ECE1756 assignment 3 RAM mapper - Report Zhiyuan Gu (1004920400)

1. Mapping algorithm

   The RAM mapper doesn't exhaustively explore all combinations of mapping solutions for each logical RAM. Instead, it tries to calculate good initial solutions for all logical RAMs and then changes solutions of some logical RAMs to balance the use of different types of physical RAMs.

   1) Initial solution for each logical RAM is achieved by calculating all solutions when mapping the logical RAM to all different types of physical RAMs. Note that only one type of physical RAM is used for any solution of a logical RAM. In Stratix IV-like architecture, there are three types of physical RAMs, so each logical RAM would have three solutions mapping to each of the physical RAMs. Then the area of each solution is calculated regardless of ratio among physical RAMs. This means the area calculated only considers RAMs that are used but ignores possible extra logic blocks that might be introduced due to architectural relationship. After that, the best solution among the three is chosen for each logical RAM. This initial solution is achieved very fast because the calculation is constant time for each logical RAM to map. As a result, time complexity of finding initial solution is O(n), where n is the number of logical RAMs.
   2) The second step is to balance the use of different physical RAMs in case some are overused and lead to extreme area increase. This includes steps to change solutions of logical RAMs from the most area-requiring type of physical RAM to the least requiring one. This process continues until no more area can be saved. The time complexity to calculate which type of physical RAM is the limiting factor is O(n) by gathering the total number used in each balancing step. There are three arrays representing three physical RAMs storing all the logical RAMs that use them. Creating the array also requires O(n) time for each step. In order to determine which logical RAM should be chosen to change solution, the array of the most area-requiring type of physical RAM is sorted according to area increase in percentage when changing from old to new solution. It is preferred to change the solutions that create less overhead after changing but help balance the use of different physical RAMs. The sorting process consumes O(nlog(n)) time for each balancing step. Balancing continues until no more area could be saved. This either means a well-balanced usage or too much overhead for further balancing. Balancing is unidirectional, relieving the most area-requiring type of physical RAM. Though the most and least area-requiring type would change during balancing, solutions are not changed back. Because solution changes are chosen to save area, the reverse of the process leads to increase in overall area which would terminate the whole balancing process. Consequently, total balancing steps should be less than the number of logic RAMs and time complexity of balancing is $O(n^2\log(n))$ which includes n balancing steps with sorting being the dominant calculation in each step.
   3) In fact, area comparison is done after changing the solution for logical RAMs and updating physical RAM usage information. As a result, the last step of balancing is bound to increase total area requirement. To tackle this, one more step is done after balancing to reverse the last solution change and achieve the best area found in balancing process.

4) In conclusion, timing complexity of the mapping algorithm is $O(n^2 \log(n))$ as explained in the balancing part. On the other hand, space complexity is merely $O(n)$ because each logic RAM only has limited amount of solutions to store during calculation.

2. Results for a RAM architecture (similar to Stratix IV)

Table 1: RAM mapping results for example Stratix-IV like architecture

Circuit	Type 1	Type 2	Type 3	Blocks	Tiles	Area
0	784	361	12	2941	3725	1.85812e+008
1	664	354	26	2906	7800	3.89925e+008
2	1	57	2	1836	1837	9.16599e+007
3	2	73	3	2808	2810	1.40161e+008
4	488	642	21	7907	8395	4.18785e+008
5	1	262	9	3692	3693	1.84322e+008
6	39	182	6	1853	1892	9.43017e+007
7	263	425	14	3947	4250	2.12318e+008
8	118	546	17	5342	5460	2.72778e+008
9	1	32	0	1636	1637	8.13783e+007
10	378	177	8	1418	2400	1.19977e+008
11	163	84	4	1329	1492	7.37385e+007
12	1	10	2	1632	1633	8.12283e+007
13	1	21	1	4491	4492	2.2371e+008
14	53	188	8	1808	2400	1.19977e+008
15	4	87	4	1956	1960	9.75276e+007
16	1	55	2	2181	2182	1.08827e+008
17	1	59	1	1165	1166	5.74767e+007
18	175	12	8	2034	2400	1.19977e+008
19	159	175	15	2230	4500	2.24957e+008
20	46	264	7	2679	2725	1.36105e+008
21	1	55	2	5100	5101	2.54989e+008
22	308	351	11	3012	3510	1.74871e+008
23	0	106	11	5230	5230	2.61081e+008
24	44	421	14	4325	4369	2.17842e+008
25	1	79	8	4517	4518	2.25729e+008
26	43	129	20	1323	6000	2.99943e+008
27	32	0	0	1496	1528	7.62288e+007
28	98	281	5	1993	2810	1.40161e+008
29	82	301	10	3025	3107	1.54949e+008
30	2	199	7	5419	5421	2.70929e+008
31	126	1	0	4347	4473	2.22804e+008
32	182	353	27	3473	8100	4.04923e+008
33	30	400	10	4006	4036	2.01318e+008
34	51	160	30	1705	9000	4.49914e+008
35	20	150	5	1488	1508	7.52857e+007

36	230	170	43	1561	12900	6.44877e+008
37	0	48	0	14969	14969	7.47457e+008
38	19	113	18	3190	5400	2.69948e+008
39	218	220	7	1884	2200	1.09695e+008
40	1	146	6	3060	3061	1.52838e+008
41	156	250	11	1955	3300	1.64968e+008
42	1	69	3	1337	1338	6.64187e+007
43	126	123	4	1212	1338	6.64187e+007
44	1	194	7	2114	2115	1.05639e+008
45	2	12	1	2782	2784	1.38897e+008
46	352	268	23	3360	6900	3.44934e+008
47	1	50	2	1439	1440	7.13057e+007
48	92	112	44	10128	13200	6.59874e+008
49	96	1196	84	11883	25200	1.25976e+009
50	1	523	18	11884	11885	5.93563e+008
51	1	419	11	4204	4205	2.10147e+008
52	1	565	19	9603	9604	4.80058e+008
53	13	712	24	10817	10830	5.41311e+008
54	1	847	29	10903	10904	5.44762e+008
55	159	1045	35	10341	10500	5.249e+008
56	1	266	9	4578	4579	2.28595e+008
57	4	417	14	7145	7149	3.56589e+008
58	860	254	2	7700	8560	4.27464e+008
59	152	2305	76	22589	23050	1.15157e+009
60	10	552	0	20371	20381	1.01805e+009
61	0	1926	50	15079	19260	9.62646e+008
62	64	482	16	4888	4952	2.47102e+008
63	0	327	19	4846	5700	2.84945e+008
64	1109	936	38	10451	11560	5.77436e+008
65	1	315	12	12721	12722	6.35613e+008
66	4	594	20	6310	6314	3.15561e+008
67	334	555	15	4612	5550	2.77022e+008
68	1	184	7	4850	4851	2.42349e+008

Total CPU time: 10.742s (on my laptop) or 0.41s (on ECF machine)

Geometric Average Area: $2.287*10^8$

3. Results for a RAM architecture (No LUTRAM, one type of block RAM)

Table 2: Results for an architecture without LUTRAM

Block RAM size	Max width	Logic blocks / RAM block	Geometric average area
1kbit	8	4	$7.27*10^8$
2kbit	16	4	$4.90*10^8$
4kbit	16	5	$3.45*10^8$
8kbit	32	7	$2.66*10^8$
16kbit	32	8	$2.22*10^8$
32kbit	64	12	$2.43*10^8$
64kbit	64	16	$2.84*10^8$
128kbit	128	26	$3.52*10^8$

Trend:

Maximum width increases with the size of block RAM because deep RAMs need to be flexible enough to cater to wider logical RAMs that is not so deep. Since there is no other type of physical RAMs, lack in flexibility simply means huge waste. On the other hand, number of logic blocks per RAM block is also increasing due to the larger size of block RAMs. Less RAMs need to be serialized or parallelized to create a large logical RAM. Finally, the geometric average area sees a lowest point in the middle at block RAM size 16kbit. This is reasonable because smaller RAMs introduce more overhead when serializing while larger RAMs cause more waste for small requirements. The one block RAM size that better reflects the requirement of the circuits wins out.

4. Results for a RAM architecture (50% LUTRAM, one type of block RAM)

Table 3: Results for an architecture with LUTRAM

Block RAM size	Max width	Logic blocks / RAM block	Geometric average area
1kbit	8	5	$7.25*10^8$
2kbit	8	4	$4.85*10^8$
4kbit	16	6	$3.35*10^8$
8kbit	16	8	$2.55*10^8$
16kbit	32	11	$2.05*10^8$
32kbit	64	20	$2.16*10^8$
64kbit	64	28	$2.34*10^8$
128kbit	128	45	$2.67*10^8$

Trend:

One can tell the main trend of results remains the same with the architecture excluding LUTRAMs. However, the need for block RAMs goes down in that LUTRAMs now play a fair part in mapping. More choices also guarantee better results regarding geometric average area when compared with the architecture not supporting LUTRAM. This improvement is more apparent when larger block RAMs are chosen because LUTRAM can map small logical RAMs very efficiently which is the weakness of large block RAMs.

5. Results for a new architecture (SRAM for block RAMs)

Geometric average area:$1.949*10^8$ (compared with $2.287*10^8$ for Stratix IV-like architecture)

Table 4: Parameters of the new architecture (SRAM for block RAMs)

	maximum bit count	maximum width	ratio
LUTRAM	N/A	N/A	6.6
BRAM1(SRAM)	4k	16	12
BRAM2(SRAM)	32k	32	36

The comparison between the two previous sections shows that offering LUTRAM is a good idea in reducing total area. As included in the table, "ratio" of LUTRAM means on average one LUTRAM is required for every 6.6 logic blocks. My architecture is superior to the one of Stratix IV in that my architecture better reflects the requirement of benchmark circuits. Less overhead and waste is created because all the configurable parameters are carefully chosen to give better area results. Ratio of each block RAM is just enough without being a waste.

6. Results for a new architecture (MTJ for block RAMs)

Geometric average area: $1.730*10^8$ (compared with $1.949*10^8$ for SRAM-based architecture)

Table 5: Parameters of the new architecture (MTJ for block RAMs)

	maximum bit count	maximum width	ratio
LUTRAM	N/A	N/A	6.03
BRAM1(MTJ)	4k	16	10
BRAM2(MTJ)	64k	32	36

At first, the same two block RAM sizes were chosen to test the effect of MTJ RAMs. Not surprisingly, the ratio of two block RAMs dropped a lot which means more of them are needed. This is due to less area cost of MTJ-based block RAMs. What was counterintuitive was that the ratio of LUTRAMs also dropped. More logical RAMs should now be using block RAMs and less should be using LUTRAMs. However, the fact that larger block RAMs are now cheaper to use also means less LUTRAMs will need to be serialized to create large logical RAMs and thus creating less extra LUT overhead. In other words, less logic blocks are needed in the same time that less LUTRAMs are used. As a result, the ratio of LUTRAM also decreased.

Afterwards, block RAM sizes are changed to try different architectures. It is found that larger block RAMs work better than the SRAM case. It is the huge area that stops larger block RAMs from being used. Now the area is less an issue, larger block RAMs are preferred more. Results show increasing the second block RAM size from 32k to 64k gives better area result.