 downto 96)<=(others=>'0');
177                 alu_out(127)<='1';
178             else
179                 alu_out(127 downto 96)<=temp2;
180             end if;
181
182             temp1:=std_logic_vector (signed(r2(79 downto 64))* signed(r3(79
downto 64)));
183
184             temp2:=std_logic_vector (signed(r1(95 downto 64)) - signed(temp1));
185             if temp2(31)='1' and temp1(31)='1' and r1(95)='0' then
186                 alu_out(95 downto 64)<=(others=>'1');
187                 alu_out(95)<='0';
188             elsif temp2(31)='0' and temp1(31)='0' and r1(95)='1' then
189                 alu_out(95 downto 64)<=(others=>'0');
190                 alu_out(95)<='1';
191             else

```

```

186         alu_out(95 downto 64) <= temp2;
187     end if;
188
189     temp1 := std_logic_vector(signed(r2(47 downto 32)) * signed(r3(47
downto 32)));
190     temp2 := std_logic_vector(signed(r1(63 downto 32)) - signed(temp1));
191     if temp2(31) = '1' and temp1(31) = '1' and r1(63) = '0' then
192         alu_out(63 downto 32) <= (others => '1');
193         alu_out(63) <= '0';
194     elsif temp2(31) = '0' and temp1(31) = '0' and r1(63) = '1' then
195         alu_out(63 downto 32) <= (others => '0');
196         alu_out(63) <= '1';
197     else
198         alu_out(63 downto 32) <= temp2;
199     end if;
200
201     temp1 := std_logic_vector(signed(r2(15 downto 0)) * signed(r3(15 downto
0)));
202     temp2 := std_logic_vector(signed(r1(31 downto 0)) - signed(temp1));
203     if temp2(31) = '1' and temp1(31) = '1' and r1(31) = '0' then
204         alu_out(31 downto 0) <= (others => '1');
205         alu_out(31) <= '0';
206     elsif temp2(31) = '0' and temp1(31) = '0' and r1(31) = '1' then
207         alu_out(31 downto 0) <= (others => '0');
208         alu_out(31) <= '1';
209     else
210         alu_out(31 downto 0) <= temp2;
211     end if;
212
213     -----Signed Integer Multiply-Subtract High with
Saturation-----
214     when "011" =>
215         temp1 := std_logic_vector(signed(r2(127 downto 112)) * signed(r3(127
downto 112)));
216         temp2 := std_logic_vector(signed(r1(127 downto 96)) - signed(temp1));
217         if temp2(31) = '1' and temp1(31) = '1' and r1(127) = '0' then
218             alu_out(127 downto 96) <= (others => '1');
219             alu_out(127) <= '0';
220         elsif temp2(31) = '0' and temp1(31) = '0' and r1(127) = '1' then
221             alu_out(127 downto 96) <= (others => '0');
222             alu_out(127) <= '1';
223         else
224             alu_out(127 downto 96) <= temp2;
225         end if;
226
227         temp1 := std_logic_vector(signed(r2(95 downto 80)) * signed(r3(95
downto 80)));
228         temp2 := std_logic_vector(signed(r1(95 downto 64)) - signed(temp1));
229         if temp2(31) = '1' and temp1(31) = '1' and r1(95) = '0' then
230             alu_out(95 downto 64) <= (others => '1');
231             alu_out(95) <= '0';
232         elsif temp2(31) = '0' and temp1(31) = '0' and r1(95) = '1' then
233             alu_out(95 downto 64) <= (others => '0');
234             alu_out(95) <= '1';
235         else
236             alu_out(95 downto 64) <= temp2;
237         end if;
238
239         temp1 := std_logic_vector(signed(r2(63 downto 48)) * signed(r3(63
downto 48)));
240         temp2 := std_logic_vector(signed(r1(63 downto 32)) - signed(temp1));
241         if temp2(31) = '1' and temp1(31) = '1' and r1(63) = '0' then
242             alu_out(63 downto 32) <= (others => '1');
243             alu_out(63) <= '0';
244         elsif temp2(31) = '0' and temp1(31) = '0' and r1(63) = '1' then
245             alu_out(63 downto 32) <= (others => '0');

```

```

246         alu_out(63)<='1';
247     else
248         alu_out(63 downto 32)<=temp2;
249     end if;
250
251     temp1:=std_logic_vector(signed(r2(31 downto 16))* signed(r3(31
downto 16)));
252
253     temp2:=std_logic_vector(signed(r1(31 downto 0)) - signed(temp1));
254     if temp2(31)='1' and temp1(31)='1' and r1(31)='0' then
255         alu_out(31 downto 0)<=(others=>'1');
256         alu_out(31)<='0';
257     elsif temp2(31)='0' and temp1(31)='0' and r1(31)='1' then
258         alu_out(31 downto 0)<=(others=>'0');
259         alu_out(31)<='1';
260     else
261         alu_out(31 downto 0)<=temp2;
262     end if;
263
264     -----Signed Long Integer Multiply-Add Low with
Saturation-----
265     when "100" =>
266         temp3:=std_logic_vector(signed(r2(95 downto 64))* signed(r3(95
downto 64)));
267
268         temp4:=std_logic_vector(signed(temp3) + signed(r1(127 downto 64)));
269         if temp4(63)='1' and temp3(63)='0' and r1(127)='0' then
270             alu_out(127 downto 64)<=(others=>'1');
271             alu_out(127)<='0';
272         elsif temp4(63)='0' and temp3(63)='1' and r1(127)='1' then
273             alu_out(127 downto 64)<=(others=>'0');
274             alu_out(127)<='1';
275         else
276             alu_out(127 downto 64)<=temp4;
277         end if;
278
279         temp3:=std_logic_vector(signed(r2(31 downto 0))* signed(r3(31 downto
0)));
280
281         temp4:=std_logic_vector(signed(temp3) + signed(r1(63 downto 0)));
282         if temp4(63)='1' and temp3(63)='0' and r1(63)='0' then
283             alu_out(63 downto 0)<=(others=>'1');
284             alu_out(63)<='0';
285         elsif temp4(63)='0' and temp3(63)='1' and r1(63)='1' then
286             alu_out(63 downto 0)<=(others=>'0');
287             alu_out(63)<='1';
288         else
289             alu_out(63 downto 0)<=temp4;
290         end if;
291
292     -----Signed Long Integer Multiply-Add High with
Saturation-----
293     when "101" =>
294         temp3:=std_logic_vector(signed(r2(127 downto 96))* signed(r3(127
downto 96)));
295
296         temp4:=std_logic_vector(signed(temp3) + signed(r1(127 downto 64)));
297         if temp4(63)='1' and temp3(63)='0' and r1(127)='0' then
298             alu_out(127 downto 64)<=(others=>'1');
299             alu_out(127)<='0';
300         elsif temp4(63)='0' and temp3(63)='1' and r1(127)='1' then
301             alu_out(127 downto 64)<=(others=>'0');
302             alu_out(127)<='1';
303         else
304             alu_out(127 downto 64)<=temp4;
305         end if;
306
307         temp3:=std_logic_vector(signed(r2(63 downto 32))* signed(r3(63
downto 32)));
308
309         temp4:=std_logic_vector(signed(temp3) + signed(r1(63 downto 0)));
310         if temp4(63)='1' and temp3(63)='0' and r1(63)='0' then

```

```

305         alu_out(63 downto 0) <= (others => '1');
306         alu_out(63) <= '0';
307     elsif temp4(63) = '0' and temp3(63) = '1' and r1(63) = '1' then
308         alu_out(63 downto 0) <= (others => '0');
309         alu_out(63) <= '1';
310     else
311         alu_out(63 downto 0) <= temp4;
312     end if;
313     -----Signed Long Integer Multiply-Subtract Low with
Saturation-----
314     when "110" =>
315         temp3 := std_logic_vector(signed(r2(95 downto 64)) * signed(r3(95
downto 64)));
316         temp4 := std_logic_vector(signed(r1(127 downto 64)) - signed(temp3));
317         if temp4(63) = '1' and temp3(63) = '1' and r1(127) = '0' then
318             alu_out(127 downto 64) <= (others => '1');
319             alu_out(127) <= '0';
320         elsif temp4(63) = '0' and temp3(63) = '0' and r1(127) = '1' then
321             alu_out(127 downto 64) <= (others => '0');
322             alu_out(127) <= '1';
323         else
324             alu_out(127 downto 64) <= temp4;
325         end if;
326
327         temp3 := std_logic_vector(signed(r2(31 downto 0)) * signed(r3(31 downto
0)));
328         temp4 := std_logic_vector(signed(r1(63 downto 0)) - signed(temp3));
329         if temp4(63) = '1' and temp3(63) = '1' and r1(63) = '0' then
330             alu_out(63 downto 0) <= (others => '1');
331             alu_out(63) <= '0';
332         elsif temp4(63) = '0' and temp3(63) = '0' and r1(63) = '1' then
333             alu_out(63 downto 0) <= (others => '0');
334             alu_out(63) <= '1';
335         else
336             alu_out(63 downto 0) <= temp4;
337         end if;
338     -----Signed Long Integer Multiply-Subtract High with
Saturation-----
339     when "111" =>
340         temp3 := std_logic_vector(signed(r2(127 downto 96)) * signed(r3(127
downto 96)));
341         temp4 := std_logic_vector(signed(r1(127 downto 64)) - signed(temp3));
342         if temp4(63) = '1' and temp3(63) = '1' and r1(127) = '0' then
343             alu_out(127 downto 64) <= (others => '1');
344             alu_out(127) <= '0';
345         elsif temp4(63) = '0' and temp3(63) = '0' and r1(127) = '1' then
346             alu_out(127 downto 64) <= (others => '0');
347             alu_out(127) <= '1';
348         else
349             alu_out(127 downto 64) <= temp4;
350         end if;
351
352         temp3 := std_logic_vector(signed(r2(63 downto 32)) * signed(r3(63
downto 32)));
353         temp4 := std_logic_vector(signed(r1(63 downto 0)) - signed(temp3));
354         if temp4(63) = '1' and temp3(63) = '0' and r1(63) = '1' then
355             alu_out(63 downto 0) <= (others => '1');
356             alu_out(63) <= '0';
357         elsif temp4(63) = '0' and temp3(63) = '1' and r1(63) = '0' then
358             alu_out(63 downto 0) <= (others => '0');
359             alu_out(63) <= '1';
360         else
361             alu_out(63 downto 0) <= temp4;
362         end if;
363     when others => null;
364

```

```

365         end case;
366
367         -----4.3 R3 instruction-----
368         elsif ins(24 downto 23) = "11" then
369             case ins(19 downto 15) is
370                 -----NOP-----
371                 when "00000" => null;
372                 -----A: add word-----
373                 when "00001" =>
374                     alu_out(127 downto 96) <= std_logic_vector(unsigned(r2(127 downto 96
375 ))+unsigned(r1(127 downto 96)));
376                     alu_out(95 downto 64) <= std_logic_vector(unsigned(r2(95 downto 64))+
377 unsigned(r1(95 downto 64)));
378                     alu_out(63 downto 32) <= std_logic_vector(unsigned(r2(63 downto 32))+
379 unsigned(r1(63 downto 32)));
380                     alu_out(31 downto 0) <= std_logic_vector(unsigned(r2(31 downto 0))+
381 unsigned(r1(31 downto 0)));
382                 -----AH: add halfword-----
383                 when "00010" =>
384                     alu_out(127 downto 112) <= std_logic_vector(unsigned(r2(127 downto 112
385 ))+unsigned(r1(127 downto 112)));
386                     alu_out(111 downto 96) <= std_logic_vector(unsigned(r2(111 downto 96
387 ))+unsigned(r1(111 downto 96)));
388                     alu_out(95 downto 80) <= std_logic_vector(unsigned(r2(95 downto 80))+
389 unsigned(r1(95 downto 80)));
390                     alu_out(79 downto 64) <= std_logic_vector(unsigned(r2(79 downto 64))+
391 unsigned(r1(79 downto 64)));
392                     alu_out(63 downto 48) <= std_logic_vector(unsigned(r2(63 downto 48))+
393 unsigned(r1(63 downto 48)));
394                     alu_out(47 downto 32) <= std_logic_vector(unsigned(r2(47 downto 32))+
395 unsigned(r1(47 downto 32)));
396                     alu_out(31 downto 16) <= std_logic_vector(unsigned(r2(31 downto 16))+
397 unsigned(r1(31 downto 16)));
398                     alu_out(15 downto 0) <= std_logic_vector(unsigned(r2(15 downto 0))+
399 unsigned(r1(15 downto 0)));
400                 -----AHS: add halfword saturated-----
401                 when "00011" =>
402                     alu_out(127 downto 112) <= std_logic_vector(signed(r2(127 downto 112)) +
403 signed(r1(127 downto 112)));
404                     if alu_out(127)='1' and r2(127) = '0' and r1(127) = '0' then
405                         alu_out(127 downto 112) <= (others=>'1');
406                         alu_out(127) <= '0';
407                     elsif alu_out(127) = '0' and r2(127) = '1' and r1(127)='1' then
408                         alu_out(127 downto 112) <= (others=>'0');
409                         alu_out(127) <= '1';
410                     end if;
411                     alu_out(111 downto 96) <= std_logic_vector(signed(r2(111 downto 96))+
412 signed(r1(111 downto 96)));
413                     if alu_out(111)='1' and r2(111) = '0' and r1(111) = '0' then
414                         alu_out(111 downto 96) <= (others=>'1');
415                         alu_out(111) <= '0';
416                     elsif alu_out(111) = '0' and r2(111) = '1' and r1(111)='1' then
417                         alu_out(111 downto 96) <= (others=>'0');
418                         alu_out(111) <= '1';
419                     end if;
420                     alu_out(95 downto 80) <= std_logic_vector(signed(r2(95 downto 80))+signed(
421 r1(95 downto 80)));
422                     if alu_out(95)='1' and r2(95) = '0' and r1(95) = '0' then
423                         alu_out(95 downto 80) <= (others=>'1');
424                         alu_out(95) <= '0';
425                     elsif alu_out(95) = '0' and r2(95) = '1' and r1(95)='1' then
426                         alu_out(95 downto 80) <= (others=>'0');

```

```

413         alu_out(95)<='1';
414     end if;
415     alu_out(79 downto 64)<=std_logic_vector(signed(r2(79 downto 64))+signed(
r1(79 downto 64)));
416     if alu_out(79)='1' and r2(79) = '0' and r1(79) = '0' then

417         alu_out(79 downto 64)<=(others=>'1');
418         alu_out(79)<='0';
419     elsif alu_out(79) = '0' and r2(79) = '1' and r1(79)='1' then
420         alu_out(79 downto 64)<=(others=>'0');
421         alu_out(79)<='1';
422     end if;
423     alu_out(63 downto 48)<=std_logic_vector(signed(r2(63 downto 48))+signed(
r1(63 downto 48)));
424     if alu_out(63)='1' and r2(63) = '0' and r1(63) = '0' then

425         alu_out(63 downto 48)<=(others=>'1');
426         alu_out(63)<='0';
427     elsif alu_out(63) = '0' and r2(63) = '1' and r1(63)='1' then
428         alu_out(63 downto 48)<=(others=>'0');
429         alu_out(63)<='1';
430     end if;
431     alu_out(47 downto 32)<=std_logic_vector(signed(r2(47 downto 32))+signed(
r1(47 downto 32)));
432     if alu_out(47)='1' and r2(47) = '0' and r1(32) = '0' then

433         alu_out(47 downto 32)<=(others=>'1');
434         alu_out(47)<='0';
435     elsif alu_out(47) = '0' and r2(47) = '1' and r1(32)='1' then
436         alu_out(47 downto 32)<=(others=>'0');
437         alu_out(47)<='1';
438     end if;
439     alu_out(31 downto 16)<=std_logic_vector(signed(r2(31 downto 16))+signed(
r1(31 downto 16)));
440     if alu_out(31)='1' and r2(31) = '0' and r1(31) = '0' then

441         alu_out(31 downto 16)<=(others=>'1');
442         alu_out(31)<='0';
443     elsif alu_out(31) = '0' and r2(31) = '1' and r1(31)='1' then
444         alu_out(31 downto 16)<=(others=>'0');
445         alu_out(31)<='1';
446     end if;
447     alu_out(15 downto 0)<=std_logic_vector(signed(r2(15 downto 0))+signed(r1
(15 downto 0)));
448     if alu_out(15)='1' and r2(15) = '0' and r1(15) = '0' then

449         alu_out(15 downto 0)<=(others=>'1');
450         alu_out(15)<='0';
451     elsif alu_out(15) = '0' and r2(15) = '1' and r1(15)='1' then
452         alu_out(15 downto 0)<=(others=>'0');
453         alu_out(15)<='1';
454     end if;
455     -----AND: bitwise logical and
456     when "00100" =>
457         alu_out <= r1 and r2;
458     -----BCW: broadcast word
459     when "00101" =>
460         alu_out(127 downto 96)<=r1(31 downto 0);
461         alu_out(95 downto 64)<=r1(31 downto 0);
462         alu_out(63 downto 32)<=r1(31 downto 0);
463         alu_out(31 downto 0)<=r1(31 downto 0);
464
465     -----CLZ: count leading zeros in words
466     when "00110" =>
467         zero_count := 0;
468         for i in 127 downto 96 loop

```

```

469         if r1(i) = '0' then
470             zero_count := zero_count+1;
471         else
472             exit;
473         end if;
474     end loop;
475     alu_out(127 downto 96) <= std_logic_vector(to_unsigned(zero_count, 32
));
476     zero_count := 0;
477
478     for i in 95 downto 64 loop
479         if r1(i) = '0' then
480             zero_count := zero_count+1;
481         else
482             exit;
483         end if;
484     end loop;
485     alu_out(95 downto 64) <= std_logic_vector(to_unsigned(zero_count, 32
));
486     zero_count := 0;
487
488     for i in 63 downto 32 loop
489         if r1(i) = '0' then
490             zero_count := zero_count+1;
491         else
492             exit;
493         end if;
494     end loop;
495     alu_out(63 downto 32) <= std_logic_vector(to_unsigned(zero_count, 32
));
496     zero_count := 0;
497
498     for i in 31 downto 0 loop
499         if r1(i) = '0' then
500             zero_count := zero_count+1;
501         else
502             exit;
503         end if;
504     end loop;
505     alu_out(31 downto 0) <= std_logic_vector(to_unsigned(zero_count, 32
));
506     zero_count := 0;
507     -----MAX: max signed word
508     when "00111" =>
509         if signed(r1(127 downto 96)) > signed(r2(127 downto 96)) then
510             alu_out(127 downto 96) <= r1(127 downto 96);
511         else
512             alu_out(127 downto 96) <= r2(127 downto 96);
513         end if;
514
515         if(signed(r1(95 downto 64)) > signed(r2(95 downto 64))) then
516             alu_out(95 downto 64) <= r1(95 downto 64);
517         else
518             alu_out(95 downto 64) <= r2(95 downto 64);
519         end if;
520
521         if(signed(r1(63 downto 32)) > signed(r2(63 downto 32))) then
522             alu_out(63 downto 32) <= r1(63 downto 32);
523         else
524             alu_out(63 downto 32) <= r2(63 downto 32);
525         end if;
526
527         if(signed(r1(31 downto 0)) > signed(r2(31 downto 0))) then
528             alu_out(31 downto 0) <= r1(31 downto 0);
529         else
530             alu_out(31 downto 0) <= r2(31 downto 0);

```



```

531         end if;
532
533         -----MIN: min signed word
534         when "01000" =>
535             if (to_integer(signed(r1(127 downto 96))) > to_integer(signed(r2(127
536 downto 96)))) then
537                 alu_out(127 downto 96) <= r2(127 downto 96);
538             else
539                 alu_out(127 downto 96) <= r1(127 downto 96);
540             end if;
541
542             if signed(r1(95 downto 64)) > signed(r2(95 downto 64)) then
543                 alu_out(95 downto 64) <= r2(95 downto 64);
544             else
545                 alu_out(95 downto 64) <= r1(95 downto 64);
546             end if;
547
548             if signed(r1(63 downto 32)) > signed(r2(63 downto 32)) then
549                 alu_out(63 downto 32) <= r2(63 downto 32);
550             else
551                 alu_out(63 downto 32) <= r1(63 downto 32);
552             end if;
553
554             if signed(r1(31 downto 0)) > signed(r2(31 downto 0)) then
555                 alu_out(31 downto 0) <= r2(31 downto 0);
556             else
557                 alu_out(31 downto 0) <= r1(31 downto 0);
558             end if;
559
560         -----MSGN: multiply signed
561         -----???
562         when "01001" =>
563             if to_integer(signed(r1(127 downto 96))) = -2147483648 and r2(127) = '1'
564 then
565                 alu_out(127 downto 96) <= (Others => '1');
566                 alu_out(127) <= '0';
567             else
568                 if r2(127) = '1' then
569                     --signed(r1(127 downto 96))*(-1)
570                     alu_out(127 downto 96) <= std_logic_vector(to_signed(to_integer(
signed(r1(127 downto 96)))*(-1), 32));
571
572                     elsif r2(127 downto 96) = "00000000000000000000000000000000" then
573                         alu_out(127 downto 96) <= "00000000000000000000000000000000";
574                     else
575                         alu_out(127 downto 96) <= r1(127 downto 96);
576                     end if;
577                 end if;
578
579                 if to_integer(signed(r1(95 downto 64))) = -2147483648 and (r2(127) = '1')
580 then
581                     alu_out(95 downto 64) <= (Others => '1');
582                     alu_out(95) <= '0';
583                 else
584                     if (r2(95) = '1') then
585                         alu_out(95 downto 64) <= std_logic_vector(to_signed(to_integer(
signed(r1(95 downto 64)))*(-1), 32));
586                     elsif unsigned(r2(95 downto 64)) = 0 then
587                         alu_out(95 downto 64) <= "00000000000000000000000000000000";
588                     else
589                         alu_out(95 downto 64) <= r1(95 downto 64);
590                     end if;
591                 end if;

```

```

592
593         if to_integer(signed(r1(63 downto 32)))= -2147483648 and (r2(127)='1')
then
594             alu_out(63 downto 32)<=(Others=>'1');
595             alu_out(63)<='0';
596         else
597             if(r2(63)='1') then
598                 signed(r1(63 downto 32))<=std_logic_vector(to_signed(to_integer(
signed(r1(63 downto 32)))*(-1), 32));
599                 elsif unsigned(r2(63 downto 32))=0 then
600                     alu_out(63 downto 32)<="00000000000000000000000000000000" ;
601                 else
602                     alu_out(63 downto 32)<=r1(63 downto 32);
603                 end if;
604             end if;
605
606
607         if to_integer(signed(r1(31 downto 0)))= -2147483648 and (r2(127)='1')
then
608             alu_out(31 downto 0)<=(Others=>'1');
609             alu_out(31)<='0';
610         else
611             if(r2(31)='1') then
612                 signed(r1(31 downto 0))<=std_logic_vector(to_signed(to_integer(
signed(r1(31 downto 0)))*(-1), 32));
613                 elsif unsigned(r2(31 downto 0))=0 then
614                     alu_out(31 downto 0)<="00000000000000000000000000000000" ;
615                 else
616                     alu_out(31 downto 0)<=r1(31 downto 0);
617                 end if;
618             end if;
619             -----MPYU: multiply unsigned
620             when "01010" =>
621                 alu_out(127 downto 96)<=std_logic_vector(to_unsigned(to_integer(unsigned
(r1(111 downto 96)))*to_integer(unsigned(r2(111 downto 96))), 32));
622                 alu_out(95 downto 64)<=std_logic_vector(to_unsigned(to_integer(unsigned(
r1(79 downto 64)))*to_integer(unsigned(r2(79 downto 64))), 32));
623                 alu_out(63 downto 32)<=std_logic_vector(to_unsigned(to_integer(unsigned(
r1(47 downto 32)))*to_integer(unsigned(r2(47 downto 32))), 32));
624                 alu_out(31 downto 0)<=std_logic_vector(to_unsigned(to_integer(unsigned(
r1(15 downto 0)))*to_integer(unsigned(r2(15 downto 0))), 32));
625             -----OR: bitwise logical or
626             when "01011" =>
627                 alu_out <= r1 or r2;
628
629             -----POPCNTH: count ones in halfwords
630             when "01100" =>
631                 one_count := 0;
632                 for i in 127 downto 112 loop
633                     if r1(i)='1' then
634                         one_count := one_count+1;
635                     end if;
636                 end loop;
637                 alu_out(127 downto 112) <= std_logic_vector(to_unsigned(one_count, 16
));
638                 one_count:=0;
639
640                 for i in 111 downto 96 loop
641                     if r1(i)='1' then
642                         one_count := one_count +1;
643                     end if;
644                 end loop;
645                 alu_out(111 downto 96) <= std_logic_vector(to_unsigned(one_count, 16
));
646                 one_count:=0;
647

```

```

648         for i in 95 downto 80 loop
649             if r1(i)='1' then
650                 one_count := one_count+1;
651             end if;
652         end loop;
653         alu_out(95 downto 80) <= std_logic_vector(to_unsigned(one_count,16
654     ));
655     one_count:=0;
656     for i in 79 downto 64 loop
657         if r1(i)='1' then
658             one_count := one_count +1;
659         end if;
660     end loop;
661     alu_out(79 downto 64) <= std_logic_vector(to_unsigned(one_count,16
662 ));
663     one_count:=0;
664     for i in 63 downto 48 loop
665         if r1(i)='1' then
666             one_count := one_count+1;
667         end if;
668     end loop;
669     alu_out(63 downto 48) <= std_logic_vector(to_unsigned(one_count,16
670 ));
671     one_count:=0;
672     for i in 47 downto 32 loop
673         if r1(i)='1' then
674             one_count := one_count +1;
675         end if;
676     end loop;
677     alu_out(47 downto 32) <= std_logic_vector(to_unsigned(one_count,16
678 ));
679     one_count:=0;
680     for i in 31 downto 16 loop
681         if r1(i)='1' then
682             one_count := one_count+1;
683         end if;
684     end loop;
685     alu_out(31 downto 16) <= std_logic_vector(to_unsigned(one_count,16
686 ));
687     one_count:=0;
688     for i in 15 downto 0 loop
689         if r1(i)='1' then
690             one_count := one_count +1;
691         end if;
692     end loop;
693     alu_out(15 downto 0) <= std_logic_vector(to_unsigned(one_count,16));
694     one_count:=0;
695
696     -----ROT: rotate bits right
697     when "01101" =>
698         bits_of_shift:=to_integer(unsigned(r2(6 downto 0)));
699         alu_out<=r1 ror bits_of_shift;
700
701     -----ROTW: rotate bits in word
702     when "01110" =>
703         bits_of_shift:=to_integer(unsigned(r2(100 downto 96)));
704         alu_out(127 downto 96)<=r1(127 downto 96) ror bits_of_shift;
705         bits_of_shift:=to_integer(unsigned(r2(68 downto 64)));
706         alu_out(95 downto 64)<=r1(95 downto 64) ror bits_of_shift;
707         bits_of_shift:=to_integer(unsigned(r2(36 downto 32)));
708         alu_out(63 downto 32)<=r1(63 downto 32) ror bits_of_shift;

```

```

709         bits_of_shift:=to_integer(unsigned(r2(4 downto 0)));
710         alu_out(31 downto 0)<=r1(31 downto 0) ror bits_of_shift;
711
712         -----SHLHI: shift left halfword immediate:
713         -----???
714         when "01111" =>
715             bits_of_shift:=to_integer(unsigned(r2(115 downto 112)));
716             alu_out(127 downto 112)<=r1(127 downto 112) SLL bits_of_shift;
717             bits_of_shift:=to_integer(unsigned(r2(99 downto 96)));
718             alu_out(111 downto 96)<=r1(111 downto 96) SLL bits_of_shift;
719             bits_of_shift:=to_integer(unsigned(r2(83 downto 80)));
720             alu_out(95 downto 80)<=r1(95 downto 80) SLL bits_of_shift;
721             bits_of_shift:=to_integer(unsigned(r2(67 downto 64)));
722             alu_out(79 downto 64)<=r1(79 downto 64) SLL bits_of_shift;
723             bits_of_shift:=to_integer(unsigned(r2(51 downto 48)));
724             alu_out(63 downto 48)<=r1(63 downto 48) SLL bits_of_shift;
725             bits_of_shift:=to_integer(unsigned(r2(35 downto 32)));
726             alu_out(47 downto 32)<=r1(47 downto 32) SLL bits_of_shift;
727             bits_of_shift:=to_integer(unsigned(r2(19 downto 16)));
728             alu_out(31 downto 16)<=r1(31 downto 16) SLL bits_of_shift;
729             bits_of_shift:=to_integer(unsigned(r2(3 downto 0)));
730             alu_out(15 downto 0)<=r1(15 downto 0) SLL bits_of_shift;
731
732         -----SFH: subtract from halfword immediate
733         when "10000" =>
734             alu_out(127 downto 112)<=std_logic_vector(to_unsigned(to_integer(
735 unsigned(r2(127 downto 112))-to_integer(unsigned(r2(127 downto 112))),16));
736             alu_out(111 downto 96)<=std_logic_vector(to_unsigned(to_integer(unsigned(
737 (r2(111 downto 96))-to_integer(unsigned(r2(111 downto 96))),16));
738             alu_out(95 downto 80)<=std_logic_vector(to_unsigned(to_integer(unsigned(
739 r2(95 downto 80))-to_integer(unsigned(r2(95 downto 80))),16));
740             alu_out(79 downto 64)<=std_logic_vector(to_unsigned(to_integer(unsigned(
741 r2(79 downto 64))-to_integer(unsigned(r2(79 downto 64))),16));
742             alu_out(63 downto 48)<=std_logic_vector(to_unsigned(to_integer(unsigned(
743 r2(63 downto 48))-to_integer(unsigned(r2(63 downto 48))),16));
744             alu_out(47 downto 32)<=std_logic_vector(to_unsigned(to_integer(unsigned(
745 r2(47 downto 32))-to_integer(unsigned(r2(47 downto 32))),16));
746             alu_out(31 downto 16)<=std_logic_vector(to_unsigned(to_integer(unsigned(
747 r2(31 downto 16))-to_integer(unsigned(r2(31 downto 16))),16));
748             alu_out(15 downto 0)<=std_logic_vector(to_unsigned(to_integer(unsigned(
749 r2(15 downto 0))-to_integer(unsigned(r2(15 downto 0))),16));
750
751         -----SFW: subtract from word
752         when "10001" =>
753             alu_out(127 downto 96)<=std_logic_vector(to_unsigned(to_integer(unsigned(
754 (r2(127 downto 96))-to_integer(unsigned(r2(127 downto 96))),32));
755             alu_out(95 downto 64)<=std_logic_vector(to_unsigned(to_integer(unsigned(
756 r2(95 downto 64))-to_integer(unsigned(r2(95 downto 64))),32));
757             alu_out(63 downto 32)<=std_logic_vector(to_unsigned(to_integer(unsigned(
758 r2(63 downto 32))-to_integer(unsigned(r2(63 downto 32))),32));
759             alu_out(31 downto 0)<=std_logic_vector(to_unsigned(to_integer(unsigned(
760 r2(31 downto 0))-to_integer(unsigned(r2(31 downto 0))),32));
761
762         -----SFHS: subtract from halfword saturated
763         -----saturation??
764         when "10010" =>
765             alu_out(127 downto 112)<=std_logic_vector(signed(r2(127 downto 112)) -
766 signed(r1(127 downto 112)));
767             if alu_out(127)='0' and r2(127)='1' and r1(127)='0' then
768                 alu_out(127 downto 112) <= (others=>'0');
769                 alu_out(127)<='1';
770             elsif alu_out(127)='1' and r2(127)='0' and r2(127)='1' then
771                 alu_out(127 downto 112) <= (others=>'1');
772                 alu_out(127)<='0';
773             end if;

```

```

762         alu_out(111 downto 96) <= std_logic_vector (signed(r2(111 downto 96)) -
signed(r1(111 downto 96)));
763         if alu_out(111)='0' and r2(111)='1' and r1(111)='0' then
764             alu_out(111 downto 96) <= (others=>'0');
765             alu_out(111) <='1';
766         elsif alu_out(111)='1' and r2(111)='0' and r2(111)='1' then
767             alu_out(111 downto 96) <= (others=>'1');
768             alu_out(111) <='0';
769         end if;
770
771         alu_out(95 downto 80) <= std_logic_vector (signed(r2(95 downto 80)) - signed(
r1(95 downto 80)));
772         if alu_out(95)='0' and r2(95)='1' and r1(95)='0' then
773             alu_out(95 downto 80) <= (others=>'0');
774             alu_out(95) <='1';
775         elsif alu_out(95)='1' and r2(95)='0' and r2(95)='1' then
776             alu_out(95 downto 80) <= (others=>'1');
777             alu_out(95) <='0';
778         end if;
779
780         alu_out(79 downto 64) <= std_logic_vector (signed(r2(79 downto 64)) - signed(
r1(79 downto 64)));
781         if alu_out(79)='0' and r2(79)='1' and r1(79)='0' then
782             alu_out(79 downto 64) <= (others=>'0');
783             alu_out(79) <='1';
784         elsif alu_out(79)='1' and r2(79)='0' and r2(79)='1' then
785             alu_out(79 downto 64) <= (others=>'1');
786             alu_out(79) <='0';
787         end if;
788
789         alu_out(63 downto 48) <= std_logic_vector (signed(r2(63 downto 48)) - signed(
r1(63 downto 48)));
790         if alu_out(63)='0' and r2(63)='1' and r1(63)='0' then
791             alu_out(63 downto 48) <= (others=>'0');
792             alu_out(63) <='1';
793         elsif alu_out(63)='1' and r2(63)='0' and r2(63)='1' then
794             alu_out(63 downto 48) <= (others=>'1');
795             alu_out(63) <='0';
796         end if;
797
798         alu_out(47 downto 32) <= std_logic_vector (signed(r2(47 downto 32)) - signed(
r1(47 downto 32)));
799         if alu_out(47)='0' and r2(47)='1' and r1(47)='0' then
800             alu_out(47 downto 32) <= (others=>'0');
801             alu_out(47) <='1';
802         elsif alu_out(47)='1' and r2(47)='0' and r2(47)='1' then
803             alu_out(47 downto 32) <= (others=>'1');
804             alu_out(47) <='0';
805         end if;
806
807         alu_out(31 downto 16) <= std_logic_vector (signed(r2(31 downto 16)) - signed(
r1(31 downto 16)));
808         if alu_out(31)='0' and r2(31)='1' and r1(31)='0' then
809             alu_out(31 downto 16) <= (others=>'0');
810             alu_out(31) <='1';
811         elsif alu_out(31)='1' and r2(31)='0' and r2(31)='1' then
812             alu_out(31 downto 16) <= (others=>'1');
813             alu_out(31) <='0';
814         end if;
815
816         alu_out(15 downto 0) <= std_logic_vector (signed(r2(15 downto 0)) - signed(r1
(15 downto 0)));
817         if alu_out(15)='0' and r2(15)='1' and r1(15)='0' then
818             alu_out(15 downto 0) <= (others=>'0');
819             alu_out(15) <='1';
820         elsif alu_out(15)='1' and r2(15)='0' and r2(15)='1' then

```

```
821             alu_out(15 downto 0) <= (others=>'1');
822             alu_out(15)<='0';
823         end if;
824
825         -----XOR: bitwise logical exclusive-or
826         when "10011" =>
827             alu_out <= r1 xor r2;
828         when others => null;
829
830     end case;
831 end if;
832 end if;
833 end process;
834
835 end behavioral;
```

```

1  -----
2  --
3  -- Title       : forwarding_unit
4  -- Design      : processor
5  -- Author      :
6  -- Company     :
7  --
8  -----
9  --
10 -- File        :
11 C:\Users\gavin\Desktop\Study\ESE345\project\processor\processor\src\forwarding_unit.vhd
12 -- Generated   : Sat Nov 30 17:16:23 2019
13 -- From        : interface description file
14 -- By          : Itf2Vhdl ver. 1.22
15 --
16 -----
17 -- Description :
18 --
19 -----
20
21 --{{ Section below this comment is automatically maintained
22 --   and may be overwritten
23 --{entity {forwarding_unit} architecture {\behavioral \}}
24
25 library ieee;
26 use ieee.std_logic_1164.all;
27 use ieee.numeric_std.all;
28
29 entity forwarding_unit is
30     port(
31         ALU_out : in std_logic_vector (127 downto 0);
32         r1_in : in std_logic_vector (127 downto 0);
33         r2_in : in std_logic_vector (127 downto 0);
34         r3_in : in std_logic_vector (127 downto 0);
35         ins : in std_logic_vector (24 downto 0);
36         old_ins : in std_logic_vector (24 downto 0);
37         r1_out : out std_logic_vector (127 downto 0);
38         r2_out : out std_logic_vector (127 downto 0);
39         r3_out : out std_logic_vector (127 downto 0)
40     );
41 end forwarding_unit;
42
43 --}} End of automatically maintained section
44
45 architecture behavioral of forwarding_unit is
46     signal older_ins : std_logic_vector (24 downto 0);
47     signal old_out : std_logic_vector (127 downto 0);
48     begin
49         process (ALU_out, r1_in, r2_in, r3_in, ins, old_ins)
50         begin
51             if ins(24) = '0' then
52                 if old_ins(4 downto 0) = ins(4 downto 0) then
53                     r1_out <= ALU_out;
54                     r2_out <= r2_in;
55                     r3_out <= r3_in;
56                 elsif older_ins(4 downto 0) = ins(4 downto 0) then
57                     r1_out <= old_out;
58                     r2_out <= r2_in;
59                     r3_out <= r3_in;
60                 else
61                     r1_out <= r1_in;
62                     r2_out <= r2_in;
63                     r3_out <= r3_in;
64                 end if;
65             elsif ins(24) = '1' then

```

```

66         if ins(23) = '1' then
67             if ins(19 downto 15) = "00000" then
68                 r1_out <= r1_in;
69                 r2_out <= r2_in;
70                 r3_out <= r3_in;
71             else
72                 if old_ins(4 downto 0) = ins(9 downto 5) then
73                     r1_out <= ALU_out;
74                     r2_out <= r2_in;
75                     r3_out <= r3_in;
76                 elsif old_ins(4 downto 0) = ins(14 downto 10) then
77                     r2_out <= ALU_out;
78                     r1_out <= r1_in;
79                     r3_out <= r3_in;
80                 elsif older_ins(4 downto 0) = ins(9 downto 5) then
81                     r1_out <= old_out;
82                     r2_out <= r2_in;
83                     r3_out <= r3_in;
84                 elsif older_ins(4 downto 0) = ins(14 downto 10) then
85                     r2_out <= old_out;
86                     r1_out <= r1_in;
87                     r3_out <= r3_in;
88                 else
89                     r1_out <= r1_in;
90                     r2_out <= r2_in;
91                     r3_out <= r3_in;
92                 end if;
93             end if;
94         elsif ins(23) = '0' then
95             if old_ins(4 downto 0) = ins(9 downto 5) then
96                 r1_out <= ALU_out;
97                 r2_out <= r2_in;
98                 r3_out <= r3_in;
99             elsif old_ins(4 downto 0) = ins(14 downto 10) then
100                 r2_out <= ALU_out;
101                 r1_out <= r1_in;
102                 r3_out <= r3_in;
103             elsif old_ins(4 downto 0) = ins(19 downto 15) then
104                 r3_out <= ALU_out;
105                 r1_out <= r1_in;
106                 r2_out <= r2_in;
107             elsif older_ins(4 downto 0) = ins(9 downto 5) then
108                 r1_out <= old_out;
109                 r2_out <= r2_in;
110                 r3_out <= r3_in;
111             elsif older_ins(4 downto 0) = ins(14 downto 10) then
112                 r2_out <= old_out;
113                 r1_out <= r1_in;
114                 r3_out <= r3_in;
115             elsif older_ins(4 downto 0) = ins(19 downto 15) then
116                 r3_out <= old_out;
117                 r1_out <= r1_in;
118                 r2_out <= r2_in;
119             else
120                 r1_out <= r1_in;
121                 r2_out <= r2_in;
122                 r3_out <= r3_in;
123             end if;
124         end if;
125     end if;
126     older_ins <= old_ins;
127     old_out <= ALU_out;
128 end process;
129
130
131     -- enter your statements here --

```


File: f:/project345/src/forwarding_unit.vhd (/file_io_tb/UUT/u4)

```
132
133     end behavioral;
134
```

```
25  library ieee;
26  use ieee.std_logic_1164.all;
27  use ieee.numeric_std.all;
28
29  package insbuffer_type is
30  type insBuffer is array(63 downto 0) of std_logic_vector (24 downto 0);
31  end package insbuffer_type;
32  use work.insbuffer_type.all;
33
34  library ieee;
35  use ieee.std_logic_1164.all;
36  use ieee.numeric_std.all;
37  use IEEE.STD_LOGIC_UNSIGNED.ALL;
38
39  entity ins_buffer is
40  port(
41  instructions : in insBuffer;
42  clk : in std_logic;
43  ins_out : out std_logic_vector(24 downto 0)
44  );
45  end ins_buffer;
46
47  --}} End of automatically maintained section
48
49  architecture behavioral of ins_buffer is
50  signal PC : std_logic_vector(5 downto 0) := "000000";
51  begin
52  process(clk, instructions)
53  begin
54  if rising_edge(clk) then
55  ins_out <= instructions(to_integer(unsigned(PC)));
56  PC <= PC + std_logic_vector(to_unsigned(1, PC'length));
57  end if;
58  end process;
59
60  -- enter your statements here --
61
62  end behavioral;
```

```

1  -----
2  --
3  -- Title       : register_file
4  -- Design      : processor
5  -- Author      :
6  -- Company     :
7  --
8  -----
9  --
10 -- File        :
    C:\Users\gavin\Desktop\Study\ESE345\project\processor\processor\src\register_file.vhd
11 -- Generated   : Sat Nov 30 15:35:50 2019
12 -- From        : interface description file
13 -- By          : Itf2Vhdl ver. 1.22
14 --
15 -----
16 --
17 -- Description :
18 --
19 -----
20
21 --{{ Section below this comment is automatically maintained
22 --   and may be overwritten
23 --{entity {register_file} architecture {behavioral}}
24
25 library IEEE;
26 use IEEE.std_logic_1164.all;
27 use ieee.numeric_std.all;
28
29 entity register_file is
30     port(
31         WrData : in STD_LOGIC_VECTOR (127 downto 0);
32         ins : in STD_LOGIC_VECTOR (24 downto 0);
33         old_ins : in std_logic_vector (24 downto 0);
34         r1 : out STD_LOGIC_VECTOR (127 downto 0);
35         r2 : out STD_LOGIC_VECTOR (127 downto 0);
36         r3 : out STD_LOGIC_VECTOR (127 downto 0);
37         ins_out : out std_logic_vector (24 downto 0);
38         clk : in std_logic
39     );
40 end register_file;
41
42 --}} End of automatically maintained section
43
44 architecture behavioral of register_file is
45     type RegisterFile is array (31 downto 0) of std_logic_vector (127 downto 0);
46     signal regs : RegisterFile := ((others=>(others=>'0')));
47
48 begin
49     process (clk, WrData, ins)
50     begin
51         if rising_edge (clk) then
52             if old_ins (24 downto 23) = "11" then
53                 if old_ins (19 downto 15) = "00000" then
54                     null;
55                 else
56                     regs (to_integer (unsigned (old_ins (4 downto 0)))) <= WrData;
57                 end if;
58             else
59                 regs (to_integer (unsigned (old_ins (4 downto 0)))) <= WrData;
60             end if;
61             if ins (24) = '0' then
62                 r1 <= regs (to_integer (unsigned (ins (4 downto 0))));
63                 r2 <= regs (to_integer (unsigned (ins (14 downto 10))));
64
65                 r3 <= regs (to_integer (unsigned (ins (19 downto 15))));

```

```
64         end if;
65         if ins(9 downto 5) = old_ins(4 downto 0) then
66             r1 <= WrData;
67             r2 <= regs(to_integer(unsigned(ins(14 downto 10))));
68
69             r3 <= regs(to_integer(unsigned(ins(19 downto 15))));
70         elsif ins(9 downto 5) = old_ins(4 downto 0) then
71             r2 <= WrData;
72             r1 <= regs(to_integer(unsigned(ins(9 downto 5))));
73
74             r3 <= regs(to_integer(unsigned(ins(19 downto 15))));
75         elsif ins(9 downto 5) = old_ins(4 downto 0) then
76             r3 <= WrData ;
77             r2 <= regs(to_integer(unsigned(ins(14 downto 10))));
78
79             r1 <= regs(to_integer(unsigned(ins(9 downto 5))));
80         else
81             r1 <= regs(to_integer(unsigned(ins(9 downto 5))));
82             r2 <= regs(to_integer(unsigned(ins(14 downto 10))));
83
84             r3 <= regs(to_integer(unsigned(ins(19 downto 15))));
85         end if;
86         ins out <= ins;
87     end if;
88 end process;
89
90     -- enter your statements here --
91
92 end behavioral;
```

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  use STD.textio.all;
5  use ieee.std_logic_textio.all;
6  use work.insbuffer_type.all;
7
8  entity file_io_tb is
9  end file_io_tb;
10
11
12  architecture behave of file_io_tb is
13
14      -----
15      --
16      -- Declare the Component Under Test
17      -----
18
19      signal tb_clk: std_logic;
20      signal tb_alu_out: std_logic_vector(127 downto 0);
21      signal rom_data: insBuffer;
22      signal count: integer := 0;
23      signal alu_out: std_logic_vector(127 downto 0);
24
25      constant period : time := 20 ns;
26
27      -----
28      --
29      -- Instantiate and Map UUT
30      -----
31
32      file file_input : text;
33      file file_output: text;
34
35      begin
36          UUT: entity pipeline_unit port map(
37              ins_in => rom_data,
38              clk => tb_clk,
39              ALU_o => tb_alu_out
40          );
41
42
43
44      -----
45      -- This procedure reads the file input_vectors.txt which is located in
46      -- the
47      -- simulation project area.
48      -- It will read the data in and send it to the ripple-adder component
49      -- to perform the operations. The result is written to the
50      -- output_results.txt file, located in the same directory.
51      -----
52
53      process
54          variable ILINE : line;
55          variable INS_LINE : std_logic_vector(24 downto 0);
56          variable count: integer:=0;
```

```
55     variable v_out : line;
56
57     begin
58         file_open(file_input, "opcode.txt", read_mode);
59         file_open(file_output, "alu_out.txt", write_mode);
60
61
62         while not endfile(file_input) loop
63             readline(file_input, ILINE);
64             read(ILINE, INS_LINE);
65             rom_data(count) <= INS_LINE;
66             count := count+1;
67         end loop;
68         file_close(file_input);
69
70
71         for i in 0 to count +4 loop
72             wait for period;
73             write(v_out, tb_alu_out);
74             writeline(file_output, v_out);
75         end loop;
76         file_close(file_output);
77
78     end process;
79
80
81
82     clock: process
83     begin
84         tb_clk <='0';
85         wait for 10ns;
86         tb_clk<='1';
87         wait for 10ns;
88     end process;
89 end behave;
```