Part (a) Algorithm Description

First, the algorithm generates all the physical rams with different dimensions according to the input memory types and their maximum widths. Then, the algorithm iterates through all the circuits. It only does basic mappings, where each logical ram is mapped to only one type of physical ram. The logical rams of each circuit are sorted in a descending order of logical area before mappings start.

Resource Availability Aware

To allocate the physical resources (BRAMs and LBs) wisely, the algorithm keeps track of both the amount of resources required on the FPGA and the amount actually used. Specifically, it keeps updating the 2 groups of values listed below after mapping each logical ram so that it can be always "aware" of the availabilities of each type of physical resource.

1) Number of required BRAM blocks for each BRAM type and number of required LBs on the FPGA.

2) Number of used BRAM blocks for each BRAM type, number of used LUTRAMs and number of used non-ram related LBs.

Suppose the algorithm decides to map a logical ram to an 8k BRAM on an FPGA with a Stratix-IV like architecture. The number of used 8k BRAM blocks will increment by 1. If there is already one or more 8k rams as a required BRAM block on the FPGA, the group of required numbers will not be updated. Otherwise, the required number of 8k BRAM blocks will increment by 1 while the required number of LBs will have to be incremented to the smallest value divisible by 10 as the ratio between LBs and 8k BRAMs is 10:1.

Next-Fit like policy

In a standard Next-Fit algorithm [1], the next "bin" won't be created until the current "item" can't fit the current "bin". Similarly, my algorithm won't add more required resources on the FPGA until the current logical ram can't be mapped to any type of physical rams with any dimensions plus LBs needed limited by the currently available resources.

If the available resources are sufficient, the algorithm will choose a mapping that minimizes area consumption. Otherwise, the algorithm will choose a mapping that allocates the least number of extra LBs. As the physical resources are required in fixed ratios, minimum number of extra LBs means the numbers of any other types of extra required resource blocks are also minimized.

Part (b) Time Complexity

The algorithm main stages and their corresponding Big-O analysis are listed below. The number of input physical memory types is considered as a constant.

1) Generate all physical rams with different dimensions: O(1)

2) Sort the logical rams (N) in descending area order for all circuits (M): O(M*NlogN)

3) Generate mappings for the logical rams in all circuits. Each logical ram is only visited once and the update of required and used resources is constant time: O(M*N)

Therefore, the overall time complexity of the algorithm is O(M*NlogN), where M is the number of circuits and N is the average number of logical rams from each circuit.

Part (c) Mapping Tool Source Code

See Appendix A.

Part (d) Results for the Stratix-IV-like architecture

Total CPU Runtime on UG machine: **0.72s - 0.74s**

Geometric Average Area: **2.04004e08**

Command to run: **python lab3.py**

Mapping Results

Figure 1 shows the mapping results for each circuit.

Part (e) Area with No LUTRAM and One BRAM Type

The Max Widths and LBs / BRAM ratios explored for different BRAM sizes are [1, 2 ,4, 8, 16, 32, 64, 128, 256, 512] and [1, 2, 4, 6, 8, 10, 20, 32, 40, 50, 80, 100, 128, 256, 300] respectively. See Appendix B for the top 20 lowest areas for BRAMs of sizes from 1k to 128k.

Figure 2 shows the Max Widths and LBs / BRAM ratios minimizing the geometric average areas for each BRAM of different sizes.

The first trend observed is that the geometric average area decreases when BRAM Size increases until the size is 8 kbit, where the area reaches the lowest point 2.143e8. When BRAM Size becomes larger than 8 kbit, the area starts to increase with BRAM Size. Initially

Circuit	Type1	Type2	Type3	Blocks	Tiles	Area	Circuit	Type1	Type2	Type3	Blocks	Tiles	Area	Circuit	Type1	Type2	Type3	Blocks	Tiles	Area
0	811	372	11	2941	3752	**1.87E+08**	30	0	215	0	5419	5419	**2.71E+08**	60	0	558	0	20371	20371	**1.02E+09**
1	1390	452	15	3130	4520	**2.26E+08**	31	0	80	0	4347	4347	**2.17E+08**	61	0	1890	62	15079	18900	**9.45E+08**
2	0	62	0	1836	1836	**9.16E+07**	32	596	512	17	3618	5120	**2.56E+08**	62	122	502	16	4901	5023	**2.50E+08**
3	0	81	1	2808	2808	**1.40E+08**	33	26	403	10	4006	4032	**2.01E+08**	63	0	391	15	4846	4846	**2.42E+08**
4	233	816	27	7927	8160	**4.08E+08**	34	51	416	14	1705	4200	**2.10E+08**	64	1931	1306	43	11135	13066	**6.53E+08**
5	0	276	4	3692	3692	**1.84E+08**	35	56	142	4	1372	1428	**7.07E+07**	65	0	357	0	12721	12721	**6.36E+08**
6	34	190	6	1853	1900	**9.47E+07**	36	270	534	18	1593	5400	**2.70E+08**	66	0	597	20	6310	6310	**3.15E+08**
7	366	438	14	3975	4380	**2.18E+08**	37	0	48	0	14969	14969	**7.47E+08**	67	1840	482	16	2987	4827	**2.41E+08**
8	6	620	20	5342	6200	**3.09E+08**	38	0	281	9	3202	3202	**1.59E+08**	68	0	192	0	4850	4850	**2.42E+08**
9	0	33	0	1636	1636	**8.13E+07**	39	380	226	7	1879	2260	**1.13E+08**							
10	590	206	6	1477	2067	**1.03E+08**	40	0	167	1	3060	3060	**1.53E+08**							
11	98	143	4	1333	1431	**7.09E+07**	41	611	271	9	2102	2713	**1.36E+08**							
12	0	11	2	1632	1632	**8.12E+07**	42	0	72	2	1337	1337	**6.64E+07**							
13	0	24	0	4491	4491	**2.24E+08**	43	112	132	4	1212	1324	**6.58E+07**							
14	117	196	6	1824	1960	**9.75E+07**	44	0	196	6	2114	2114	**1.06E+08**							
15	0	90	4	1956	1956	**9.73E+07**	45	0	13	1	2782	2782	**1.39E+08**							
16	0	56	2	2181	2181	**1.09E+08**	46	787	427	14	3485	4272	**2.13E+08**							
17	0	61	0	1165	1165	**5.74E+07**	47	0	55	0	1439	1439	**7.12E+07**							
18	0	156	6	2036	2036	**1.01E+08**	48	0	574	20	6875	6875	**3.43E+08**							
19	349	263	8	2281	2630	**1.31E+08**	49	1432	1331	44	11883	13315	**6.65E+08**							
20	30	270	9	2679	2709	**1.35E+08**	50	0	580	0	11884	11884	**5.94E+08**							
21	0	60	1	5100	5100	**2.55E+08**	51	1	420	14	4204	4205	**2.10E+08**							
22	593	305	9	2457	3050	**1.52E+08**	52	0	641	0	9603	9603	**4.80E+08**							
23	0	106	11	5230	5230	**2.61E+08**	53	0	816	0	10817	10817	**5.41E+08**							
24	36	436	14	4325	4361	**2.18E+08**	54	0	885	0	10903	10903	**5.45E+08**							
25	0	112	0	4517	4517	**2.26E+08**	55	187	1052	35	10341	10528	**5.26E+08**							
26	458	241	8	1469	2410	**1.20E+08**	56	0	278	6	4578	4578	**2.29E+08**							
27	0	20	0	1496	1496	**7.39E+07**	57	0	466	0	7145	7145	**3.56E+08**							
28	133	263	8	1997	2630	**1.31E+08**	58	0	770	8	7700	7700	**3.84E+08**							
29	92	311	10	3026	3118	**1.55E+08**	59	4344	1798	59	13626	17980	**8.98E+08**							

Figure 1: Mapping Results

the area drops when the BRAM Size increases because the number of BRAMs in series and parallel required for the same logical ram is likely to be less when the physical depth and width of a BRAM increase, so that the logic blocks used for muxes will be fewer. But when BRAM Size is overly large, the wasted area will increase so that the overall area becomes larger.

The second trend observed is that Max Width and LBs / BRAM ratios minimizing average area increase when BRAM Size increases. LBs / BRAM ratio increases because the number of logic blocks that causes an extra BRAM should grow higher to maintain a low area allocation when BRAM size increases. Max Width also increases because the deep area in a physical ram can be a large waste if the same amount of area was not allocated to cover a wider space. The waste is higher when BRAM Size is larger.

BRAM Size	Max Width	LBs / BRAM	Geometric Average Area
1 kbit	4	1	2.498e8
2 kbit	8	2	2.273e8
4 kbit	16	4	2.172e8
8 kbit	32	6	2.143e8
16 kbit	32	8	2.216e8
32 kbit	64	10	2.471e8
64 kbit	64	20	2.873e8
128 kbit	128	32	3.581e8

Figure 2: One BRAM Type with No LUTRAM

Part (f) Area with LUTRAM and One BRAM Type

Same as Part (e), the Max Widths and LBs / BRAM ratios explored for different BRAM sizes are [1, 2 ,4, 8, 16, 32, 64, 128, 256, 512] and [1, 2, 4, 6, 8, 10, 20, 32, 40, 50, 80, 100, 128, 256, 300] respectively. See Appendix B for the top 20 lowest areas for BRAMs of sizes from 1k to 128k.

Figure 3 shows the Max Widths and LBs / BRAM ratios minimizing the geometric average areas for each BRAM of different sizes when 50% of all the logic blocks can be used as LUTRAMs.

Compared to part (e) where LUTRAM is not allowed, the lowest average areas reachable for BRAMs of different sizes are all lower. This is because the logic blocks coming in ratio with the BRAMs can be used as LUTRAMs to cover small logical rams, which saves much more area comparing to part (e) where small logical rams are covered by large BRAMs.

The trends of Max Width, LBs / BRAM ratios and average area changing with BRAM size are similar to part (e).

BRAM Size	Max Width	LBs / BRAM	Geometric Average Area
1 kbit	4	1	2.419e8
2 kbit	4	2	2.191e8
4 kbit	8	4	2.076e8
8 kbit	16	6	2.034e8
16 kbit	32	10	2.057e8
32 kbit	64	20	2.125e8
64 kbit	64	32	2.265e8
128 kbit	128	50	2.502e8

Figure 3: One BRAM Type with LUTRAM

Part (g) Best RAM blocks organization

Figure 4 shows the BRAM sizes, Max Widths and the LBs / BRAM ratios explored to find the best RAM blocks organization considering the results of Part(e) and Part(f). The percentage of LBs capable of being LUTRAMs varies from 10% to 100%.

All combinations of 3 types of BRAMs or 2 types of BRAMs plus LUTRAMs are checked. See Appendix C for the top 20 best organizations for 3 BRAMs and 2 BRAMs + LUTRAMs.

Figure 5 shows the best RAM blocks organization given the exploration ranges above. The geometric average area is **2.02467e08**. This organization is efficient first because the sizes of the two BRAMs used are moderate. Compared to smaller BRAMs, physical ram copies in series and parallel will be fewer so that the LBs for muxes will be fewer. Compared to larger

	RAM1	RAM2	RAM3	RAM4	RAM5	RAM6	RAM7	RAM8
BRAM Size	1024	2048	4096	8192	16384	32768	65536	131072
Max Width	4	8	16	32	32	64	64	128
LBs / BRAM	2	6	10	20	20	50	100	256

Figure 4: Best RAM Organization Exploration Range

BRAMs, less space will be wasted when mapping to small logical rams. Second, the LBs / RAM ratio is suitable. Compared to smaller ratios, the number of extra BRAMs needed due to the increase of LBs will be smaller. Compared to larger ratios, there will be fewer LBs wasted due to the increase of BRAMs. Third, the BRAMs can support a max width of 32 bits so that there are sufficient physical memory dimensions to handle logical rams of different shapes. Lastly, LBs can be used as LUTRAMs to map small logical rams to have less space wasted when the required LBs are more than the LBs used.

	8k BRAM	16k BRAM	LUTRAM (40% LBs capable)
Width Range	{1, 2, 4, 8, 16, 32}	{1, 2, 4, 8, 16, 32}	{10, 20}
LBs / RAM	20	20	1

Figure 5: Best RAM Organization

Reference

[1] "Next-fit bin packing," Wikipedia, 16-Oct-2021. [Online]. Available: https://en.wikipedia.org/wiki/Nex
fit_bin_packing. [Accessed: 25-Nov-2021].

Appendix A

```
import␣sys
import␣os
from␣os␣import␣path
import␣re
import␣math
import␣functools


NUM_CIRCUITS␣=␣69
NUM_DIFF_BRAM_SIZE␣=␣18␣#␣2^0,␣2^1,␣2^2,␣...␣,␣2^17,␣LB
```

```
LB = NUM_DIFF_BRAM_SIZE
STRATIX_IV = len(sys.argv)==1

class architecture_T():
    def __init__(self):
        self.maxWidthMp = {} # define the maximum width for each physical mem type
        self.ratioMp = {} # define the #physical mem type blocks / #LBs ratio
        self.typeMp = {} # define the type number for each physical mem type
        self.LUTRAMRatioLimit = 1.0 # Assume all the LBs can be used as LUTRAMs by defau
        if STRATIX_IV:
            self.maxWidthMp[LB] = -1 # LB --> It's a don't-care. Defined separately in t
            self.maxWidthMp[13] = 32 # 8k --> 32-bit word,
            self.maxWidthMp[17] = 128 # 128k --> 128-bit word
            self.ratioMp[LB] = 1 #LB --> 1:1.
            self.ratioMp[13] = 10 #8k --> 1:10
            self.ratioMp[17] = 300 #128k --> 1:300
            self.typeMp[LB] = 1
            self.typeMp[13] = 2
            self.typeMp[17] = 3
            self.LUTRAMRatioLimit = 0.5
        else:
            memTypes = [int(s) for s in sys.argv[1].split(',')]
            maxWidths = [int(s) for s in sys.argv[2].split(',')]
            ratios = [int(s) for s in sys.argv[3].split(',')]
            LUTRamRationLimit = float(sys.argv[4])
            for memTypeIdx in range(1,len(memTypes)+1):
                memType = memTypes[memTypeIdx-1]
                self.typeMp[memType] = memTypeIdx
            for memType, memTypeIdx in self.typeMp.items():
                self.maxWidthMp[memType] = maxWidths[memTypeIdx-1]
                self.ratioMp[memType] = ratios[memTypeIdx-1]
            self.LUTRAMRatioLimit = LUTRamRationLimit

class logicalRam_T():
    def __init__(self, ramID, mode, depth, width):
        self.ramID = ramID
        self.mode = mode
        self.depth = depth
        self.width = width
        self.size = depth * width
    def printLogicRam(self):
        print('RAM'+str(self.ramID))
```

```
        print('Mode: '+self.mode, 'Depth: '+str(self.depth), 'Width: '+str(self.width),

class physicalRam_T():
    def __init__(self, depth, width, sizeExp):
        self.depth = depth
        self.width = width
        self.sizeExp = sizeExp

class circuit_T():
    def __init__(self):
        self.logicalRams = []
        self.sortedLogicalRams = []
        self.numLB = 0
    def sortlogicalRams(self):
        def cmp(r1,r2):
            return -1 if r1.size>r2.size or (r1.size==r2.size and r1.depth>r2.depth) els
        self.sortedLogicalRams = sorted(self.logicalRams, key=functools.cmp_to_key(cmp))
    def printCircuit(self):
        for logicalRam in self.sortedLogicalRams:
            logicalRam.printLogicRam()
        print('Num logic blocks: ', self.numLB)

class mapping_T():
    def __init__(self, area, logicalRam, phyRam, s, p, numMuxes):
        self.area = area
        self.logicalRam = logicalRam
        self.phyRam = phyRam
        self.s = s
        self.p = p
        self.numMuxes = numMuxes


def genPhysicalRams(maxWidthMp):
    # generate different configurations for all choosen physical memories
    # group them as per their size
    # the last group are LUTRAMs
    physicalRams = [[] for i in range(NUM_DIFF_BRAM_SIZE+1)]
    for sizeExp in maxWidthMp:
        if sizeExp==LB:
            # append the two types of LUTRAMs
            physicalRams[sizeExp].append(physicalRam_T(64, 10, LB))
            physicalRams[sizeExp].append(physicalRam_T(32, 20, LB))
```

```
        else:
            # append all the configurations of BRAMs with the same size
            size = 1<<sizeExp
            for widthExp in range(NUM_DIFF_BRAM_SIZE):
                width = 1<<widthExp
                if width>maxWidthMp[sizeExp]: break
                depth = size//width
                physicalRams[sizeExp].append(physicalRam_T(depth, width, sizeExp))
    return physicalRams

def genCircuits(logicalRam_filePath, logicBlock_filePath):
    # get the circuits from the logical ram file
    circuits = [circuit_T() for i in range(NUM_CIRCUITS)]
    f = open(logicalRam_filePath, 'r')
    cnt = 0
    for line in f:
        if cnt>1:
            [circuitID,ramID,mode,depth,width] = line.replace('\n','',1).split()
            circuitID = int(circuitID)
            ramID = int(ramID)
            depth = int(depth)
            width = int(width)
            circuits[circuitID].logicalRams.append(logicalRam_T(ramID,mode,depth,width))
        cnt += 1
    f = open(logicBlock_filePath, 'r')
    cnt = 0
    for line in f:
        if cnt>0:
            [circuitID, numLB] = line.replace('\n','',1).split()
            numLB = int(numLB)
            circuitID = int(circuitID)
            circuits[circuitID].numLB = numLB
        cnt += 1

    # sort the logical rams
    for circuit in circuits:
        circuit.sortlogicalRams()
    return circuits

def getResourceUsage(phyRam, logicalRam):
    # get logical ram info
    ld = logicalRam.depth
```

```
    lw = logicalRam.width
    mode = logicalRam.mode
    # compute resource usage
    d = phyRam.depth
    w = phyRam.width
    s = (ld-1)//d+1
    p = (lw-1)//w+1

    if s>16:
        return False, -1, -1, -1 # Too many rams in series, too slow...
    numMuxes = ((s if s>2 else s-1) if mode!='ROM' else 0) + (0 if s==1 else (lw * (1 if
    numMuxes = 2*numMuxes if mode=='TrueDualPort' else numMuxes
    return True, s, p, numMuxes

def getArea(numPhyRams, phyRam, numMuxes, maxWidthMp):
    area = 0
    if phyRam.sizeExp != LB:
        bits = 1<<phyRam.sizeExp
        area += numPhyRams * (9000 + 5 * bits + 90 * math.sqrt(bits) + 1200 * maxWidthMp
    else:
        area += numPhyRams * 40000
    area += numMuxes/10.0 * 35000
    return area

def genCircuitMapping(generalSetup, circuit, physicalRams, phyID):

    # Keep track of the resources available or used
    srcMp = {}
    sinkMp = {}
    srcMp[LB] = circuit.numLB
    sinkMp[LB] = (circuit.numLB, 0.0)
    for phyRamType in generalSetup.typeMp:
        if phyRamType==LB: continue
        srcMp[phyRamType] = circuit.numLB // generalSetup.ratioMp[phyRamType]
        sinkMp[phyRamType] = 0.0

    # Generate mappings for each logical ram
    logicalRams = circuit.sortedLogicalRams
    mappings = [''] * len(logicalRams)
    for ramIdx in range(len(logicalRams)):

        # Get logical ram info
```

```
        logicalRam = logicalRams[ramIdx]
        mode = logicalRam.mode

        # Check if the logical ram can fit into the current "bin"
        possibleMappings = []
        for phyRamType in generalSetup.typeMp:
            if mode=='TrueDualPort' and phyRamType==LB: continue # Can't implement TrueD
            sameTypePhyRams = physicalRams[phyRamType]
            for i in range(len(sameTypePhyRams)):
                if mode=='TrueDualPort' and i==len(sameTypePhyRams)-1: continue # Can't

                # Compute resource usage
                phyRam = sameTypePhyRams[i]
                isLegalMapping, s, p, numMuxes = getResourceUsage(phyRam, logicalRam)
                if not isLegalMapping: continue

                # Compute area
                area = getArea(s*p, phyRam, numMuxes, generalSetup.maxWidthMp)
                possibleMappings.append(mapping_T(area,logicalRam,phyRam,s,p,numMuxes))

        # Rank the resource usage in ascending area order
        possibleMappings = sorted(possibleMappings, key = lambda x : x.area)
        if len(possibleMappings)==0:
            print('No legal mapping is found')
            sys.exit()

        # Iterate through the mappings in ascending area order
        # Check if there is a mapping that can fit in the current "bin"
        canFitCurrentBin = False
        possibleUpdatedSrcSinkMps = []
        chosenMapping = ''
        for mapping in possibleMappings:

            # Additional physical resources required
            phyRamType = mapping.phyRam.sizeExp
            numPhyRam = mapping.s * mapping.p
            numLB = (mapping.numMuxes / 10.0)

            # Calculate number of logic blocks and physical ram blocks required
            numLBUsed = sinkMp[LB][0]
            numLUTRAMUsed = sinkMp[LB][1]
            if phyRamType==LB:
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣minRequiredLB␣=␣((numLUTRAMUsed␣+␣numPhyRam)␣/␣generalSetup.LUTRAMRatioL
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣actualRequiredLB␣=␣numLBUsed␣+␣numLB␣+␣numLUTRAMUsed␣+␣numPhyRam
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣requiredLB␣=␣max(minRequiredLB,␣actualRequiredLB)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣availLB␣=␣srcMp[LB]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣availLB␣>=␣requiredLB:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣canFitCurrentBin␣=␣True
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sinkMp[LB]␣=␣(numLBUsed␣+␣numLB,␣numLUTRAMUsed␣+␣numPhyRam)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣chosenMapping␣=␣mapping
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣break
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#␣Get␣the␣updated␣sinkMp
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣updSinkMp␣=␣{}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣for␣srcType␣in␣sinkMp:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣srcType==LB:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣updSinkMp[srcType]␣=␣(numLBUsed␣+␣numLB,␣numLUTRAMUsed␣+␣numPhyR
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣updSinkMp[srcType]␣=␣sinkMp[srcType]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#␣Get␣the␣updated␣srcMp␣with␣extra␣LBs␣added
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣updSrcLB␣=␣requiredLB
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣updSrcMp␣=␣{}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣for␣srcType␣in␣srcMp:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣updSrcMp[srcType]␣=␣max(srcMp[srcType],␣updSrcLB␣//␣(generalSetup.ra
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#␣Calculate␣number␣of␣extra␣logic␣blocks␣needed
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣extraLB␣=␣updSrcLB␣-␣availLB
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣possibleUpdatedSrcSinkMps.append((extraLB,␣updSrcMp,␣updSinkMp,␣mapping)
␣␣␣␣␣␣␣␣␣␣␣␣else:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣minRequiredLB␣=␣(numLUTRAMUsed␣/␣generalSetup.LUTRAMRatioLimit)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣actualRequiredLB␣=␣numLBUsed␣+␣numLB␣+␣numLUTRAMUsed
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣requiredLB␣=␣max(minRequiredLB,␣actualRequiredLB)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣availLB␣=␣srcMp[LB]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣requiredBRAM␣=␣sinkMp[phyRamType]␣+␣numPhyRam
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣availBRAM␣=␣srcMp[phyRamType]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣availLB␣>=␣requiredLB␣and␣availBRAM␣>=␣requiredBRAM:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣canFitCurrentBin␣=␣True
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sinkMp[LB]␣=␣(numLBUsed␣+␣numLB,␣numLUTRAMUsed)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sinkMp[phyRamType]␣=␣requiredBRAM
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣chosenMapping␣=␣mapping
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣break
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#␣Get␣the␣updated␣sinkMp
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣updSinkMp␣=␣{}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣for␣srcType␣in␣sinkMp:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣srcType==LB:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣updSinkMp[srcType]␣=␣(numLBUsed␣+␣numLB,␣numLUTRAMUsed)
```

```
                    elif srcType==phyRamType:
                        updSinkMp[srcType] = requiredBRAM
                    else:
                        updSinkMp[srcType] = sinkMp[srcType]
                # Get the updated srcMp
                # Look at logic blocks first
                updSrcLB = requiredLB
                updSrcMp = {}
                for srcType in srcMp:
                    updSrcMp[srcType] = max(srcMp[srcType], updSrcLB // (generalSetup.ra
                # Then look at BRAMs
                updSrcBRAM = requiredBRAM
                updSrcLB = updSrcBRAM * generalSetup.ratioMp[phyRamType]
                for srcType in srcMp:
                    updSrcMp[srcType] = max(updSrcMp[srcType], updSrcLB // (generalSetup
                # Calculate number of extra logic blocks needed
                extraLB = updSrcMp[LB] - availLB
                possibleUpdatedSrcSinkMps.append((extraLB, updSrcMp, updSinkMp, mapping)

        if not canFitCurrentBin:
            # If the current 'bin' can't have the logical ram fit in
            # Iterate through all the possbile mappings again. This time extra physical
            possibleUpdatedSrcSinkMps = sorted(possibleUpdatedSrcSinkMps, key=lambda x:
            srcMp = possibleUpdatedSrcSinkMps[0][1]
            sinkMp = possibleUpdatedSrcSinkMps[0][2]
            chosenMapping = possibleUpdatedSrcSinkMps[0][3]

        mappings[ramIdx] = (phyID, chosenMapping)
        phyID += 1

    return phyID, mappings

def genFile(mappings, fileName):
    f = open(fileName, 'w')
    for mapping in mappings:
        f.write(mapping+'\n')
    f.close()

def genSolution():

    # Initialize the architecture setup
    generalSetup = architecture_T()
```

```
    # generate all physical RAMs available
    physicalRams = genPhysicalRams(generalSetup.maxWidthMp)
    # generate all circuits
    circuits = genCircuits('logical_rams.txt', 'logic_block_count.txt');

    mappings = []
    phyID = 0
    for circuitID in range(len(circuits)):
        circuit = circuits[circuitID]
        phyID, oneCircuitMappings = genCircuitMapping(generalSetup, circuit, physicalRam
        # basic mapping format ...
        for mappingPair in oneCircuitMappings:
            pId = mappingPair[0]
            mapping = mappingPair[1]
            mappings.append(' '.join([
                str(circuitID),
                str(mapping.logicalRam.ramID),
                str(mapping.numMuxes),
                "LW "+str(mapping.logicalRam.width),
                "LD "+str(mapping.logicalRam.depth),
                "ID "+str(pId),
                "S "+str(mapping.s),
                "P "+str(mapping.p),
                "Type "+ str(generalSetup.typeMp[mapping.phyRam.sizeExp]),
                "Mode "+mapping.logicalRam.mode,
                "W "+str(mapping.phyRam.width),
                "D "+str(mapping.phyRam.depth)
            ]))

    genFile(mappings, 'basic.txt')

import time
startTime = time.time()
genSolution()
endTime = time.time()
print('CPU runtime(sec) is: ', endTime-startTime)
```

Appendix B

Top 20 lowest Area with No LUTRAM and One BRAM Type

See Figures 6-8

1	BRAMSize=1024	MaxWidth=4	LBs/BRAM=1	2.50E+08	BRAMSize=2048	MaxWidth=8	LBs/BRAM=2	2.27E+08	BRAMSize=4096	MaxWidth=16	LBs/BRAM=4	2.17E+08
2	BRAMSize=1024	MaxWidth=8	LBs/BRAM=1	2.57E+08	BRAMSize=2048	MaxWidth=16	LBs/BRAM=2	2.35E+08	BRAMSize=4096	MaxWidth=16	LBs/BRAM=3	2.17E+08
3	BRAMSize=1024	MaxWidth=8	LBs/BRAM=2	2.67E+08	BRAMSize=2048	MaxWidth=16	LBs/BRAM=3	2.38E+08	BRAMSize=4096	MaxWidth=32	LBs/BRAM=4	2.25E+08
4	BRAMSize=1024	MaxWidth=16	LBs/BRAM=2	2.80E+08	BRAMSize=2048	MaxWidth=8	LBs/BRAM=3	2.47E+08	BRAMSize=4096	MaxWidth=32	LBs/BRAM=5	2.27E+08
5	BRAMSize=1024	MaxWidth=2	LBs/BRAM=1	2.80E+08	BRAMSize=2048	MaxWidth=4	LBs/BRAM=1	2.51E+08	BRAMSize=4096	MaxWidth=8	LBs/BRAM=3	2.28E+08
6	BRAMSize=1024	MaxWidth=4	LBs/BRAM=2	2.89E+08	BRAMSize=2048	MaxWidth=4	LBs/BRAM=2	2.54E+08	BRAMSize=4096	MaxWidth=16	LBs/BRAM=5	2.28E+08
7	BRAMSize=1024	MaxWidth=16	LBs/BRAM=1	2.92E+08	BRAMSize=2048	MaxWidth=16	LBs/BRAM=4	2.55E+08	BRAMSize=4096	MaxWidth=8	LBs/BRAM=2	2.30E+08
8	BRAMSize=1024	MaxWidth=8	LBs/BRAM=3	3.04E+08	BRAMSize=2048	MaxWidth=8	LBs/BRAM=1	2.58E+08	BRAMSize=4096	MaxWidth=32	LBs/BRAM=6	2.35E+08
9	BRAMSize=1024	MaxWidth=16	LBs/BRAM=3	3.06E+08	BRAMSize=2048	MaxWidth=32	LBs/BRAM=3	2.60E+08	BRAMSize=4096	MaxWidth=32	LBs/BRAM=3	2.35E+08
10	BRAMSize=1024	MaxWidth=32	LBs/BRAM=2	3.25E+08	BRAMSize=2048	MaxWidth=32	LBs/BRAM=4	2.70E+08	BRAMSize=4096	MaxWidth=16	LBs/BRAM=2	2.36E+08
11	BRAMSize=1024	MaxWidth=16	LBs/BRAM=4	3.38E+08	BRAMSize=2048	MaxWidth=32	LBs/BRAM=2	2.70E+08	BRAMSize=4096	MaxWidth=16	LBs/BRAM=6	2.41E+08
12	BRAMSize=1024	MaxWidth=32	LBs/BRAM=3	3.39E+08	BRAMSize=2048	MaxWidth=16	LBs/BRAM=5	2.75E+08	BRAMSize=4096	MaxWidth=8	LBs/BRAM=4	2.43E+08
13	BRAMSize=1024	MaxWidth=8	LBs/BRAM=4	3.46E+08	BRAMSize=2048	MaxWidth=8	LBs/BRAM=4	2.75E+08	BRAMSize=4096	MaxWidth=32	LBs/BRAM=7	2.44E+08
14	BRAMSize=1024	MaxWidth=4	LBs/BRAM=3	3.51E+08	BRAMSize=2048	MaxWidth=32	LBs/BRAM=5	2.83E+08	BRAMSize=4096	MaxWidth=32	LBs/BRAM=8	2.54E+08
15	BRAMSize=1024	MaxWidth=32	LBs/BRAM=4	3.63E+08	BRAMSize=2048	MaxWidth=2	LBs/BRAM=1	2.84E+08	BRAMSize=4096	MaxWidth=64	LBs/BRAM=5	2.56E+08
16	BRAMSize=1024	MaxWidth=32	LBs/BRAM=1	3.68E+08	BRAMSize=2048	MaxWidth=16	LBs/BRAM=1	2.91E+08	BRAMSize=4096	MaxWidth=16	LBs/BRAM=7	2.56E+08
17	BRAMSize=1024	MaxWidth=2	LBs/BRAM=2	3.69E+08	BRAMSize=2048	MaxWidth=16	LBs/BRAM=6	2.96E+08	BRAMSize=4096	MaxWidth=64	LBs/BRAM=6	2.58E+08
18	BRAMSize=1024	MaxWidth=16	LBs/BRAM=5	3.72E+08	BRAMSize=2048	MaxWidth=32	LBs/BRAM=6	2.99E+08	BRAMSize=4096	MaxWidth=4	LBs/BRAM=2	2.59E+08
19	BRAMSize=1024	MaxWidth=32	LBs/BRAM=5	3.91E+08	BRAMSize=2048	MaxWidth=4	LBs/BRAM=3	3.00E+08	BRAMSize=4096	MaxWidth=64	LBs/BRAM=4	2.61E+08
20	BRAMSize=1024	MaxWidth=8	LBs/BRAM=5	3.94E+08	BRAMSize=2048	MaxWidth=8	LBs/BRAM=5	3.07E+08	BRAMSize=4096	MaxWidth=64	LBs/BRAM=7	2.64E+08

Figure 6: 1k - 4k

1	BRAMSize=8192	MaxWidth=32	LBs/BRAM=6	2.14E+08	BRAMSize=16384	MaxWidth=32	LBs/BRAM=8	2.22E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=10	2.47E+08
2	BRAMSize=8192	MaxWidth=32	LBs/BRAM=7	2.15E+08	BRAMSize=16384	MaxWidth=64	LBs/BRAM=10	2.26E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=10	2.50E+08
3	BRAMSize=8192	MaxWidth=32	LBs/BRAM=5	2.17E+08	BRAMSize=16384	MaxWidth=32	LBs/BRAM=10	2.26E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=8	2.54E+08
4	BRAMSize=8192	MaxWidth=16	LBs/BRAM=5	2.18E+08	BRAMSize=16384	MaxWidth=32	LBs/BRAM=6	2.27E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=20	2.58E+08
5	BRAMSize=8192	MaxWidth=32	LBs/BRAM=8	2.20E+08	BRAMSize=16384	MaxWidth=64	LBs/BRAM=8	2.30E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=20	2.63E+08
6	BRAMSize=8192	MaxWidth=16	LBs/BRAM=4	2.20E+08	BRAMSize=16384	MaxWidth=16	LBs/BRAM=6	2.39E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=8	2.64E+08
7	BRAMSize=8192	MaxWidth=16	LBs/BRAM=6	2.24E+08	BRAMSize=16384	MaxWidth=64	LBs/BRAM=6	2.48E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=10	2.75E+08
8	BRAMSize=8192	MaxWidth=32	LBs/BRAM=4	2.26E+08	BRAMSize=16384	MaxWidth=16	LBs/BRAM=8	2.49E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=6	2.77E+08
9	BRAMSize=8192	MaxWidth=32	LBs/BRAM=9	2.27E+08	BRAMSize=16384	MaxWidth=16	LBs/BRAM=4	2.52E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=8	2.88E+08
10	BRAMSize=8192	MaxWidth=64	LBs/BRAM=8	2.29E+08	BRAMSize=16384	MaxWidth=128	LBs/BRAM=10	2.54E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=6	2.91E+08
11	BRAMSize=8192	MaxWidth=64	LBs/BRAM=7	2.30E+08	BRAMSize=16384	MaxWidth=32	LBs/BRAM=4	2.60E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=20	2.94E+08
12	BRAMSize=8192	MaxWidth=16	LBs/BRAM=3	2.31E+08	BRAMSize=16384	MaxWidth=64	LBs/BRAM=20	2.66E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=20	2.97E+08
13	BRAMSize=8192	MaxWidth=64	LBs/BRAM=9	2.32E+08	BRAMSize=16384	MaxWidth=128	LBs/BRAM=8	2.67E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=6	2.98E+08
14	BRAMSize=8192	MaxWidth=16	LBs/BRAM=7	2.33E+08	BRAMSize=16384	MaxWidth=16	LBs/BRAM=10	2.68E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=8	3.00E+08
15	BRAMSize=8192	MaxWidth=64	LBs/BRAM=6	2.34E+08	BRAMSize=16384	MaxWidth=128	LBs/BRAM=20	2.74E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=10	3.00E+08
16	BRAMSize=8192	MaxWidth=32	LBs/BRAM=10	2.36E+08	BRAMSize=16384	MaxWidth=8	LBs/BRAM=4	2.86E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=32	3.02E+08
17	BRAMSize=8192	MaxWidth=64	LBs/BRAM=10	2.37E+08	BRAMSize=16384	MaxWidth=64	LBs/BRAM=4	2.94E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=32	3.15E+08
18	BRAMSize=8192	MaxWidth=64	LBs/BRAM=5	2.43E+08	BRAMSize=16384	MaxWidth=32	LBs/BRAM=20	2.95E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=32	3.29E+08
19	BRAMSize=8192	MaxWidth=8	LBs/BRAM=3	2.44E+08	BRAMSize=16384	MaxWidth=128	LBs/BRAM=6	2.95E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=4	3.32E+08
20	BRAMSize=8192	MaxWidth=16	LBs/BRAM=8	2.44E+08	BRAMSize=16384	MaxWidth=8	LBs/BRAM=6	3.06E+08	BRAMSize=32768	MaxWidth	LBs/BRAM=10	3.34E+08

Figure 7: 8k - 32k

Top 20 lowest Area with LUTRAM and One BRAM Type

See Figures 9-11

BRAMSize=65536	MaxWidth=64	LBs/BRAM=20	2.87E+08		BRAMSize=131072	MaxWidth=128	LBs/BRAM=32	3.58E+08
BRAMSize=65536	MaxWidth=128	LBs/BRAM=20	2.88E+08		BRAMSize=131072	MaxWidth=128	LBs/BRAM=20	3.59E+08
BRAMSize=65536	MaxWidth=128	LBs/BRAM=32	3.10E+08		BRAMSize=131072	MaxWidth=64	LBs/BRAM=20	3.63E+08
BRAMSize=65536	MaxWidth=64	LBs/BRAM=10	3.11E+08		BRAMSize=131072	MaxWidth=256	LBs/BRAM=32	3.77E+08
BRAMSize=65536	MaxWidth=32	LBs/BRAM=10	3.14E+08		BRAMSize=131072	MaxWidth=128	LBs/BRAM=40	3.78E+08
BRAMSize=65536	MaxWidth=256	LBs/BRAM=20	3.20E+08		BRAMSize=131072	MaxWidth=64	LBs/BRAM=32	3.83E+08
BRAMSize=65536	MaxWidth=64	LBs/BRAM=32	3.27E+08		BRAMSize=131072	MaxWidth=256	LBs/BRAM=20	3.89E+08
BRAMSize=65536	MaxWidth=32	LBs/BRAM=20	3.31E+08		BRAMSize=131072	MaxWidth=256	LBs/BRAM=40	3.93E+08
BRAMSize=65536	MaxWidth=256	LBs/BRAM=32	3.32E+08		BRAMSize=131072	MaxWidth=128	LBs/BRAM=50	4.09E+08
BRAMSize=65536	MaxWidth=32	LBs/BRAM=8	3.33E+08		BRAMSize=131072	MaxWidth=64	LBs/BRAM=40	4.13E+08
BRAMSize=65536	MaxWidth=128	LBs/BRAM=40	3.37E+08		BRAMSize=131072	MaxWidth=256	LBs/BRAM=50	4.20E+08
BRAMSize=65536	MaxWidth=128	LBs/BRAM=10	3.39E+08		BRAMSize=131072	MaxWidth=32	LBs/BRAM=20	4.24E+08
BRAMSize=65536	MaxWidth=64	LBs/BRAM=8	3.44E+08		BRAMSize=131072	MaxWidth=512	LBs/BRAM=32	4.29E+08
BRAMSize=65536	MaxWidth=256	LBs/BRAM=40	3.56E+08		BRAMSize=131072	MaxWidth=512	LBs/BRAM=40	4.40E+08
BRAMSize=65536	MaxWidth=64	LBs/BRAM=40	3.64E+08		BRAMSize=131072	MaxWidth=64	LBs/BRAM=10	4.42E+08
BRAMSize=65536	MaxWidth=128	LBs/BRAM=50	3.75E+08		BRAMSize=131072	MaxWidth=32	LBs/BRAM=10	4.51E+08
BRAMSize=65536	MaxWidth=128	LBs/BRAM=8	3.79E+08		BRAMSize=131072	MaxWidth=64	LBs/BRAM=50	4.53E+08
BRAMSize=65536	MaxWidth=32	LBs/BRAM=6	3.82E+08		BRAMSize=131072	MaxWidth=512	LBs/BRAM=20	4.58E+08
BRAMSize=65536	MaxWidth=16	LBs/BRAM=8	3.83E+08		BRAMSize=131072	MaxWidth=512	LBs/BRAM=50	4.62E+08
BRAMSize=65536	MaxWidth=16	LBs/BRAM=10	3.83E+08		BRAMSize=131072	MaxWidth=128	LBs/BRAM=10	4.69E+08

Figure 8: 64k - 128k

1	memType=10	maxWidth=4	ratio=1	2.42E+08	memType=11	maxWidth=4	ratio=2	2.19E+08	memType=12	maxWidth=8	ratio=4	2.08E+08	memType=13	maxWidth=16	ratio=6	2.03E+08
2	memType=10	maxWidth=2	ratio=1	2.42E+08	memType=11	maxWidth=8	ratio=2	2.19E+08	memType=12	maxWidth=16	ratio=4	2.08E+08	memType=13	maxWidth=16	ratio=8	2.04E+08
3	memType=10	maxWidth=4	ratio=2	2.44E+08	memType=11	maxWidth=8	ratio=3	2.20E+08	memType=12	maxWidth=16	ratio=5	2.08E+08	memType=13	maxWidth=32	ratio=8	2.04E+08
4	memType=10	maxWidth=8	ratio=2	2.47E+08	memType=11	maxWidth=4	ratio=3	2.25E+08	memType=12	maxWidth=8	ratio=3	2.10E+08	memType=13	maxWidth=32	ratio=10	2.07E+08
5	memType=10	maxWidth=2	ratio=2	2.56E+08	memType=11	maxWidth=16	ratio=3	2.28E+08	memType=12	maxWidth=8	ratio=5	2.11E+08	memType=13	maxWidth=16	ratio=10	2.08E+08
6	memType=10	maxWidth=8	ratio=1	2.59E+08	memType=11	maxWidth=8	ratio=4	2.31E+08	memType=12	maxWidth=16	ratio=6	2.12E+08	memType=13	maxWidth=32	ratio=6	2.10E+08
7	memType=10	maxWidth=8	ratio=3	2.69E+08	memType=11	maxWidth=2	ratio=2	2.35E+08	memType=12	maxWidth=16	ratio=3	2.16E+08	memType=13	maxWidth=8	ratio=6	2.12E+08
8	memType=10	maxWidth=16	ratio=2	2.69E+08	memType=11	maxWidth=16	ratio=4	2.35E+08	memType=12	maxWidth=8	ratio=6	2.17E+08	memType=13	maxWidth=16	ratio=4	2.15E+08
9	memType=10	maxWidth=4	ratio=3	2.71E+08	memType=11	maxWidth=16	ratio=2	2.36E+08	memType=12	maxWidth=16	ratio=7	2.18E+08	memType=13	maxWidth=8	ratio=8	2.16E+08
10	memType=10	maxWidth=16	ratio=3	2.84E+08	memType=11	maxWidth=4	ratio=4	2.42E+08	memType=12	maxWidth=4	ratio=3	2.20E+08	memType=13	maxWidth=8	ratio=4	2.16E+08
11	memType=10	maxWidth=2	ratio=3	2.92E+08	memType=11	maxWidth=8	ratio=5	2.45E+08	memType=12	maxWidth=32	ratio=5	2.20E+08	memType=13	maxWidth=64	ratio=10	2.19E+08
12	memType=10	maxWidth=16	ratio=1	2.96E+08	memType=11	maxWidth=16	ratio=5	2.46E+08	memType=12	maxWidth=32	ratio=6	2.21E+08	memType=13	maxWidth=64	ratio=8	2.21E+08
13	memType=10	maxWidth=8	ratio=4	2.96E+08	memType=11	maxWidth=2	ratio=3	2.48E+08	memType=12	maxWidth=4	ratio=4	2.22E+08	memType=13	maxWidth=8	ratio=10	2.24E+08
14	memType=10	maxWidth=4	ratio=4	3.04E+08	memType=11	maxWidth=4	ratio=1	2.49E+08	memType=12	maxWidth=32	ratio=4	2.24E+08	memType=13	maxWidth=32	ratio=4	2.31E+08
15	memType=10	maxWidth=16	ratio=4	3.07E+08	memType=11	maxWidth=2	ratio=1	2.50E+08	memType=12	maxWidth=32	ratio=7	2.25E+08	memType=13	maxWidth=64	ratio=6	2.33E+08
16	memType=10	maxWidth=32	ratio=2	3.14E+08	memType=11	maxWidth=32	ratio=3	2.55E+08	memType=12	maxWidth=16	ratio=8	2.25E+08	memType=13	maxWidth=4	ratio=4	2.35E+08
17	memType=10	maxWidth=32	ratio=3	3.19E+08	memType=11	maxWidth=32	ratio=4	2.57E+08	memType=12	maxWidth=8	ratio=7	2.26E+08	memType=13	maxWidth=4	ratio=6	2.35E+08
18	memType=10	maxWidth=8	ratio=5	3.23E+08	memType=11	maxWidth=16	ratio=6	2.59E+08	memType=12	maxWidth=8	ratio=2	2.27E+08	memType=13	maxWidth=32	ratio=20	2.43E+08
19	memType=10	maxWidth=2	ratio=4	3.31E+08	memType=11	maxWidth=4	ratio=5	2.60E+08	memType=12	maxWidth=4	ratio=2	2.28E+08	memType=13	maxWidth=4	ratio=8	2.44E+08
20	memType=10	maxWidth=16	ratio=5	3.31E+08	memType=11	maxWidth=8	ratio=6	2.61E+08	memType=12	maxWidth=4	ratio=5	2.29E+08	memType=13	maxWidth=64	ratio=20	2.49E+08

Figure 9: 1k – 4k, with 50% LBs supporting LUTRAM

Appendix C

Top 20 best organizations for 3 BRAMs

See Figure 12.

1	memType=13	maxWidth=16	ratio=6	2.03E+08	memType=14	maxWidth=32	ratio=10	2.06E+08	memType=15	maxWidth=64	ratio=20	2.13E+08
2	memType=13	maxWidth=16	ratio=8	2.04E+08	memType=14	maxWidth=16	ratio=10	2.10E+08	memType=15	maxWidth=32	ratio=20	2.14E+08
3	memType=13	maxWidth=32	ratio=8	2.04E+08	memType=14	maxWidth=32	ratio=8	2.11E+08	memType=15	maxWidth=64	ratio=32	2.16E+08
4	memType=13	maxWidth=32	ratio=10	2.07E+08	memType=14	maxWidth=32	ratio=20	2.12E+08	memType=15	maxWidth=128	ratio=32	2.21E+08
5	memType=13	maxWidth=16	ratio=10	2.08E+08	memType=14	maxWidth=16	ratio=8	2.12E+08	memType=15	maxWidth=32	ratio=32	2.21E+08
6	memType=13	maxWidth=32	ratio=6	2.10E+08	memType=14	maxWidth=64	ratio=20	2.13E+08	memType=15	maxWidth=128	ratio=20	2.23E+08
7	memType=13	maxWidth=8	ratio=6	2.12E+08	memType=14	maxWidth=64	ratio=10	2.16E+08	memType=15	maxWidth=64	ratio=40	2.24E+08
8	memType=13	maxWidth=16	ratio=4	2.15E+08	memType=14	maxWidth=16	ratio=6	2.21E+08	memType=15	maxWidth=128	ratio=40	2.27E+08
9	memType=13	maxWidth=8	ratio=8	2.16E+08	memType=14	maxWidth=16	ratio=20	2.24E+08	memType=15	maxWidth=16	ratio=20	2.29E+08
10	memType=13	maxWidth=8	ratio=4	2.16E+08	memType=14	maxWidth=128	ratio=20	2.26E+08	memType=15	maxWidth=32	ratio=10	2.30E+08
11	memType=13	maxWidth=64	ratio=10	2.19E+08	memType=14	maxWidth=32	ratio=6	2.27E+08	memType=15	maxWidth=32	ratio=40	2.31E+08
12	memType=13	maxWidth=64	ratio=8	2.21E+08	memType=14	maxWidth=64	ratio=8	2.27E+08	memType=15	maxWidth=64	ratio=50	2.36E+08
13	memType=13	maxWidth=8	ratio=10	2.24E+08	memType=14	maxWidth=8	ratio=8	2.29E+08	memType=15	maxWidth=16	ratio=10	2.37E+08
14	memType=13	maxWidth=32	ratio=4	2.31E+08	memType=14	maxWidth=8	ratio=10	2.29E+08	memType=15	maxWidth=128	ratio=50	2.38E+08
15	memType=13	maxWidth=64	ratio=6	2.33E+08	memType=14	maxWidth=8	ratio=6	2.33E+08	memType=15	maxWidth=64	ratio=10	2.40E+08
16	memType=13	maxWidth=4	ratio=4	2.35E+08	memType=14	maxWidth=64	ratio=32	2.37E+08	memType=15	maxWidth=256	ratio=32	2.42E+08
17	memType=13	maxWidth=4	ratio=6	2.35E+08	memType=14	maxWidth=32	ratio=32	2.40E+08	memType=15	maxWidth=16	ratio=32	2.44E+08
18	memType=13	maxWidth=32	ratio=20	2.43E+08	memType=14	maxWidth=128	ratio=32	2.45E+08	memType=15	maxWidth=256	ratio=40	2.45E+08
19	memType=13	maxWidth=4	ratio=8	2.44E+08	memType=14	maxWidth=128	ratio=10	2.45E+08	memType=15	maxWidth=32	ratio=8	2.47E+08
20	memType=13	maxWidth=64	ratio=20	2.49E+08	memType=14	maxWidth=64	ratio=6	2.50E+08	memType=15	maxWidth=32	ratio=50	2.48E+08

Figure 10: 8k – 32k, with 50% LBs supporting LUTRAM

memType=16	maxWidth=64	ratio=32	2.27E+08	memType=17	maxWidth=128	ratio=50	2.50E+08
memType=16	maxWidth=64	ratio=40	2.27E+08	memType=17	maxWidth=64	ratio=50	2.53E+08
memType=16	maxWidth=128	ratio=40	2.28E+08	memType=17	maxWidth=128	ratio=80	2.53E+08
memType=16	maxWidth=128	ratio=32	2.30E+08	memType=17	maxWidth=128	ratio=40	2.56E+08
memType=16	maxWidth=128	ratio=50	2.30E+08	memType=17	maxWidth=64	ratio=40	2.56E+08
memType=16	maxWidth=64	ratio=50	2.31E+08	memType=17	maxWidth=64	ratio=80	2.59E+08
memType=16	maxWidth=32	ratio=32	2.35E+08	memType=17	maxWidth=128	ratio=100	2.59E+08
memType=16	maxWidth=32	ratio=40	2.37E+08	memType=17	maxWidth=256	ratio=80	2.61E+08
memType=16	maxWidth=64	ratio=20	2.38E+08	memType=17	maxWidth=256	ratio=50	2.62E+08
memType=16	maxWidth=32	ratio=20	2.42E+08	memType=17	maxWidth=256	ratio=100	2.66E+08
memType=16	maxWidth=32	ratio=50	2.44E+08	memType=17	maxWidth=64	ratio=32	2.66E+08
memType=16	maxWidth=256	ratio=50	2.44E+08	memType=17	maxWidth=128	ratio=32	2.66E+08
memType=16	maxWidth=256	ratio=40	2.44E+08	memType=17	maxWidth=64	ratio=100	2.66E+08
memType=16	maxWidth=128	ratio=80	2.47E+08	memType=17	maxWidth=128	ratio=128	2.70E+08
memType=16	maxWidth=128	ratio=20	2.47E+08	memType=17	maxWidth=256	ratio=40	2.71E+08
memType=16	maxWidth=256	ratio=32	2.49E+08	memType=17	maxWidth=32	ratio=50	2.71E+08
memType=16	maxWidth=64	ratio=80	2.51E+08	memType=17	maxWidth=32	ratio=40	2.72E+08
memType=16	maxWidth=256	ratio=80	2.57E+08	memType=17	maxWidth=256	ratio=128	2.75E+08
memType=16	maxWidth=16	ratio=32	2.63E+08	memType=17	maxWidth=32	ratio=32	2.79E+08
memType=16	maxWidth=16	ratio=20	2.63E+08	memType=17	maxWidth=64	ratio=128	2.79E+08

Figure 11: 64k – 128k, with 50% LBs supporting LUTRAM

Top 20 best organizations for 2 BRAMs + LUTRAM

See Figure 13.

1	t1=14	t2=13	t3=12	mw1=32	mw2=32	mw3=16	r1=20	r2=20	r3=10	LUTlimit=0.1	2.13E+08
2	t1=14	t2=13	t3=12	mw1=32	mw2=32	mw3=16	r1=20	r2=20	r3=10	LUTlimit=0.2	2.13E+08
3	t1=14	t2=13	t3=12	mw1=32	mw2=32	mw3=16	r1=20	r2=20	r3=10	LUTlimit=0.3	2.13E+08
4	t1=14	t2=13	t3=12	mw1=32	mw2=32	mw3=16	r1=20	r2=20	r3=10	LUTlimit=0.4	2.13E+08
5	t1=14	t2=13	t3=12	mw1=32	mw2=32	mw3=16	r1=20	r2=20	r3=10	LUTlimit=0.5	2.13E+08
6	t1=14	t2=13	t3=12	mw1=32	mw2=32	mw3=16	r1=20	r2=20	r3=10	LUTlimit=0.6	2.13E+08
7	t1=14	t2=13	t3=12	mw1=32	mw2=32	mw3=16	r1=20	r2=20	r3=10	LUTlimit=0.7	2.13E+08
8	t1=14	t2=13	t3=12	mw1=32	mw2=32	mw3=16	r1=20	r2=20	r3=10	LUTlimit=0.8	2.13E+08
9	t1=14	t2=13	t3=12	mw1=32	mw2=32	mw3=16	r1=20	r2=20	r3=10	LUTlimit=0.9	2.13E+08
10	t1=14	t2=13	t3=12	mw1=32	mw2=32	mw3=16	r1=20	r2=20	r3=10	LUTlimit=1.0	2.13E+08
11	t1=15	t2=13	t3=12	mw1=64	mw2=32	mw3=16	r1=50	r2=20	r3=10	LUTlimit=0.1	2.13E+08
12	t1=15	t2=13	t3=12	mw1=64	mw2=32	mw3=16	r1=50	r2=20	r3=10	LUTlimit=0.2	2.13E+08
13	t1=15	t2=13	t3=12	mw1=64	mw2=32	mw3=16	r1=50	r2=20	r3=10	LUTlimit=0.3	2.13E+08
14	t1=15	t2=13	t3=12	mw1=64	mw2=32	mw3=16	r1=50	r2=20	r3=10	LUTlimit=0.4	2.13E+08
15	t1=15	t2=13	t3=12	mw1=64	mw2=32	mw3=16	r1=50	r2=20	r3=10	LUTlimit=0.5	2.13E+08
16	t1=15	t2=13	t3=12	mw1=64	mw2=32	mw3=16	r1=50	r2=20	r3=10	LUTlimit=0.6	2.13E+08
17	t1=15	t2=13	t3=12	mw1=64	mw2=32	mw3=16	r1=50	r2=20	r3=10	LUTlimit=0.7	2.13E+08
18	t1=15	t2=13	t3=12	mw1=64	mw2=32	mw3=16	r1=50	r2=20	r3=10	LUTlimit=0.8	2.13E+08
19	t1=15	t2=13	t3=12	mw1=64	mw2=32	mw3=16	r1=50	r2=20	r3=10	LUTlimit=0.9	2.13E+08
20	t1=15	t2=13	t3=12	mw1=64	mw2=32	mw3=16	r1=50	r2=20	r3=10	LUTlimit=1.0	2.13E+08

Figure 12: Top 20 best organizations for 3 BRAMs

1	t1=14	t2=13	t3=	mw1=32	mw2=32	mw3=	r1=20	r2=20	r3=	LUTlimit=0.4	2.02E+08
2	t1=14	t2=13	t3=	mw1=32	mw2=32	mw3=	r1=20	r2=20	r3=	LUTlimit=0.5	2.02E+08
3	t1=14	t2=13	t3=	mw1=32	mw2=32	mw3=	r1=20	r2=20	r3=	LUTlimit=0.6	2.02E+08
4	t1=14	t2=13	t3=	mw1=32	mw2=32	mw3=	r1=20	r2=20	r3=	LUTlimit=0.7	2.02E+08
5	t1=14	t2=13	t3=	mw1=32	mw2=32	mw3=	r1=20	r2=20	r3=	LUTlimit=0.8	2.02E+08
6	t1=14	t2=13	t3=	mw1=32	mw2=32	mw3=	r1=20	r2=20	r3=	LUTlimit=0.9	2.02E+08
7	t1=14	t2=13	t3=	mw1=32	mw2=32	mw3=	r1=20	r2=20	r3=	LUTlimit=1.0	2.02E+08
8	t1=14	t2=12	t3=	mw1=32	mw2=16	mw3=	r1=20	r2=10	r3=	LUTlimit=0.4	2.03E+08
9	t1=14	t2=12	t3=	mw1=32	mw2=16	mw3=	r1=20	r2=10	r3=	LUTlimit=0.5	2.03E+08
10	t1=14	t2=12	t3=	mw1=32	mw2=16	mw3=	r1=20	r2=10	r3=	LUTlimit=0.6	2.03E+08
11	t1=14	t2=12	t3=	mw1=32	mw2=16	mw3=	r1=20	r2=10	r3=	LUTlimit=0.7	2.03E+08
12	t1=14	t2=12	t3=	mw1=32	mw2=16	mw3=	r1=20	r2=10	r3=	LUTlimit=0.8	2.03E+08
13	t1=14	t2=12	t3=	mw1=32	mw2=16	mw3=	r1=20	r2=10	r3=	LUTlimit=0.9	2.03E+08
14	t1=14	t2=12	t3=	mw1=32	mw2=16	mw3=	r1=20	r2=10	r3=	LUTlimit=1.0	2.03E+08
15	t1=14	t2=13	t3=	mw1=32	mw2=32	mw3=	r1=20	r2=20	r3=	LUTlimit=0.3	2.03E+08
16	t1=14	t2=12	t3=	mw1=32	mw2=16	mw3=	r1=20	r2=10	r3=	LUTlimit=0.3	2.03E+08
17	t1=14	t2=12	t3=	mw1=32	mw2=16	mw3=	r1=20	r2=10	r3=	LUTlimit=0.2	2.04E+08
18	t1=15	t2=12	t3=	mw1=64	mw2=16	mw3=	r1=50	r2=10	r3=	LUTlimit=0.5	2.04E+08
19	t1=15	t2=12	t3=	mw1=64	mw2=16	mw3=	r1=50	r2=10	r3=	LUTlimit=0.6	2.04E+08
20	t1=15	t2=12	t3=	mw1=64	mw2=16	mw3=	r1=50	r2=10	r3=	LUTlimit=0.7	2.04E+08

Figure 13: Top 20 best organizations for 2 BRAMs + LUTRAM