dEQP-VK.graphicsfuzz.complex-nested-loops-and-call #204

New issue

Closed   paulthomson opened this issue on 13 Sep 2019 · 20 comments

paulthomson commented on 13 Sep 2019   Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

complex-nested-loops-and-call.zip

amdrexu assigned **linqun** on 17 Sep 2019

jaebaek commented on 23 Sep 2019   Collaborator

I built it in Release 64bit mode based on AMDVLK and spvgen, but I cannot reproduce this bug.

Did someone fix this already?

paulthomson commented on 24 Sep 2019 • edited   Collaborator   Author

I used a Debug build on dev branch.

kuhar self-assigned this on 24 Sep 2019

amdrexu unassigned **linqun** on 25 Sep 2019

Assignees

No one assigned

Labels

fixed in LLVM

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

8 participants

kuhar commented on 25 Sep 2019 • edited

Collaborator

Seems like there is a bug in `SILowerI1Copies.cpp` (an LLVM MachineFunction pass).

It happens during the 54th iteration of the inner for loop over machine instructions in `SILowerI1Copies::lowerPhis` when a phi has a use in the same machine basic block:

```
bb.39 (%ir-block.176):
; predecessors: %bb.38
  successors: %bb.41(0x40000000),
%bb.42(0x40000000); %bb.41(50.00%),
%bb.42(50.00%)

  %457:vgpr_32 = PHI %458:vgpr_32, %bb.38
  %153:sreg_64 = PHI %315:vreg_1, %bb.38    ;
<== def
  %154:vgpr_32 = PHI %316:vgpr_32, %bb.38
  %155:vgpr_32 = PHI %316:vgpr_32, %bb.38
  %156:vgpr_32 = PHI %316:vgpr_32, %bb.38
  %320:sreg_64_xexec = COPY %153:sreg_64    ;
<== use
  %319:vgpr_32 = V_CNDMASK_B32_e64 0, 0, 0, 1,
%320:sreg_64_xexec, implicit $exec
  %321:sreg_64 = S_MOV_B64 -1
  %318:vreg_1 = COPY %321:sreg_64
```

```
%319:vgpr_32, implicit-def $vcc, implicit $exec
  $vcc = S_AND_B64 $exec, $vcc, implicit-def
$scc
  S_CBRANCH_VCCNZ %bb.41, implicit $vcc
  S_BRANCH %bb.42
```

The pass attempts to create a new register and replace the Phi with a new register placed at the end of the block. This is how the machine basic blocks looks at the end of that loop iteration:

```
bb.39 (%ir-block.176):
; predecessors: %bb.38
  successors: %bb.41(0x40000000),
%bb.42(0x40000000); %bb.41(50.00%),
%bb.42(50.00%)

  %457:vgpr_32 = PHI %458:vgpr_32, %bb.38
```

```
    %584:sreg_64 = PHI %315:vreg_1, %bb.38      ;
  <== NewReg (old instruction, will be erased)
    %154:vgpr_32 = PHI %316:vgpr_32, %bb.38
    %155:vgpr_32 = PHI %316:vgpr_32, %bb.38
    %156:vgpr_32 = PHI %316:vgpr_32, %bb.38
    %320:sreg_64_xexec = COPY %153:sreg_64      ;
  <== use
    %319:vgpr_32 = V_CNDMASK_B32_e64 0, 0, 0, 1,
  %320:sreg_64_xexec, implicit $exec
    %321:sreg_64 = S_MOV_B64 -1
    %318:vreg_1 = COPY %321:sreg_64
    %322:sreg_32_xm0 = S_MOV_B32 1
    %323:sreg_32 = IMPLICIT_DEF
    V_CMP_NE_U32_e32 killed %322:sreg_32_xm0,
  %319:vgpr_32, implicit-def $vcc, implicit $exec
    $vcc = S_AND_B64 $exec, $vcc, implicit-def
  $scc
    %153:sreg_64 = IMPLICIT_DEF                ;
  <== DstReg (new instruction, to be used instead)
    S_CBRANCH_VCCNZ %bb.41, implicit $vcc
    S_BRANCH %bb.42
```

After this iteration, the function verifiers complains that the definition doesn't dominate the use. I think that the new value from the MachineSSAUpdater should be placed before the use, instead of at the end of the block, which would require a small modification to the MachineSSAUpdater.

I hope that this makes sense and I'd really appreciate some tips/relevant documentation if you have any. I'll resume the investigation tomorrow from this point.

trenouf commented on 26 Sep 2019 Collaborator

Hi Jakub

I'll pass this on to Matt and Stas (AMD engineers on the compute side), who look like the people most linked with SILowerI1Copies, and see what we get back.

nhaehnle commented on 26 Sep 2019 • edited

Collaborator

I haven't been able to do a full analysis on this and will have to take a closer look at the CFG overall. What I *can* say is that the new value %153 is almost certainly correct where it is (or at least "more correct" than moving it would be). This is because we certainly want a defined value to be used in the V_CNDMASK_B32, since thats what's happens in the original code.

That is to say, it seems more likely that the correct fix would be to change the use of %153 into a use of some other newly generated instruction that does dominate.

arsenm commented on 26 Sep 2019 • edited

This is some confusing shadowing to me; I would assume the second MBB really intended to be the current iterated block, not the predecessor
`for (MachineBasicBlock *MBB : PIA.predecessors()) SSAUpdater.AddAvailableValue(MBB, insertUndefLaneMask(*MBB));`

kuhar commented on 26 Sep 2019

Collaborator

Thanks **@trenouf**!

@nhaehnle, **@arsenm** thank you for looking into this and your insights.
Can you recommend me some resources to learn more about what SILowerI1Copies is trying to achieve and why? I don't fully understand the transformation after reading the header and the implementation files.
To be specific, I'm confused about what values are supposed to flow into the uses in the middle of blocks and what the semantics of implicit defs are.
Without a better understanding of the pass I don't know if I can be of much use when it comes to this bug, unless somebody tells me exactly what is broken here.

arsenm commented on 26 Sep 2019

Fundamentally SILowerI1Copies is a SelectionDAG workaround. In the DAG we don't have the necessary context to know how to treat a boolean value. We use the pseudo register-class VReg_1, and then SILowerI1Copies sorts out when it's appropriate to use a real lane mask for these values.

trenouf commented on 26 Sep 2019 Collaborator

Stas said:

If it did replace %153 with %584 it should probably also replace its uses with the new %584. I see that COPY left using %153.

kuhar removed their assignment on 26 Sep 2019

kuhar commented on 26 Sep 2019 • edited Collaborator

@trenouf

> Stas said:
>
> If it did replace %153 with %584 it should probably also replace its uses with the new %584. I see that COPY left using %153.

Yes, but the new value %584 does not dominate the use, even if the use gets updated.

trenouf commented on 26 Sep 2019 Collaborator

Nicolai said (in reply to Stas's comment):

%584 may just be an artefact of how the registers are swapped around in order to erase instructions.

nhaehnle commented on 27 Sep 2019 Collaborator

@kuhar, I've root-caused a first issue to a subtle bug in the MachineSSAUpdater, but there's a follow-on issue that I'm currently looking at.

kuhar commented on 27 Sep 2019 Collaborator

@nhaehnle OK. Let me know if you need help with this issue.

nhaehnle added a commit to nhaehnle/llvm-project that referenced this issue on 27 Sep 2019

`MachineSSAUpdater: insert IMPLICIT_DEF at top of basic block …` 782f7ff

nhaehnle added a commit to nhaehnle/llvm-project that referenced this issue on 27 Sep 2019

`AMDGPU: Propagate undef flag during pre-RA exec mask optimizations …` b2e1caa

nhaehnle commented on 27 Sep 2019 Collaborator

@kuhar, the branch here with two simple changes allows the problematic shader to compile: https://github.com/nhaehnle/llvm-project/tree/wip-github-llpc-204

I'm currently travelling and not setup to run amber tests. Could you please check whether those changes fully fix the issue in the dEQP-VK test?

kuhar commented on 27 Sep 2019 • edited Collaborator

@nhaehnle Applying the patches seems to fix the issue - MIR verifieas correctly now.

nhaehnle commented on 28 Sep 2019 Collaborator

See https://reviews.llvm.org/D68183 and https://reviews.llvm.org/D68184

llvm-git-migration pushed a commit to llvm/llvm-project that referenced this issue on 8 Oct 2019

`MachineSSAUpdater: insert IMPLICIT_DEF at top of basic block …` ✓ 7febdb7

llvm-git-migration pushed a commit to llvm/llvm-project that referenced this issue on 8 Oct 2019

`AMDGPU: Propagate undef flag during pre-RA exec mask optimizations …` ✓ df6e676

dtzWill pushed a commit to llvm-mirror/llvm that referenced this issue on 8 Oct 2019

`MachineSSAUpdater: insert IMPLICIT_DEF at top of basic block …` aef8d68

dtzWill pushed a commit to llvm-mirror/llvm that referenced this issue on 8 Oct 2019

`AMDGPU: Propagate undef flag during pre-RA exec mask optimizations …` c54901d

chapuni pushed a commit to llvm-project/llvm-project-20170507 that referenced this issue on 8 Oct 2019

`MachineSSAUpdater: insert IMPLICIT_DEF at top of basic block …` 22f78c9

chapuni pushed a commit to llvm-project/llvm-project-20170507 that referenced this issue on 8 Oct 2019

`AMDGPU: Propagate undef flag during pre-RA exec mask optimizations …` 9a6a16b

chapuni pushed a commit to llvm-project/llvm-project-submodule that referenced this issue on 8 Oct 2019

`MachineSSAUpdater: insert IMPLICIT_DEF at top of basic block …` 391a133

chapuni pushed a commit to llvm-project/llvm-project-submodule that referenced this issue on 8 Oct 2019

`AMDGPU: Propagate undef flag during pre-RA exec mask optimizations …` 0021f04

earl pushed a commit to earl/llvm-mirror that referenced this issue on 8 Oct 2019

`MachineSSAUpdater: insert IMPLICIT_DEF at top of basic block` … `dd03d52`

earl pushed a commit to earl/llvm-mirror that referenced this issue on 8 Oct 2019

`AMDGPU: Propagate undef flag during pre-RA exec mask optimizations` … `7ecc141`

nhaehnle commented on 8 Oct 2019 Collaborator

@amdrexu @paulthomson @kuhar We should probably close this issue given that the fix has gone into upstream LLVM, but I don't seem to have the rights to actually do that.

paulthomson commented on 8 Oct 2019 • edited Collaborator Author

Or we could close it when the LLVM version is updated, possibly adding a "fixed in LLVM" label in the meantime (or something like that). We don't have continuous running of these tests yet, but when we do, the automation could create/re-open issues like this one automatically unless the dev branch is passing.

jinjianrong added the **fixed in LLVM** label on 9 Oct 2019

paulthomson commented on 11 Nov 2019 Collaborator Author

This still seems to fail.

paulthomson commented on 12 Nov 2019 Collaborator Author

This appears to be fixed.

paulthomson closed this on 12 Nov 2019

---

arichardson added a commit to arichardson/llvm-project that referenced this issue on 16 Nov 2019

`MachineSSAUpdater: insert IMPLICIT_DEF at top of basic block …` 62db175

arichardson added a commit to arichardson/llvm-project that referenced this issue on 16 Nov 2019

`AMDGPU: Propagate undef flag during pre-RA exec mask optimizations …` 57285bc

JohnHolmesII pushed a commit to JohnHolmesII/llvm-project that referenced this issue on 12 Oct 2020

`MachineSSAUpdater: insert IMPLICIT_DEF at top of basic block …` b96707a

JohnHolmesII pushed a commit to JohnHolmesII/llvm-project that referenced this issue on 12 Oct 2020

`AMDGPU: Propagate undef flag during pre-RA exec mask optimizations …` 6d444de

GPUOpen-Drivers / **llpc**

dEQP-VK.graphicsfuzz.loop-dead-if-loop #207

New issue

Closed **paulthomson** opened this issue on 13 Sep 2019 · 8 comments

paulthomson commented on 13 Sep 2019 Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

loop-dead-if-loop.zip

amdrexu assigned **jiaolu** on 17 Sep 2019

jaebaek commented on 23 Sep 2019 Collaborator

I built it in Release 64bit mode based on AMDVLK and spvgen, but I cannot reproduce this bug.

Did someone fix this already?

paulthomson commented on 24 Sep 2019 • edited Collaborator Author

I used a Debug build on dev branch.

Assignees

kuhar

Labels

fixed in LLVM

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

4 participants

kuhar commented on 24 Sep 2019 Collaborator

I can reproduce in Debug mode on the master branch.
Looking into it.

kuhar self-assigned this on 24 Sep 2019

kuhar commented on 24 Sep 2019 Collaborator

I submitted an LLVM commit fixing the issue here:
https://reviews.llvm.org/D67974

amdrexu unassigned **jiaolu** on 25 Sep 2019

kuhar commented on 25 Sep 2019 Collaborator

The fix was committed in
https://reviews.llvm.org/rL372874.

kuhar commented on 27 Sep 2019 Collaborator

After doing some cleanup in LLVM when working on this issue, a new issue with MachineDominators surfaced: PHIElimination lied about preserving MDT. I submitted a fix here: https://reviews.llvm.org/D68154
Not sure if the AMDGPU target uses this part of the machine pass pipeline, but may be work merging in this fix too.

kuhar commented on 2 Oct 2019 • edited

Collaborator

The Machine (Post)Dominators should be fully fixed now by the series of LLVM commits:

0. https://reviews.llvm.org/rL372874 : [Dominators][AMDGPU] Don't use virtual exit node in findNearestCommonDominator.
1. https://reviews.llvm.org/rL373341 : [Dominators][CodeGen] Add MachinePostDominatorTree verification.
2. https://reviews.llvm.org/rL373376 : Reapply [Dominators][CodeGen] Clean up MachineDominators.
3. https://reviews.llvm.org/rL373377 : [Dominators][CodeGen] Fix MachineDominatorTree preservation in PHIElimination.
4. https://reviews.llvm.org/rL373378 : [Dominators][CodeGen] Don't mark MachineDominatorTree as preserved in MachineLICM.
5. https://reviews.llvm.org/rL373382 : Add a missing pass in ARM O3 pipeline.

kuhar added the fixed in LLVM label on 9 Oct 2019

paulthomson commented on 11 Nov 2019

Collaborator Author

Appears to be fixed!

paulthomson closed this on 11 Nov 2019

GPUOpen-Drivers / **llpc**

Code | Issues 30 | Pull requests 6 | Actions | Projects 7

dEQP-VK.graphicsfuzz.loops-ifs-continues-call #209

New issue

Closed **paulthomson** opened this issue on 13 Sep 2019 · 6 comments

paulthomson commented on 13 Sep 2019 — Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

loops-ifs-continues-call.zip

amdrexu assigned **jiaolu** on 17 Sep 2019

jaebaek self-assigned this on 23 Sep 2019

amdrexu unassigned **jiaolu** on 24 Sep 2019

MarcinKantochAMD commented on 24 Sep 2019

I can't find this test yet available. Can you verify Paul that the name is correct.

MarcinKantochAMD commented on 24 Sep 2019

Sorry Paul. I was not aware that your issues relate to Debug build. I was testing with a Release build.

Assignees

jaebaek

Labels

fixed in LLVM

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

5 participants

paulthomson commented on 24 Sep 2019 Collaborator Author

Some of the tests are still pending CLs but the AmberScript file and the SPIR-V shader is attached.

jaebaek commented on 25 Sep 2019 Collaborator

Sent a PR: https://reviews.llvm.org/D68032

jaebaek commented on 26 Sep 2019 • edited Collaborator

The PR was merged: llvm/llvm-project@ `d98cb81`
Once it is applied to LLPC, we can close this issue.

kuhar added the fixed in LLVM label on 9 Oct 2019

paulthomson commented on 11 Nov 2019 Collaborator Author

Appears to be fixed!

paulthomson closed this on 11 Nov 2019

dEQP-VK.graphicsfuzz.two-loops-with-break #217

New issue

Closed **paulthomson** opened this issue on 13 Sep 2019 · 4 comments

paulthomson commented on 13 Sep 2019 — Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

two-loops-with-break.zip

amdrexu assigned **xuechen417** on 17 Sep 2019

kuhar commented on 17 Sep 2019 — Collaborator

This seems to a BB iterator invalidation bug in LLVM's FlattenCFG pass. The code is unchanged in ToT LLVM. I'll look further into this, try to reproduce it in LLVM, and report back.

kuhar commented on 17 Sep 2019 • edited — Collaborator

I reproduced the bug in FlattenCFG in the ToT LLVM and submitted a patch for review here: https://reviews.llvm.org/D67672

Assignees

kuhar

Labels

fixed in LLVM

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

3 participants

kuhar commented on 19 Sep 2019 Collaborator

The fix was committed in https://reviews.llvm.org/rL372347 and should resolve the issue after updating LLVM.

amdrexu unassigned **xuechen417** on 25 Sep 2019

kuhar self-assigned this on 25 Sep 2019

kuhar added the fixed in LLVM label on 9 Oct 2019

paulthomson commented on 11 Nov 2019 Collaborator Author

Appears to be fixed!

paulthomson closed this on 11 Nov 2019

dEQP-VK.graphicsfuzz.call-if-while-switch #203

New issue

Closed · paulthomson opened this issue on 13 Sep 2019 · 8 comments

paulthomson commented on 13 Sep 2019 · Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

call-if-while-switch.zip

amdrexu commented on 16 Sep 2019 · Collaborator

Are those issues urgent for you? We need to know the priority.

paulthomson commented on 16 Sep 2019 · Collaborator · Author

These are not urgent. Thanks.

amdrexu commented on 17 Sep 2019 · Collaborator

Thank you for feedbacks. We will arrange resources to handle those issues.

Assignees

kuhar

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

7 participants

jinjianrong commented on 17 Sep 2019 Member

@paulthomson
Paul, would you like to take the ownership to fix these issues?

amdrexu commented on 17 Sep 2019 Collaborator

I will assign those issues to developers. If someone else believes he can handle this, feel free to reassign it to yourself and make comments in the issue ticket.

amdrexu assigned **linqun** on 17 Sep 2019

dnovillo commented on 17 Sep 2019 Collaborator

@jinjianrong I will assign them to myself for now. I will re-assign as folks start working on them internally.

dnovillo commented on 17 Sep 2019 Collaborator

@jinjianrong

We will need access to this repository. I am not allowed to assign issue to me at the moment. How do we get access? Thanks.

amdrexu unassigned **linqun** on 25 Sep 2019

kuhar self-assigned this on 26 Sep 2019

kuhar mentioned this issue on 27 Sep 2019

Add missing phi entries for duplicate incoming arcs #244 Merged

kuhar commented on 18 Oct 2019 Collaborator

This issue is fixed and can be closed.

s-perron closed this on 7 Nov 2019

dEQP-VK.graphicsfuzz.control-flow-in-function #205

New issue

Closed paulthomson opened this issue on 13 Sep 2019 · 6 comments

paulthomson commented on 13 Sep 2019 Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

control-flow-in-function.zip

amdrexu assigned **linqun** on 17 Sep 2019

jaebaek commented on 23 Sep 2019 Collaborator

I built it in Release 64bit mode based on AMDVLK and spvgen, but I cannot reproduce this bug.

Did someone fix this already?

paulthomson commented on 24 Sep 2019 • edited Collaborator Author

I used a Debug build on dev branch.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

4 participants

amdrexu commented on 24 Sep 2019 Collaborator

To reproduce this issue, a debug version of AMDLLPC might be required.

amdrexu unassigned **linqun** on 25 Sep 2019

jaebaek self-assigned this on 25 Sep 2019

jaebaek commented on 26 Sep 2019 Collaborator

Yes, I can reproduce it with the debug build. Thank for you doublechecking it.
I am investigating this issue now.

jaebaek commented on 16 Oct 2019 Collaborator

Sorry. I spent some time but I do not understand it. It is complicated. Let me unassign me for now.

jaebaek removed their assignment on 16 Oct 2019

paulthomson commented on 11 Nov 2019 Collaborator Author

Appears to be fixed!

paulthomson closed this on 11 Nov 2019

GPUOpen-Drivers / **llpc**

Code   Issues 30   Pull requests 6   Actions   Projects 7

dEQP-VK.graphicsfuzz.loop-nested-ifs #208

New issue

Closed   **paulthomson** opened this issue on 13 Sep 2019 · 8 comments

paulthomson commented on 13 Sep 2019   Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

loop-nested-ifs.zip

amdrexu assigned **jiaolu** on 17 Sep 2019

kuhar self-assigned this on 23 Sep 2019

kuhar mentioned this issue on 23 Sep 2019

Ensure duplicate incoming blocks share the same incoming values #235   Merged

amdrexu unassigned **jiaolu** on 24 Sep 2019

MarcinKantochAMD commented on 24 Sep 2019

I can't reproduce the issue with our LLPC stack.

Assignees

kuhar

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

4 participants

paulthomson commented on 24 Sep 2019 Collaborator Author

This still fails for me. Are you using a 64 bit debug build of the dev branch?

paulthomson commented on 24 Sep 2019 Collaborator Author

```
amdllpc -gfxip=9.0.0 -verify-ir -o temp.out
*.spv
PHI node has multiple entries for the same basic
block with different incoming values!
  %__llpc_output_proxy_.4.0 = phi float [ %34,
%25 ], [ %33, %25 ], [ %40, %35 ], [
1.000000e+00, %41 ]
label %25
  %33 = extractelement <4 x float> %spec.select,
i32 0
  %34 = extractelement <4 x float> %spec.select,
i32 0
in function llpc.shader.FS.main
ERROR: LLVM FATAL ERROR:Broken function found,
compilation aborted!
```

kuhar commented on 24 Sep 2019 Collaborator

I can reproduce it on master in Debug mode.

MarcinKantochAMD commented on 24 Sep 2019

Sorry Paul. I was not aware that your issues relate to Debug build. I was testing with a Release build.

paulthomson commented on 24 Sep 2019 Collaborator Author

Yes, sorry I should have made this clearer.

kuhar commented on 9 Oct 2019 Collaborator

This issue is fixed and can be closed.

paulthomson commented on 11 Nov 2019 Collaborator Author

Appears to be fixed!

paulthomson closed this on 11 Nov 2019

dEQP-VK.graphicsfuzz.max-mix-conditional-discard #210

New issue

Closed **paulthomson** opened this issue on 13 Sep 2019 · 4 comments

paulthomson commented on 13 Sep 2019 Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

max-mix-conditional-discard.zip

amdrexu self-assigned this on 17 Sep 2019

MarcinKantochAMD commented on 24 Sep 2019

I tested today and the test is passing on our LLPC stack.

MarcinKantochAMD commented on 24 Sep 2019

Sorry Paul. I was not aware that your issues relate to Debug build. I was testing with a Release build.

amdrexu removed their assignment on 25 Sep 2019

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

4 participants

kuhar commented on 27 Sep 2019 • edited

Collaborator

LLPC fails in `SIInstrInfo.cpp:3197` when verifying the following instruction:

```
bb.4 (%ir-block.34):
; predecessors: %bb.1
  successors: %bb.5(0x40000000),
%bb.6(0x40000000); %bb.5(50.00%), %bb.6(50.00%)

  S_CMPK_GT_I32 8, 179, implicit-def $scc
;  <===
  S_CBRANCH_SCC1 %bb.6, implicit killed $scc
$3 = void
```

The diagnostic message is for the 0th operand and says: `Illegal immediate value for operand.`

I don't understand MIR enough to tell what the issue here really is.

paulthomson commented on 11 Nov 2019

Collaborator Author

Appears to be fixed!

paulthomson closed this on 11 Nov 2019

GPUOpen-Drivers / **llpc**

Code | Issues 30 | Pull requests 6 | Actions | Projects 7

dEQP-VK.graphicsfuzz.modf-gl-color #212

New issue

Closed **paulthomson** opened this issue on 13 Sep 2019 · 5 comments

paulthomson commented on 13 Sep 2019 Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

modf-gl-color.zip

amdrexu self-assigned this on 17 Sep 2019

jaebaek commented on 23 Sep 2019 Collaborator

I cannot reproduce the failure. I want to doublecheck if this is already fixed.

paulthomson commented on 24 Sep 2019 • edited Collaborator Author

I used a Debug build on dev branch.

MarcinKantochAMD commented on 24 Sep 2019

I ran the test today and it is passing on our LLPC stack.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

4 participants

MarcinKantochAMD commented on 24 Sep 2019

Sorry Paul. I was not aware that your issues relate to Debug build. I was testing with a Release build.

amdrexu removed their assignment on 25 Sep 2019

paulthomson commented on 11 Nov 2019 Collaborator Author

Appears to be fixed!

paulthomson closed this on 11 Nov 2019

GPUOpen-Drivers / llpc

Code Issues 30 Pull requests 6 Actions Projects 7

dEQP-VK.graphicsfuzz.modf-temp-modf-color #213

New issue

Closed paulthomson opened this issue on 13 Sep 2019 · 5 comments

paulthomson commented on 13 Sep 2019 Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

modf-temp-modf-color.zip

amdrexu self-assigned this on 17 Sep 2019

kuhar self-assigned this on 18 Sep 2019

amdrexu removed their assignment on 20 Sep 2019

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

5 participants

kuhar commented on 20 Sep 2019 Collaborator

Not whether this input spv is valid.

The crash happens when translating the last OpExtInst:

```
               OpCapability Shader
          %1 = OpExtInstImport "GLSL.std.450"
               OpMemoryModel Logical GLSL450
               OpEntryPoint Fragment %main
"main" %_GLF_color
               OpExecutionMode %main
OriginUpperLeft
               OpSource ESSL 310
               OpName %main "main"
               OpName %temp "temp"
               OpName %_GLF_color "_GLF_color"
               OpDecorate %_GLF_color Location 0
       %void = OpTypeVoid
          %6 = OpTypeFunction %void
      %float = OpTypeFloat 32
    %v4float = OpTypeVector %float 4
    %float_1 = OpConstant %float 1
         %10 = OpConstantComposite %v4float
%float_1 %float_1 %float_1 %float_1
%_ptr_Function_v4float = OpTypePointer Function
%v4float
    %float_0 = OpConstant %float 0
         %13 = OpConstantComposite %v4float
%float_1 %float_0 %float_0 %float_1
                  ...
         %17 = OpExtInst %v4float %1 Modf %13
%_GLF_color     <====
               OpReturn
               OpFunctionEnd
```

The second argument to `Modf` is in address space 65, instead of 5. Due to type mismatch

This seems similar to the issue reported here: KhronosGroup/SPIRV-Cross#1123.

If the input spv is valid, would it be enough to just introduce an address space cast before passing the `%_GLF_color` to `Modf` ?

trenouf commented on 21 Sep 2019 Collaborator

I wonder if this is fixed by my outstanding PR #169, which reimplements Modf with a separate ModfStruct and store in the spir-v reader.

I guess the problem at the moment might be that the modf is a library function, so it eventually expands to have a store, but not in time for LowerGlobal to spot that it is a store to an output.

MarcinKantochAMD commented on 24 Sep 2019

I ran the test today and it is passing on our LLPC stack.

MarcinKantochAMD commented on 24 Sep 2019

Sorry Paul. I was not aware that your issues relate to Debug build. I was testing with a Release build.

kuhar removed their assignment on 31 Oct 2019

paulthomson commented on 11 Nov 2019 Collaborator Author

Appears to be fixed!

paulthomson closed this on 11 Nov 2019

GPUOpen-Drivers / llpc

Code | Issues 30 | Pull requests 6 | Actions | Projects 7

dEQP-VK.graphicsfuzz.two-loops-matrix #215

New issue

Closed **paulthomson** opened this issue on 13 Sep 2019 · 6 comments

paulthomson commented on 13 Sep 2019 Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

two-loops-matrix.zip

amdrexu assigned **xuechen417** on 17 Sep 2019

Assignees

kuhar

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

5 participants

kuhar commented on 17 Sep 2019 • edited ▾ Collaborator

This fails in llpcCompiler.cpp:220:

```
class LlpcDiagnosticHandler : public
llvm::DiagnosticHandler
{
    bool handleDiagnostics(const DiagnosticInfo&
diagInfo) override
    {
        if (EnableOuts() || EnableErrs())
        {
            if ((diagInfo.getSeverity() ==
DS_Error) || (diagInfo.getSeverity() ==
DS_Warning))
            {
                DiagnosticPrinterRawOStream
printStream(outs());
                printStream << "ERROR: LLVM
DIAGNOSIS INFO: ";
                diagInfo.print(printStream);
                printStream << "\n";
                outs().flush();
            }
           ...
        }
==>     LLPC_ASSERT(diagInfo.getSeverity() !=
DS_Error);
        return true;
    }
};
```

With the error message: `unsupported dynamic alloca`. Not sure what the intention was here -- perhaps to emit diagnostics and terminate on `DS_Error` s? In that case, I'd expect to see something like `llvm::report_fatal_error` used instead of the assertion.

Similar applies to issue #214.

linqun commented on 18 Sep 2019 Collaborator

please check whether there are some alloca in the middle of function. if yes, please try to find which pass add it and try to move it to the begin of function.

kuhar self-assigned this on 20 Sep 2019

kuhar mentioned this issue on 23 Sep 2019

Do not clone allocas in PeepholeOpt #234 Merged

amdrexu unassigned **xuechen417** on 24 Sep 2019

MarcinKantochAMD commented on 24 Sep 2019

Hi Paul. I tested today and can't reproduce the issue anymore.

MarcinKantochAMD commented on 24 Sep 2019

Sorry Paul. I was not aware that your issues relate to Debug build. I was testing with a Release build.

kuhar commented on 9 Oct 2019 Collaborator

This issue is fixed and can be closed.

paulthomson commented on 11 Nov 2019 Collaborator Author

Appears to be fixed!

paulthomson closed this on 11 Nov 2019

dEQP-VK.graphicsfuzz.two-loops-set-struct #216

New issue

Closed paulthomson opened this issue on 13 Sep 2019 · 5 comments

paulthomson commented on 13 Sep 2019 Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

two-loops-set-struct.zip

amdrexu assigned **xuechen417** on 17 Sep 2019

jaebaek commented on 23 Sep 2019 Collaborator

I cannot reproduce the failure. I want to doublecheck if this is already fixed.

paulthomson commented on 24 Sep 2019 • edited Collaborator Author

I used a Debug build on dev branch.

MarcinKantochAMD commented on 24 Sep 2019

Hi Paul. I tested today and can't reproduce the issue.

Assignees

s-perron

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

5 participants

MarcinKantochAMD commented on 24 Sep 2019

Sorry Paul. I was not aware that your issues relate to Debug build. I was testing with a Release build.

amdrexu unassigned **xuechen417** on 25 Sep 2019

s-perron self-assigned this on 25 Sep 2019

paulthomson commented on 11 Nov 2019 Collaborator Author

Appears to be fixed!

paulthomson closed this on 11 Nov 2019

GPUOpen-Drivers / llpc

Code · Issues 30 · Pull requests 6 · Actions · Projects 7

dEQP-VK.graphicsfuzz.discard-in-loop-in-function #330

New issue

Closed · paulthomson opened this issue on 11 Nov 2019 · 1 comment

paulthomson commented on 11 Nov 2019 · Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

discard-in-loop-in-function.zip

kuhar mentioned this issue on 12 Nov 2019

dEQP-VK.graphicsfuzz.two-nested-do-whiles #333 · Closed

paulthomson commented on 15 Nov 2019 · Collaborator · Author

Closing as this apparently does not reproduce.

paulthomson closed this on 15 Nov 2019

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

1 participant

GPUOpen-Drivers / **llpc**

Code · Issues 30 · Pull requests 6 · Actions · Projects 7

dEQP-VK.graphicsfuzz.discard-in-loop #331

New issue

Closed **paulthomson** opened this issue on 11 Nov 2019 · 2 comments

paulthomson commented on 11 Nov 2019 Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

discard-in-loop.zip

paulthomson commented on 15 Nov 2019 Collaborator Author

Closing as this apparently does not reproduce.

paulthomson closed this on 15 Nov 2019

kuhar commented on 16 Nov 2019 Collaborator

Sidenote: this does not validate:

```
ERROR: Fails to validate SPIR-V:
error: 59: block <ID> 37[%37] exits the
selection headed by <ID> 11[%11], but not via a
structured exit
  %37 = OpLabel
```

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

2 participants

GPUOpen-Drivers / **llpc**

Code | **Issues 30** | Pull requests 6 | Actions | Projects 7

dEQP-VK.graphicsfuzz.return-float-from-while-loop #332

New issue

Closed **paulthomson** opened this issue on 11 Nov 2019 · 11 comments

paulthomson commented on 11 Nov 2019 Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

return-float-from-while-loop.zip

nhaehnle commented on 12 Nov 2019 Collaborator

Hi Paul, aside from the fact that this is probably not a bug on the latest dev branch based on discussions on other bugs, I'd also ask you in the future to take a step back and produce higher quality bug reports.

For instance, **how** does the test fail? Does it hit an assertion or other error in the compiler? Does it generate incorrect code? Since when do these failures occur? (CTS tests are usually passing since CTS testing is performed regularly, so you should be able to find a regression point.)

Furthermore, shotgunning lots of issues like you did helps nobody. Once you've collected more information such as suggested in the previous paragraph, you will most likely find that many or even all of the issues are likely to have the same root cause (for example, they fail to compile with the same error message). It is then better to open a single issue with that error message as the title, and provide a mention of the various CTS failures that exhibit the problem in the issue body. Thank you!

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

4 participants

paulthomson commented on 12 Nov 2019

Collaborator Author

Apologies for the poor quality bug reports; I was indeed being slightly lazy here, as the process is not automated. Apart from not providing the log, I also forgot to add that I was using a debug build of amdllpc when running these tests.

We are occasionally fuzzing amdllpc and creating dEQP test cases that expose the issues. The tests will always start with `dEQP-VK.graphicsfuzz.` . But we fuzz and add tests for many tools and devices other than LLPC.

If you are indeed testing the Khronos dEQP Gerrit master branch (and perhaps pending CLs) on the debug build of the dev branch of the AMDVLK driver, then there is probably no point in me reporting these. However, I believe the debug build was possibly not being used in the past and so certain assertion failures were being missed, but perhaps this has changed.

Apologies also for creating one issue per test; other projects requested that we do this, but if you are running the tests, you will see the failures eventually anyway.

Can I ask: what is the best way to test the latest version of LLPC? Is building the dev branch of AMDVLK a fairly good approach? Is there a large delay between AMDVLK being updated to use the latest dev branch commit of LLPC? Also, feel free to let me know if assertion failures in the debug build should indeed be treated as bugs; if not, I can just use the release build.

nhaehnle commented on 13 Nov 2019 Collaborator

I think bug reports in general are fine, as things do slip through cracks... for example, I believe our internal automatic testing currently doesn't have LLPC/LLVM assertions enabled (let alone runnign with anything like address sanitizer...). And thank you for the fuzzing!

Assertion failures in the compiler should absolutely be treated as bugs. Otherwise we'll just go mad over time ;)

> Can I ask: what is the best way to test the latest version of LLPC? Is building the dev branch of AMDVLK a fairly good approach? Is there a large delay between AMDVLK being updated to use the latest dev branch commit of LLPC?

I'll have to punt to **@amdrexu @linqun @trenouf** on some of the details here. I suspect it's fine to use the dev branch of AMDVLK but manually track the LLPC dev branch. That really ought to work reasonably fine, and if it doesn't, we should think about how to improve the situation.

amdrexu commented on 13 Nov 2019 Collaborator

I think building latest AMDVLK dev plus latest LLPC dev is enough. LLPC has versioning mechanism. Latest LLPC can work with not-latest AMDVLK.

paulthomson commented on 15 Nov 2019 Collaborator Author

Thanks!

paulthomson commented on 15 Nov 2019 Collaborator Author

Closing as this apparently does not reproduce.

paulthomson closed this on 15 Nov 2019

kuhar commented on 16 Nov 2019 Collaborator

Sidenote: this does not validate:

```
ERROR: Fails to validate SPIR-V:
error: 69: block <ID> 100[%100] exits the
selection headed by <ID> 91[%91], but not via a
structured exit
  %100 = OpLabel
```

paulthomson commented on 17 Nov 2019 Collaborator Author

Ah this now makes sense. The failures were all validation "errors" from the old, internal version of spirv-val. The shaders validate with a newer version of spirv-val.

I could disable validation when I run amdllpc, but I chose to leave the validation on so that we would still see these "failures" and to give an incentive to update the validator version in AMDVLK. I guess the shaders won't actually be validated at run time though?

kuhar commented on 18 Nov 2019 Collaborator

> I could disable validation when I run amdllpc, but I chose to leave the validation on so that we would still see these "failures" and to give an incentive to update the validator version in AMDVLK. I guess the shaders won't actually be validated at run time though?

IMO it would make sense to report inputs that pass spirv-val but crash llpc with assertions/sanitizers enabled. I suppose that in real-life scenarios all shaders are known ahead of time and can be validated offline.

paulthomson commented on 18 Nov 2019 Collaborator Author

Yeah, makes sense.

E.g.

```
$ amdllpc discard-continue-
return.variant_fragment_shader.frag.spv
ERROR: Fails to validate SPIR-V:
error: 67: block <ID> 34[%34] exits the
selection headed by <ID> 35[%35], but not via a
structured exit
  %34 = OpLabel


ERROR:
=====  AMDLLPC FAILED  =====
```

vs.

```
$ amdllpc -val=false discard-continue-
return.variant_fragment_shader.frag.spv
```

The old, buggy version of spirv-val is always run for me unless I pass `-val=false` . I will add `-val=false` from now on.

amdrexu commented on 19 Nov 2019 Collaborator

I think you can use latest codes of SPIRV-tools and build a spirv-val. Try it and if the error still exists we might report an issue ticket on SPIRV-tools repo. If the issue is gone, we will update imported SPIRV-tools source codes in spvgen.

GPUOpen-Drivers / **llpc**

Code Issues 30 Pull requests 6 Actions Projects 7

dEQP-VK.graphicsfuzz.two-nested-do-whiles #333

New issue

Closed **paulthomson** opened this issue on 11 Nov 2019 · 5 comments

paulthomson commented on 11 Nov 2019 Collaborator

This test fails. To reproduce: `amdllpc -gfxip=9.0.0 -verify-ir -o temp.out *.spv`

two-nested-do-whiles.zip

kuhar commented on 12 Nov 2019 Collaborator

This works for me on on the dev branch with assertions enabled. **@paulthomson**

kuhar commented on 12 Nov 2019 Collaborator

All the new issues (#330-#334) compile fine for me. Not sure if the issues are still reproducible.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

2 participants

GPUOpen-Drivers / **llpc**

Code | Issues 30 | Pull requests 6 | Actions | Projects 7

dEQP-VK.graphicsfuzz.unreachable-continue-statement #334

New issue

Closed **paulthomson** opened this issue on 11 Nov 2019 · 3 comments