ECE60827 HW Sim Assignment #1 Lab Report

Conor X Devlin, PUID: 0035599728

Part #1 Question #1: "What do these numbers say about the design of the PTX versus SASS ISA?"

We find that looking at total instructions of PTX and SASS that SASS requires *more* instructions to execute the same benchmarks as PTX but PTX requires *more* cycles to execute the same workload. The decision to be made is does your simulation cycles matter to you, and if so to what extent. We are proved correct in the basic assumption that SASS (the actual hardware ISA) is going to be faster (require less cycles) than PTX (virtual ISA, a higher level of abstraction) but require more instructions.

What this says about the *design* of PTX versus SASS is that PTX is more "human readable" and thus *easier* to write as we know the order is CUDA -> PTX -> SASS. SASS is the native hardware ISA of the GPU and is therefore going to have a more detailed design. This is obviously proved as PTX generally has less instructions per sim compared to SASS which is indicative of it's higher level of abstraction.

Part #1 Question #2: "What do these numbers say about the optimization of the PTX versus SASS ISA?"

These numbers: PTX insns < SASS insns and, PTX cycles > SASS cycles, indicate that the lower level, *native ISA* (SASS) can provide a better level of optimization for the user contingent on if they want to (or can) directly optimize their specific GPU and architecture. Whereas PTX, which is the more abstract, higher-level language, is not as optimized as PTX but assuming you could also compare against CUDA then likely you'd see the optimization chain being SASS > PTX > CUDA. PTX is more optimized than CUDA, not as much as SASS but PTX is more difficult and time consuming to write in comparison to CUDA and SASS is even *more* impractical to write than PTX but offers the programmer the most optimized level of their kernel.

[Graphs for Part #1 are found in Appendix A]

Part #2 Question #1: "Is there a connection between these divergence metrics and the application's instructions-per-cycle (IPC)?"

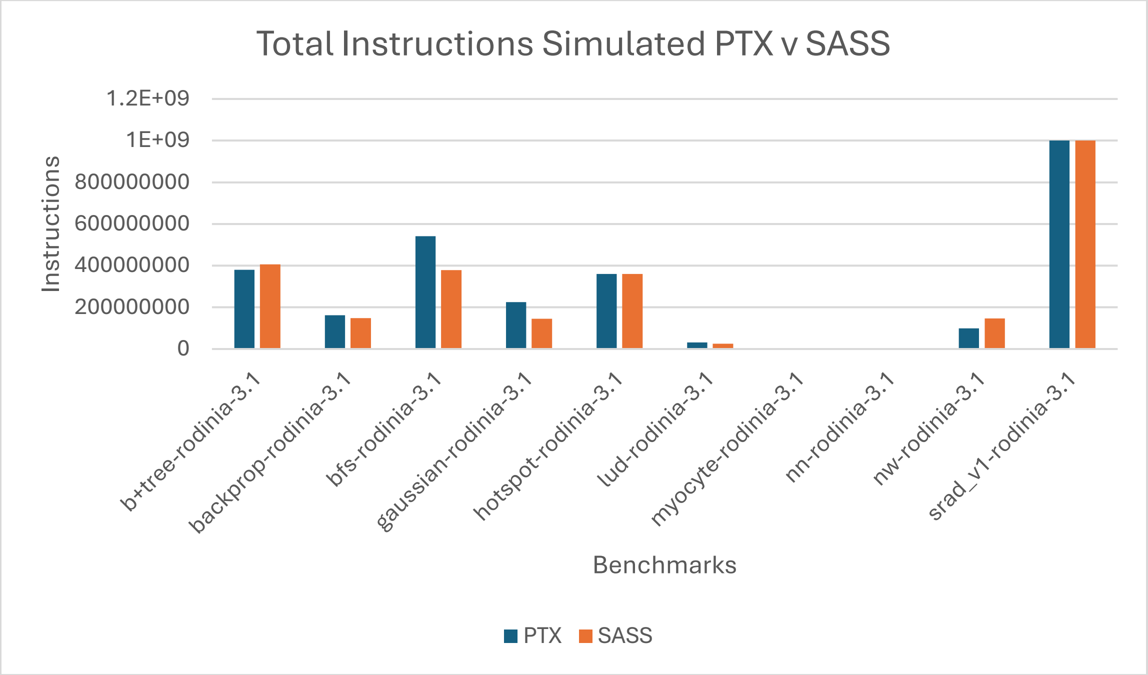
Comparing IPC to Memory Divergence there is a *weak* correlation between them. We find that b+tree and backprop both have IPCs nearing 3K but have *less* divergence compared to nn-rodinia and nw-rodinia which both have divergence >60% but IPCs between 215-58 which pale in comparison to b+tree and backprop. Moreover hotspot-rodinia has the highest IPC of ~6.5K while it's memory divergence is just under 40%. Then there are benchmarks like gaussian, lud-rodinia, and myocyte-rodinia which both have IPCs near or under 50 and then have memory divergences under 10% which would seem to *meet* expectations that less IPC means less divergence. The use of *weak* correlation means that yes we see high IPCs, relatively, and a corresponding 30-50% memory divergence but nothing that would really cause divergence to dominate, and where it does it is with a benchmark featuring and IPCs of 50-200. I'm stating that while memory divergence can impact IPC in certain workloads, the connection is app-dependent and influenced by occupancy/instructions/stalls/etc. Even if I could push an extreme IPC (>100K) it wouldn't necessarily result in high memory divergence — if memory accesses are properly coalesced, divergence could remain low despite high throughput, (assuming no other bottlenecks).

Part #2 Question #2: "Comment on why this is (or is not) the case. Remember to use the data to answer this question, not just intuition and if the data runs counter to your intuition, attempt to use other data to explain why. (Hint: Think about occupancy too)

Looking at the occupancy graph we find greater (natural) correlation of IPC to Occupancy with all of the benchmarks with high IPCs (>1000) having similarly high occupancies (>78%). Looking at nn-rodinia and nw-rodinia we find very low occupancies (<22%), low IPCs (<200) and the highest memory divergence (>60%) which does make logical sense. Furthermore, those two benchmarks *make* logical sense because if we consider that memory divergence is a cause or factor of the observed low IPC and occupancy. Moreover, a benchmark like hotspot-rodinia with an IPC of 6.5K, an occupancy of 84% is justified in having a relatively low divergence at 37%-- it's got a bit of divergence but it's firing off thousands of instructions per cycle and is able to keep its SMs highly occupied. So the added occupancy data proves the idea that if there is high memory divergence then our benchmarks should likely have lower IPCs and occupancies. Following this line of logic it would be valuable to compare stalls across all benchmarks which we see below where (aside from srad_v1-rodinia) all of our high IPC benchmarks have a relatively low amount of stalls. Ultimately IPC and memory divergence is highly benchmark or app-dependent, one doesn't necessarily imply the other as we see proven out in the data below.

[Graphs for Part #2 are found in Appendix B]

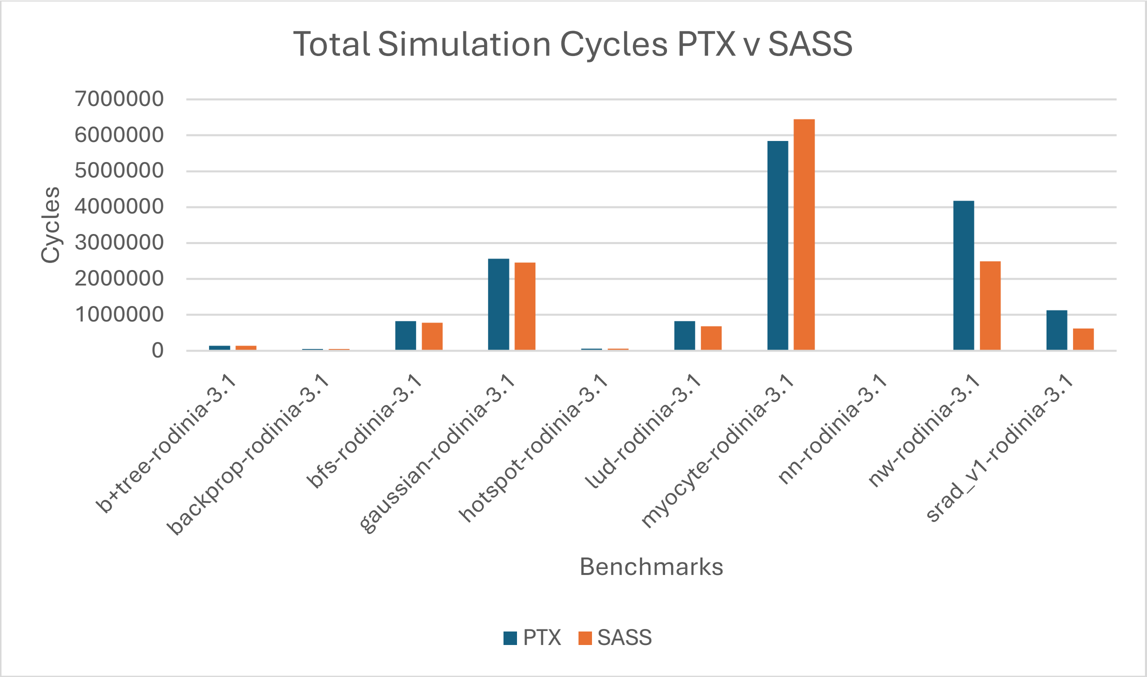
Appendix A



Graph 1

Tot Sim. Insns	**PTX**	**SASS**	Greater?
b+tree-rodinia-3.1	380891849	407021552	SASS
backprop-rodinia-3.1	161546592	148832592	PTX
bfs-rodinia-3.1	541515763	379371108	PTX
gaussian-rodinia-3.1	225527328	144833960	PTX
hotspot-rodinia-3.1	360179256	360616472	SASS
lud-rodinia-3.1	32201872	25984224	PTX
myocyte-rodinia-3.1	799070	1669062	SASS
nn-rodinia-3.1	1201052	1327636	SASS
nw-rodinia-3.1	99791872	146325504	SASS
srad_v1-rodinia-3.1	1000001571	1000004584	SASS

Table 1

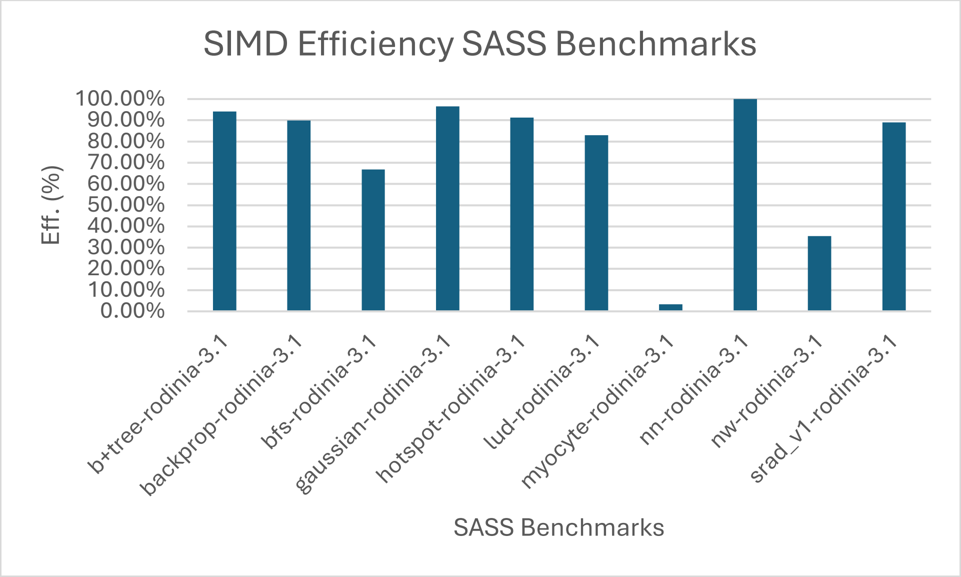


Graph 2

Tot Sim. Cycles	**PTX**	**SASS**	Greater?
b+tree-rodinia-3.1	140008	137441	PTX
backprop-rodinia-3.1	51130	52681	SASS
bfs-rodinia-3.1	822254	777265	PTX
gaussian-rodinia-3.1	2561382	2458560	PTX
hotspot-rodinia-3.1	59031	56104	PTX
lud-rodinia-3.1	824835	678793	PTX
myocyte-rodinia-3.1	5841895	6448120	SASS
nn-rodinia-3.1	6057	6251	SASS
nw-rodinia-3.1	4179943	2489587	PTX
srad_v1-rodinia-3.1	1124494	617870	PTX

Table 2

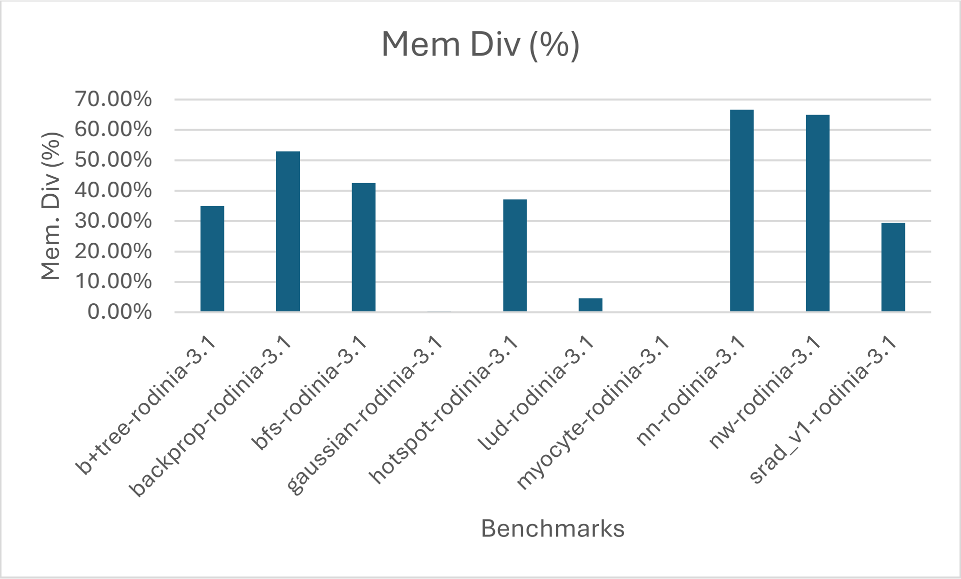
Appendix B



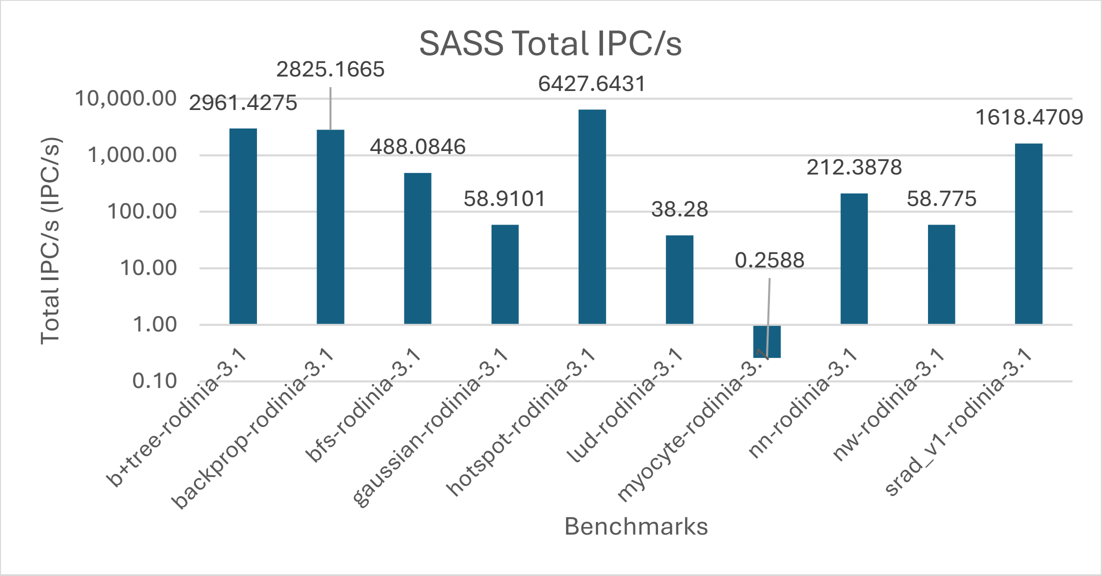
Graph 3

SASS Benchmarks	Percentage (%)
b+tree-rodinia-3.1	0.942100848
backprop-rodinia-3.1	0.899357773
bfs-rodinia-3.1	0.667711386
gaussian-rodinia-3.1	0.965423973
hotspot-rodinia-3.1	0.91200162
lud-rodinia-3.1	0.829926828
myocyte-rodinia-3.1	0.032569758
nn-rodinia-3.1	0.999629554
nw-rodinia-3.1	0.355081107
srad_v1-rodinia-3.1	0.890737891

Table 3



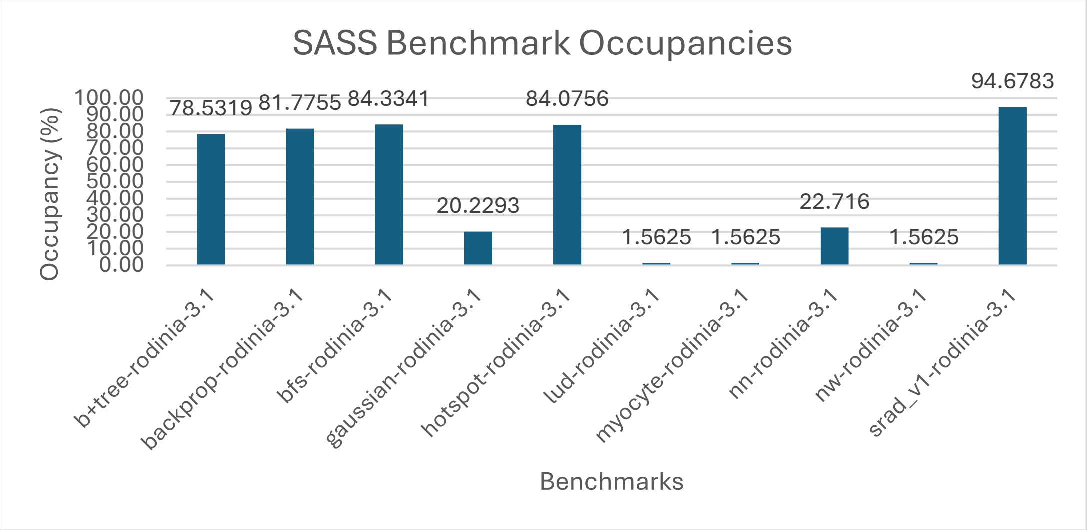
Graph 4



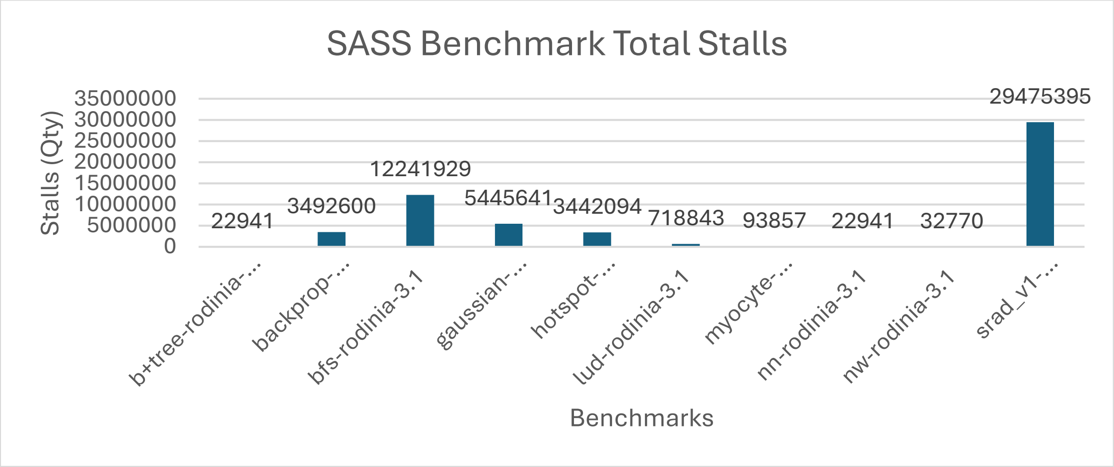
Graph 5

	gpgpu_n_mem_read_global	gpgpu_n_mem_write_global	dram read	dram write	Mem Div (%)
b+tree-rodinia-3.1	1269851	110000	476655	5456	0.3493935
backprop-rodinia-3.1	460716	524294	434188	87088	0.5292088
bfs-rodinia-3.1	8342383	3872126	3403334	1785115	0.4247775
gaussian-rodinia-3.1	797150	1711242	8216	0	0.0032754
hotspot-rodinia-3.1	691072	175104	262144	59748	0.3716242
lud-rodinia-3.1	130432	47600	8192	0	0.0460142
myocyte-rodinia-3.1	2652	14664	15	0	0.0008663
nn-rodinia-3.1	10691	5346	10691	0	0.6666459
nw-rodinia-3.1	1064960	753664	756096	425295	0.6496071
srad_v1-rodinia-3.1	3708132	2218137	30855	1713336	0.2943152

Table 4



Graph 6



Graph 7