Bug 111069 - Assertion fails in nir_opt_remove_phis.c during compilation of SPIR-V shader

Status:	RESOLVED FIXED	**Reported:**	2019-07-04 13:47 UTC by Alastair Donaldson
		Modified:	2019-09-09 10:15 UTC (History)
Alias:	None	**CC List:**	5 users (show)
Product:	Mesa	**See Also:**	
Component:	Drivers/Vulkan/intel (show other bugs)		
Version:	git		
Hardware:	x86-64 (AMD64) All		
Importance:	medium normal		
Assignee:	Caio Marcelo de Oliveira Filho		
QA Contact:	Intel 3D Bugs Mailing List		
URL:			
Whiteboard:			
Keywords:	bisected, regression		
Depends on:			
Blocks:	mesa-19.2		

Show dependency tree / graph

Attachments	
Amber test exposing the problem (3.37 KB, text/plain) 2019-07-04 13:47 UTC, Alastair Donaldson	Details
backtrace_for_the_issue (9.48 KB, text/plain) 2019-07-04 18:29 UTC, Denis	Details
	View All

Alastair Donaldson 2019-07-04 13:47:41 UTC Description

```
Created attachment 144702 [details]
Amber test exposing the problem

Running the attached test using Amber (https://github.com/google/amber):

amber do-while-loop-in-conditionals.amber

should lead to the test passing.

Instead, an assertion fails:

amber: ../src/compiler/nir/nir_opt_remove_phis.c:115: remove_phis_block: Assertion
`def != NULL' failed.

Build: Mesa 19.2.0-devel (git-243db4980c) (Debug)
Device: Intel(R) HD Graphics 630 (Kaby Lake GT2)
```

Denis 2019-07-04 18:29:50 UTC Comment 1

```
Created attachment 144705 [details]
```

backtrace_for_the_issue

Hi, thanks for bug report. I made a bisect and attached backtrace with debug symbols.
Also want to note, that bisect lead to these two commits:

```
[den@den-pc mesa]$ git bisect good
There are only 'skip'ped commits left to test.
The first bad commit could be any of:
dc349f07b5f9c257f68ecf0cbb89f9722a42233a
b3c6146925595ec3a7eece3afb9ccaad32906d4c
We cannot bisect more!
```

Commit dc349f07b5f9c257f68ecf0cbb89f9722a42233a failed to build, and commit below looked the first "bad" commit then

```
commit b3c6146925595ec3a7eece3afb9ccaad32906d4c (HEAD)
Author: Caio Marcelo de Oliveira Filho <caio.oliveira@intel.com>
Date:   Fri Sep 14 11:41:39 2018 -0700

    nir: Copy propagation between blocks

    Extend the pass to propagate the copies information along the control
    flow graph.  It performs two walks, first it collects the vars
    that were written inside each node. Then it walks applying the copy
    propagation using a list of copies previously available.  At each node
    the list is invalidated according to results from the first walk.

    This approach is simpler than a full data-flow analysis, but covers
    various cases.  If derefs are used for operating on more memory
    resources (e.g. SSBOs), the difference from a regular pass is expected
    to be more visible -- as the SSA copy propagation pass won't apply to
    those.

    A full data-flow analysis would handle more scenarios: conditional
    breaks in the control flow and merge equivalent effects from multiple
    branches (e.g. using a phi node to merge the source for writes to the
    same deref).  However, as previous commentary in the code stated, its
    complexity 'rapidly get out of hand'.  The current patch is a good
    intermediate step towards more complex analysis.

    The 'copies' linked list was modified to use util_dynarray to make it
    more convenient to clone it (to handle ifs/loops).

    Annotated shader-db results for Skylake:

        total instructions in shared programs: 15105796 -> 15105451 (<.01%)
        instructions in affected programs: 152293 -> 151948 (-0.23%)
        helped: 96
        HURT: 17

            All the HURTs and many HELPs are one instruction.  Looking
            at pass by pass outputs, the copy prop kicks in removing a
            bunch of loads correctly, which ends up altering what other
            other optimizations kick.  In those cases the copies would be
            propagated after lowering to SSA.

            In few HELPs we are actually helping doing more than was
            possible previously, e.g. consolidating load_uniforms from
            different blocks.  Most of those are from
            shaders/dolphin/ubershaders/.

        total cycles in shared programs: 566048861 -> 565954876 (-0.02%)
```

```
        cycles in affected programs: 151461830 -> 151367845 (-0.06%)
        helped: 2933
        HURT: 2950

            A lot of noise on both sides.

        total loops in shared programs: 4603 -> 4603 (0.00%)
        loops in affected programs: 0 -> 0
        helped: 0
        HURT: 0

        total spills in shared programs: 11085 -> 11073 (-0.11%)
        spills in affected programs: 23 -> 11 (-52.17%)
        helped: 1
        HURT: 0

            The shaders/dolphin/ubershaders/12.shader_test was able to
            pull a couple of loads from inside if statements and reuse
            them.

        total fills in shared programs: 23143 -> 23089 (-0.23%)
        fills in affected programs: 2718 -> 2664 (-1.99%)
        helped: 27
        HURT: 0

            All from shaders/dolphin/ubershaders/.

        LOST:   0
        GAINED: 0

    The other generations follow the same overall shape.  The spills and
    fills HURTs are all from the same game.

    shader-db results for Broadwell.

        total instructions in shared programs: 15402037 -> 15401841 (<.01%)
        instructions in affected programs: 144386 -> 144190 (-0.14%)
        helped: 86
        HURT: 9

        total cycles in shared programs: 600912755 -> 600902486 (<.01%)
        cycles in affected programs: 185662820 -> 185652551 (<.01%)
        helped: 2598
        HURT: 3053

        total loops in shared programs: 4579 -> 4579 (0.00%)
        loops in affected programs: 0 -> 0
        helped: 0
        HURT: 0

        total spills in shared programs: 80929 -> 80924 (<.01%)
        spills in affected programs: 720 -> 715 (-0.69%)
        helped: 1
        HURT: 5

        total fills in shared programs: 93057 -> 93013 (-0.05%)
        fills in affected programs: 3398 -> 3354 (-1.29%)
        helped: 27
        HURT: 5

        LOST:   0
        GAINED: 2

:
        total cycles in shared programs: 600912755 -> 600902486 (<.01%)
```

```
        cycles in affected programs: 185662820 -> 185652551 (<.01%)
        helped: 2598
        HURT: 3053

        total loops in shared programs: 4579 -> 4579 (0.00%)
        loops in affected programs: 0 -> 0
        helped: 0
        HURT: 0

        total spills in shared programs: 80929 -> 80924 (<.01%)
        spills in affected programs: 720 -> 715 (-0.69%)
        helped: 1
        HURT: 5

        total fills in shared programs: 93057 -> 93013 (-0.05%)
        fills in affected programs: 3398 -> 3354 (-1.29%)
        helped: 27
        HURT: 5

        LOST:   0
        GAINED: 2

    shader-db results for Haswell:

        total instructions in shared programs: 9231975 -> 9230357 (-0.02%)
        instructions in affected programs: 44992 -> 43374 (-3.60%)
        helped: 27
        HURT: 69

        total cycles in shared programs: 87760587 -> 87727502 (-0.04%)
        cycles in affected programs: 7720673 -> 7687588 (-0.43%)
        helped: 1609
        HURT: 1416

        total loops in shared programs: 1830 -> 1830 (0.00%)
        loops in affected programs: 0 -> 0
        helped: 0
        HURT: 0

        total spills in shared programs: 1988 -> 1692 (-14.89%)
        spills in affected programs: 296 -> 0
        helped: 1
        HURT: 0

        total fills in shared programs: 2103 -> 1668 (-20.68%)
        fills in affected programs: 438 -> 3 (-99.32%)
        helped: 4
        HURT: 0

        LOST:   0
        GAINED: 1

    v2: Remove the DISABLE prefix from tests we now pass.

    v3: Add comments about missing write_mask handling. (Caio)
        Add unreachable when switching on cf_node type. (Jason)
:
        total cycles in shared programs: 600912755 -> 600902486 (<.01%)
        cycles in affected programs: 185662820 -> 185652551 (<.01%)
        helped: 2598
        HURT: 3053

        total loops in shared programs: 4579 -> 4579 (0.00%)
        loops in affected programs: 0 -> 0
        helped: 0
```

```
        HURT: 0

        total spills in shared programs: 80929 -> 80924 (<.01%)
        spills in affected programs: 720 -> 715 (-0.69%)
        helped: 1
        HURT: 5

        total fills in shared programs: 93057 -> 93013 (-0.05%)
        fills in affected programs: 3398 -> 3354 (-1.29%)
        helped: 27
        HURT: 5

        LOST:   0
        GAINED: 2

    shader-db results for Haswell:

        total instructions in shared programs: 9231975 -> 9230357 (-0.02%)
        instructions in affected programs: 44992 -> 43374 (-3.60%)
        helped: 27
        HURT: 69

        total cycles in shared programs: 87760587 -> 87727502 (-0.04%)
        cycles in affected programs: 7720673 -> 7687588 (-0.43%)
        helped: 1609
        HURT: 1416

        total loops in shared programs: 1830 -> 1830 (0.00%)
        loops in affected programs: 0 -> 0
        helped: 0
        HURT: 0

        total spills in shared programs: 1988 -> 1692 (-14.89%)
        spills in affected programs: 296 -> 0
        helped: 1
        HURT: 0

        total fills in shared programs: 2103 -> 1668 (-20.68%)
        fills in affected programs: 438 -> 3 (-99.32%)
        helped: 4
        HURT: 0

        LOST:   0
        GAINED: 1

    v2: Remove the DISABLE prefix from tests we now pass.

    v3: Add comments about missing write_mask handling. (Caio)
        Add unreachable when switching on cf_node type. (Jason)
        Properly merge the component information in written map
        instead of replacing. (Jason)
        Explain how removal from written arrays works. (Jason)
        Use mode directly from deref instead of getting the var. (Jason)

    v4: Register the local written mode for calls. (Jason)
        Prefer cf_node instead of node. (Jason)
        Clarify that remove inside iteration only works in backward
        iterations. (Jason)

    Reviewed-by: Jason Ekstrand <jason@jlekstrand.net>
```

Jason Ekstrand 2019-07-11 00:00:28 UTC Comment 2

Ugh... This one gets sticky. The problem is a rather strange sequence of events:

 1. nir_opt_dead_cf cleans up the outer `if (true) {`.
 2. nir_opt_if figures out the condition inside condition and turns `while (gl_FragCoord.x < 0.0)` into `while (1)`.
 3. nir_opt_dead_cf cleans up an if statement with the break that was the only exit from a loop.
 4. Because the deleted break was the only exit of the loop, now unreachable code later in the shader still consumes values generated by the loop (dead_cf only cleans up phis, not things which exit the dead code without a phi).
 5. Loop unrolling comes along and attempts to partially unroll the loop (I think? I'm not actually sure on this one.) and inserts LCSSA phis which have no sources.
 6. nir_opt_remove_phis dies on the existence of phis with zero sources.

So what do we do about it? Technically, 4. is ok because the way dominance is defined, unreachable code is vacuously dominated by all other code in the shader. I see a few options:

 1. Detect when we've deleted the last exit of a loop and simply delete the entire loop immediately in nir_opt_dead_cf.
 2. Don't generate LCSSA phis if the loop exit is unreachable
 3. Make nir_opt_remove_phis replace phis with zero sources with ssa_undef instructions
 4. All of the above?

Thoughs?

Alastair Donaldson 2019-08-16 16:24:26 UTC Comment 3

Would there be any disadvantage to just doing option 3? That sounds like an elegant solution (though I don't know the code).

andrii simiklit 2019-08-20 09:55:42 UTC Comment 4

There is one more issue related to the infinity loops without exits: ~~bug111405~~

I implemented option 1. there and it helped for both issue:
https://gitlab.freedesktop.org/asimiklit/mesa/commits/fix/loop_with_no_exits

But I am not sure whether it is a correct way to fix it.
What do you think about it?

andrii simiklit 2019-08-29 08:47:20 UTC Comment 5

Option 1. is under discussion here:
https://gitlab.freedesktop.org/mesa/mesa/merge_requests/1717

Mark Janes 2019-09-05 15:26:41 UTC Comment 6

new proposal has been implemented to fix this:

https://gitlab.freedesktop.org/mesa/mesa/merge_requests/1827

andrii simiklit 2019-09-09 09:53:12 UTC Comment 7

This issue doesn't reproduce on the latest master, was fixed by MR:
https://gitlab.freedesktop.org/mesa/mesa/merge_requests/1827

Juan A. Suarez 2019-09-09 10:15:27 UTC Comment 8

MR!1827 has been merged in master

Bug 111070 - Processing of SPIR-V shader leads to segmentation fault

Status:	RESOLVED MOVED	**Reported:**	2019-07-04 22:54 UTC by Alastair Donaldson
		Modified:	2019-09-18 19:50 UTC (History)
Alias:	None	**CC List:**	1 user (show)
Product:	Mesa	**See Also:**	
Component:	Drivers/Vulkan/intel (show other bugs)		
Version:	git		
Hardware:	Other All		
Importance:	medium normal		
Assignee:	Intel 3D Bugs Mailing List		
QA Contact:	Intel 3D Bugs Mailing List		
URL:			
Whiteboard:			
Keywords:			
Depends on:			
Blocks:			

Attachments	
Amber test exposing the problem (5.28 KB, text/plain) 2019-07-04 22:54 UTC, Alastair Donaldson	Details
backtrace_unreachable-loops-in-switch.amber_debug (46.47 KB, text/plain) 2019-07-05 07:19 UTC, Denis	Details
	View All

Alastair Donaldson 2019-07-04 22:54:16 UTC Description

```
Created attachment 144706 [details]
Amber test exposing the problem

Running the attached test using Amber (https://github.com/google/amber):

amber unreachable-loops-in-switch.amber

should lead to the test passing.

Instead, a segmentation fault occurs, with a backtrace starting in
libvulkan_intel.so.

Build: Mesa 19.2.0-devel (git-243db4980c) (Debug)
Device: Intel(R) HD Graphics 630 (Kaby Lake GT2)

Bug found using GraphicsFuzz.
```

Denis 2019-07-05 07:19:45 UTC Comment 1

```
Created attachment 144709 [details]
backtrace_unreachable-loops-in-switch.amber_debug

hi, this one crashes starting from 18.0 mesa up to the master.
```

Core dump with debug symbols attached

asimiklit 2019-07-09 09:09:09 UTC Comment 2

I am investigating this issue.

asimiklit 2019-07-16 14:05:00 UTC Comment 3

The issue was found in a 'nir_opt_dead_cf'. This optimization does not optimize all dead blocks (at least by a single invocation). For example on the following nir the condition 'if ssa_5' was optimized but 'if ssa_97' wasn't, due to issue in 'dead_cf_list'. MR which should fix this issue was suggested:
https://gitlab.freedesktop.org/mesa/mesa/merge_requests/1352

```
shader: MESA_SHADER_FRAGMENT
inputs: 0
outputs: 0
uniforms: 0
shared: 0
decl_var shader_out INTERP_MODE_NONE vec4 _GLF_color (FRAG_RESULT_DATA0, 0, 0)
decl_function main (0 params)

impl main {
        decl_var  INTERP_MODE_NONE int i
        decl_var  INTERP_MODE_NONE float[1] data
        decl_var  INTERP_MODE_NONE bool fall
        decl_var  INTERP_MODE_NONE bool cont
        decl_var  INTERP_MODE_NONE bool cont@0
        decl_var  INTERP_MODE_NONE bool cont@1
        decl_var  INTERP_MODE_NONE vec4 out@_GLF_color-temp
        block block_0:
        /* preds: */
        vec1 32 ssa_0 = load_const (0x00000001 /* 0.000000 */)
        vec1 32 ssa_1 = load_const (0x40000000 /* 2.000000 */)
        vec1 1 ssa_2 = load_const (true)
        vec1 32 ssa_3 = load_const (0x00000000 /* 0.000000 */)
        vec1 32 ssa_4 = load_const (0x3f800000 /* 1.000000 */)
        vec1 1 ssa_5 = load_const (false)
        vec4 32 ssa_8 = load_const (0x3f800000 /* 1.000000 */, 0x00000000 /*
0.000000 */, 0x00000000 /* 0.000000 */, 0x3f800000 /* 1.000000 */)
        /* succs: block_1 */
        loop {
                block block_1:
                /* preds: block_0 block_23 */
                vec1 1 ssa_13 = phi block_0: ssa_5, block_23: ssa_2
                vec1 32 ssa_14 = phi block_0: ssa_3, block_23: ssa_88
                vec1 32 ssa_19 = iadd ssa_14, ssa_0
                vec1 32 ssa_88 = bcsel ssa_13, ssa_19, ssa_14
                vec1 1 ssa_26 = ilt ssa_88, ssa_0
                /* succs: block_2 block_3 */
                if ssa_26 {
                        block block_2:
                        /* preds: block_1 */
                        /* succs: block_4 */
                } else {
                        block block_3:
                        /* preds: block_1 */
                        break
                        /* succs: block_24 */
                }
                block block_4:
                /* preds: block_2 */
                vec1 32 ssa_29 = deref_var &data (function_temp float[1])
```

```
                vec1 32 ssa_30 = deref_array &(*ssa_29)[ssa_88] (function_temp
float) /* &data[ssa_88] */
                vec1 32 ssa_31 = intrinsic load_deref (ssa_30) (0) /* access=0 */
                vec1 32 ssa_34 = deref_array &(*ssa_29)[0] (function_temp float) /*
&data[0] */
                vec1 32 ssa_35 = intrinsic load_deref (ssa_34) (0) /* access=0 */
                vec1 1 ssa_36 = flt ssa_31, ssa_35
                /* succs: block_5 block_22 */
                if ssa_36 {
                        block block_5:
                        /* preds: block_4 */
                        /* succs: block_6 block_7 */
                        if ssa_5 {
                                block block_6:
                                /* preds: block_5 */
                                vec1 32 ssa_39 = i2f32 ssa_88
                                vec1 1 ssa_40 = fge ssa_39, ssa_4
                                /* succs: block_8 */
                        } else {
                                block block_7:
                                /* preds: block_5 */
                                /* succs: block_8 */
                        }
                        block block_8:
                        /* preds: block_6 block_7 */
                        vec1 1 ssa_95 = load_const (false)
                        vec1 1 ssa_96 = load_const (false)
                        vec1 1 ssa_97 = load_const (false)
                        /* succs: block_9 block_17 */
                        if ssa_97 {
                                block block_9:
                                /* preds: block_8 */
                                /* succs: block_10 */
                                loop {
                                        block block_10:
                                        /* preds: block_9 block_13 */
                                        vec1 1 ssa_54 = phi block_9: ssa_5,
block_13: ssa_2
                                        /* succs: block_11 block_12 */
                                        if ssa_2 {
                                                block block_11:
                                                /* preds: block_10 */
                                                /* succs: block_13 */
                                        } else {
                                                block block_12:
                                                /* preds: block_10 */
                                                break
                                                /* succs: block_14 */
                                        }
                                        block block_13:
                                        /* preds: block_11 */
                                        continue
                                        /* succs: block_10 */
                                }
                                block block_14:
                                /* preds: block_12 */
                                /* succs: block_15 */
                                loop {
                                        block block_15:
                                        /* preds: block_14 block_15 */
                                        vec1 1 ssa_61 = phi block_14: ssa_5,
block_15: ssa_2
                                        continue
                                        /* succs: block_15 */
                                }
```

Bug 111071 - SPIR-V shader processing fails with message about "extra dangling SSA sources"

Status:	RESOLVED FIXED	**Reported:**	2019-07-04 23:01 UTC by Alastair Donaldson
		Modified:	2019-07-15 20:00 UTC (History)
Alias:	None	**CC List:**	1 user (show)
Product:	Mesa	**See Also:**	
Component:	Drivers/Vulkan/intel (show other bugs)		
Version:	git		
Hardware:	Other All		
Importance:	medium normal		
Assignee:	Intel 3D Bugs Mailing List		
QA Contact:	Intel 3D Bugs Mailing List		
URL:			
Whiteboard:			
Keywords:			
Depends on:			
Blocks:			

Attachments	
Amber test exposing the problem (4.52 KB, text/plain) 2019-07-04 23:01 UTC, Alastair Donaldson	Details
bactrace (6.08 KB, text/plain) 2019-07-05 08:04 UTC, Denis	Details
Rectified test (4.52 KB, text/plain) 2019-07-10 22:57 UTC, Alastair Donaldson	Details
	View All

Alastair Donaldson 2019-07-04 23:01:05 UTC Description

```
Created attachment 144707 [details]
Amber test exposing the problem

Running the attached test using Amber (https://github.com/google/amber):

amber nested-ifs-and-return-in-for-loop.amber

should lead to the test passing.

Instead, Amber crashes and the following is emitted:

extra dangling SSA sources:
0x561df6f9fb88
0x561df6f9ff48
0x561df6f9f7c8
0x561df6fa0308
Aborted

It looks like this message comes from src/compiler/nir/nir_validate.c.
```

```
Build: Mesa 19.2.0-devel (git-243db4980c) (Debug)
Device: Intel(R) HD Graphics 630 (Kaby Lake GT2)

Interestingly, using Mesa 18.0.3, the test does not crash, but fails due to an
incorrect colour being probed from the image that is rendered.
```

Denis 2019-07-05 08:04:27 UTC Comment 1

```
Created attachment 144710 [details]
bactrace

added backtrace from
Mesa 19.2.0-devel (git-0cc02c9ea6)


and confirming the fail on earlier mesa (18.0.0)

[den@den-pc Debug]$ ./amber ~/Downloads/nested-ifs-and-return-in-for-loop.amber
/home/den/Downloads/nested-ifs-and-return-in-for-loop.amber: Line 159: Probe failed
at: 0, 0
  Expected RGBA: 255.000000, 0.000000, 0.000000, 255.000000
  Actual RGBA: 255.000000, 255.000000, 255.000000, 255.000000
Probe failed in 1 pixels

Summary of Failures:
  /home/den/Downloads/nested-ifs-and-return-in-for-loop.amber

Summary: 0 pass, 1 fail
```

Sergii Romantsov 2019-07-05 14:46:28 UTC Comment 2

```
will try to look on that
```

Jason Ekstrand 2019-07-10 20:18:08 UTC Comment 3

```
https://gitlab.freedesktop.org/mesa/mesa/merge_requests/1312

That fixes the validation fail and gets us back to rendering the wrong color.
Haven't debugged that issue yet.
```

Jason Ekstrand 2019-07-10 22:25:21 UTC Comment 4

Also, are you sure that test is correct? It's doing

```
> void main()
> {
>  _GLF_color = vec4(1.0, 0.0, 0.0, 1.0);
>  for(
>      int i = 0;
>      i < 10;
>      i ++
>  )
>   {
>    _GLF_color = vec4(1.0);
>    if(1.0 > injectionSwitch.y)
>     {
>      _GLF_color = vec4(1.0, 0.0, 0.0, 1.0);
>      if(true)
>       {
>        return;
>       }
>     }
```

```
>    }
> }
```

but the input data it sets up has injectionSwitch.y == 1.0 so the if will fail because 1.0 > 1.0 is false and so _GLF_color will get overwritten to vec4(1.0) and never written back to red. Our compiler seems to be compiling it correctly.

Alastair Donaldson 2019-07-10 22:57:05 UTC Comment 5

Created attachment 144756 [details]
Rectified test

Thanks for fixing the validation issue, Jason.

My bad regarding the probe in the test - I overlooked that write of vec4(1.0) when writing the probe.

You are correct that white should indeed be rendered. The attached test case exposes the original bug (the error), and this time really should pass (and indeed does pass on my Mesa 18.0.5 driver).

Jason Ekstrand 2019-07-15 20:00:15 UTC Comment 6

Fixed by the following commit now in master:

```
commit 7a19e05e8c84152af3a15868f5ef781142ac8e23
Author: Jason Ekstrand <jason@jlekstrand.net>
Date:   Wed Jul 10 15:14:42 2019 -0500

    nir/opt_if: Clean up single-src phis in opt_if_loop_terminator

    Bugzilla: https://bugs.freedesktop.org/show_bug.cgi?id=111071
    Fixes: 2a74296f24ba "nir: add opt_if_loop_terminator()"
    Reviewed-by: Timothy Arceri <tarceri@itsqueeze.com>
```

Bug 111075 - Processing of SPIR-V shader causes device hang, sometimes leading to system reboot

Status:	RESOLVED FIXED	**Reported:**	2019-07-05 10:52 UTC by Alastair Donaldson
		Modified:	2019-07-16 23:28 UTC (History)
Alias:	None	**CC List:**	2 users (show)
Product:	Mesa	**See Also:**	
Component:	Drivers/Vulkan/intel (show other bugs)		
Version:	git		
Hardware:	Other All		
Importance:	medium normal		
Assignee:	Ian Romanick		
QA Contact:	Intel 3D Bugs Mailing List		
URL:			
Whiteboard:			
Keywords:	bisected, regression		
Depends on:			
Blocks:			

Attachments	
Amber test exposing the problem (4.88 KB, text/plain) 2019-07-05 10:52 UTC, Alastair Donaldson	Details
output_after_test_running (4.22 KB, text/plain) 2019-07-05 13:26 UTC, Denis	Details
	View All

Alastair Donaldson 2019-07-05 10:52:52 UTC — Description

Created attachment 144711 [details]
Amber test exposing the problem

Running the attached test using Amber (https://github.com/google/amber):

amber return-in-loop-in-function.amber

should lead to the test passing.

Instead, a GPU hang is reported, and this sometimes leads to a system reboot.

Build: Mesa 19.2.0-devel (git-243db4980c) (Debug)
Device: Intel(R) HD Graphics 630 (Kaby Lake GT2)

Interestingly, using Mesa 18.0.3, the GPU hang does not occur; instead there is a segmentation fault in libvulkan_intel.so.

Denis 2019-07-05 13:26:36 UTC — Comment 1

Created attachment 144712 [details]
output_after_test_running

hey, looks like you opened pandora box with a lot of issues.
Adding one more bisect result below. (and attaching test output during the hang)
upd - interesting that

mesa-18.0.0 - test crashes
mesa-18.3.0 - test passes
mesa-git - test hangs the system

8fb8ebfbb05d3323481c8ba6d320b3a3580bad99 is the first bad commit
commit 8fb8ebfbb05d3323481c8ba6d320b3a3580bad99
Author: Ian Romanick <ian.d.romanick@intel.com>
Date: Tue May 22 18:56:41 2018 -0700

    intel/compiler: More peephole select

    Shader-db results:

    The one shader hurt for instructions is a compute shader that had both
    spills and fills hurt.

Danylo 2019-07-11 10:20:41 UTC Comment 2

I think I'm close to finding the issue which results in shader using uninitialized memory which in turn leads to hang.

Danylo 2019-07-12 15:05:45 UTC Comment 3

I'm unable to come up with any seemingly good solution at the moment so here what I have found:

The hang happens due to the usage of uninitialized memory in calculations of loop terminator.
Bisected commit (8fb8ebfbb05d3323481c8ba6d320b3a3580bad99) just makes other optimization go wrong.

The usage of uninitialized memory starts with nir_lower_regs_to_ssa_impl which happens after opt_if_cf_list. During the nir_lower_regs_to_ssa_impl have (pay attention to r23):

```
> decl_reg vec1 32 r23
> ...
> /* succs: block_1 */
> loop {
>     block block_1:
>     /* preds: block_0 block_12 */
>     ...
>     /* succs: block_5 */
>     loop {
>         block block_5:
>         /* preds: block_4 block_9 block_11 */
>         vec1 32 ssa_32 = phi block_4: ssa_5, block_9: r23, block_11: ssa_168
>         r23 = phi block_4: ssa_5, block_9: r27, block_11: ssa_170
>         ...
>         /* succs: block_6 block_7 */
>         if r25 {
>             block block_6:
>             /* preds: block_5 */
>             ...
>             break
>             /* succs: block_12 */
>         } else {
>             block block_7:
```

```
>             /* preds: block_5 */
>             /* succs: block_8 */
>         }
>         block block_8:
>         /* preds: block_7 */
>         vec1 32 ssa_47 = i2f32 r23
>         /* succs: block_9 block_10 */
>         if r26 {
>             block block_9:
>             /* preds: block_8 */
>             r27 = iadd r23, ssa_5
>             continue
>             /* succs: block_5 */
>         } else {
>             block block_10:
>             /* preds: block_8 */
>             break
>             /* succs: block_12 */
>         }
>         block block_11:
>         /* preds: */
>         /* succs: block_5 */
>     }
>     block block_12:
>     /* preds: block_6 block_10 */
>     ...
>     /* succs: block_1 */
> }
```

r23 is used in calculations of ssa_32 thus when we are calling nir_phi_builder_value_get_block_def with block_9 and r23 in rewrite_src we are trying to find the closest ssa_def by traversing the dominance tree.

ssa_def is found in block_5 - the only place where r23 is assigned, however it is NEEDS_PHI since we haven't processed the r23 assignment yet. So empty phi is created, placed in the list and then added to all blocks between block_9 and block_5 in dominance tree.

Next, the r23 assignment is processed, nir_phi_builder_value_set_block_def is called with new ssa for this block however the later blocks e.g. block_8 still has empty phi for r23. So when assignment to ssa_47 is processed it fetches the empty phi which corresponds to this block for r23, now ssa_47 uses incorrect source:

```
> block block_5:
> /* preds: block_4 block_9 block_11 */
> vec1 32 ssa_32 = phi block_4: ssa_5, block_9: ssa_4294967295, block_11: ssa_168
> vec1 32 ssa_223 = phi block_4: ssa_5, block_9: ssa_4294967295, block_11: ssa_170
> /* succs: block_6 block_7 */
> if ssa_225 {
>     block block_6:
>     /* preds: block_5 */
>     break
>     /* succs: block_12 */
> } else {
>     block block_7:
>     /* preds: block_5 */
>     /* succs: block_8 */
> }
> block block_8:
> /* preds: block_7 */
> vec1 32 ssa_47 = i2f32 ssa_4294967295
```

Later in nir_phi_builder_finish "virtual" phi blocks are materialized not taking

into account the fact that r23 was assigned. So we get:

```
>block block_0:
>vec1 32 ssa_248 = undefined
>loop {
>       block block_1:
>    vec1 32 ssa_249 = phi block_0: ssa_248, block_12: ssa_223

>
>    loop {
>        block block_5:
>        /* preds: block_4 block_9 block_11 */
>        vec1 32 ssa_247 = phi block_4: ssa_249, block_9: ssa_247, block_11: ssa_24
>        vec1 32 ssa_32 = phi block_4: ssa_5, block_9: ssa_247, block_11: ssa_168
>        vec1 32 ssa_223 = phi block_4: ssa_5, block_9: ssa_255, block_11: ssa_170
>        /* succs: block_6 block_7 */
>        if ssa_225 {
>            block block_6:
>            /* preds: block_5 */
>            break
>            /* succs: block_12 */
>        } else {
>            block block_7:
>            /* preds: block_5 */
>            /* succs: block_8 */
>        }
>        block block_8:
>        /* preds: block_7 */
>        vec1 32 ssa_47 = i2f32 ssa_247
```

In this last NIR ssa_47 uses the value which is essentially ssa_248 during the first loop iteration and is undefined.

I thought that the initial NIR arrangement violates some unwritten rule but commenting out last block in nir_phi_builder_value_get_block_def resulted in a correct shader and no other code has issues with such NIR.

I'm unsure if it should take into account the situation described above in some way or if such NIR arrangement shouldn't happen.

Jason Ekstrand 2019-07-12 16:08:10 UTC Comment 4

Good triage work! The real problem here isn't with visiting phi destinations but visiting phi sources. In particular, they don't exist in the block with the phi but at the end of the predecessor block.

https://gitlab.freedesktop.org/mesa/mesa/merge_requests/1331

Jason Ekstrand 2019-07-16 23:28:58 UTC Comment 5

Fixed by the following commit in master:

```
commit 6fb685fe4b762c8030f86895707516e2481e9ece (HEAD -> master, origin/master,
origin/HEAD)
Author: Jason Ekstrand <jason@jlekstrand.net>
Date:   Fri Jul 12 11:01:40 2019 -0500

    nir/regs_to_ssa: Handle regs in phi sources properly

    Sources of phi instructions act as if they occur at the very end of the
    predecessor block not the block in which the phi lives.  In order to
```

```
handle them correctly, we have to skip phi sources on the normal
instruction walk and handle them as a separate walk over the successor
phis.  While registers in phi instructions is a bit of an oddity it can
happen when we temporarily go out-of-SSA for control-flow manipulations.

Bugzilla: https://bugs.freedesktop.org/show_bug.cgi?id=111075
Cc: mesa-stable@lists.freedesktop.org
Reviewed-by: Caio Marcelo de Oliveira Filho <caio.oliveira@intel.com>
```

```
                                block block_16:
                                /* preds: */
                                /* succs: block_18 */
                        } else {
                                block block_17:
                                /* preds: block_8 */
                                /* succs: block_18 */
                        }
                        block block_18:
                        /* preds: block_16 block_17 */
                        vec1 1 ssa_91 = load_const (true)
                        vec1 1 ssa_98 = load_const (true)
                        vec1 1 ssa_99 = load_const (true)
                        vec1 1 ssa_100 = load_const (true)
                        /* succs: block_19 block_20 */
                        if ssa_100 {
                                block block_19:
                                /* preds: block_18 */
                                intrinsic store_deref (ssa_34, ssa_1) (1, 0) /*
wrmask=x */ /* access=0 */
                                /* succs: block_21 */
                        } else {
                                block block_20:
                                /* preds: block_18 */
                                /* succs: block_21 */
                        }
                        block block_21:
                        /* preds: block_19 block_20 */
                        /* succs: block_23 */
                } else {
                        block block_22:
                        /* preds: block_4 */
                        /* succs: block_23 */
                }
                block block_23:
                /* preds: block_21 block_22 */
                continue
                /* succs: block_1 */
        }
        block block_24:
        /* preds: block_3 */
        vec1 32 ssa_81 = deref_var &_GLF_color (shader_out vec4)
        intrinsic store_deref (ssa_81, ssa_8) (15, 0) /* wrmask=xyzw */ /* access=0
*/
        /* succs: block_25 */
        block block_25:
}
```

Alastair Donaldson 2019-08-16 16:25:14 UTC Comment 4

Any thoughts on how to fix this one?

Juan A. Suarez 2019-08-30 08:08:28 UTC Comment 5

(In reply to Alastair Donaldson from comment #4)
> Any thoughts on how to fix this one?

https://gitlab.freedesktop.org/mesa/mesa/merge_requests/1717 fixes this issue too

Juan A. Suarez 2019-08-30 08:10:47 UTC Comment 6

(In reply to Juan A. Suarez from comment #5)
> (In reply to Alastair Donaldson from comment #4)
> > Any thoughts on how to fix this one?
>
> https://gitlab.freedesktop.org/mesa/mesa/merge_requests/1717 fixes this
> issue too

Sorry, forget this, my mistake

Juan A. Suarez 2019-08-30 08:19:53 UTC Comment 7

(In reply to Alastair Donaldson from comment #4)
> Any thoughts on how to fix this one?

https://gitlab.freedesktop.org/mesa/mesa/merge_requests/1352 should fix this

GitLab Migration User 2019-09-18 19:50:01 UTC Comment 8

-- GitLab Migration Automatic Message --

This bug has been migrated to freedesktop.org's GitLab instance and has been closed from further activity.

You can subscribe and participate further through the new bug through this link to our GitLab instance: https://gitlab.freedesktop.org/mesa/mesa/issues/846.

Bug 110953 - Adding a redundant single-iteration do-while loop causes different image to be rendered

Status:	RESOLVED FIXED	**Reported:**	2019-06-20 17:18 UTC by Abel Briggs
		Modified:	2019-06-25 18:41 UTC (History)
Alias:	None	**CC List:**	0 users
Product:	Mesa		
Component:	glsl-compiler (show other bugs)	**See Also:**	
Version:	git		
Hardware:	x86-64 (AMD64) Linux (All)		
Importance:	medium normal		
Assignee:	Ian Romanick		
QA Contact:	Intel 3D Bugs Mailing List		
URL:			
Whiteboard:			
Keywords:			
Depends on:			
Blocks:			

Attachments	
piglit shader tests, images, diff (2.02 KB, application/zip) 2019-06-20 17:18 UTC, Abel Briggs	Details
	View All

Abel Briggs 2019-06-20 17:18:42 UTC Description

```
Created attachment 144602 [details]
piglit shader tests, images, diff

Similar bug reproduced with Intel i965 that was resolved as fixed:
https://bugs.freedesktop.org/show_bug.cgi?id=100303

I'm unsure if the error reproduced with the enclosed shaders occur on AMD/Intel -
please feel free to move this if it can't be reproduced.

The attached archive contains two shaders, a reference shader and a variant shader,
that by construction should render the same image. On the build and PC specified
below, however, these shaders render different images. Images of the shaders'
output are also supplied in the archive, as well as the minor diff between the
source code of the two shaders.

The difference between the two shaders is that the variant wraps a single-iteration
do-while loop around existing code.

46c46
<       do
---
>       if(doSwap)

48,53c48,50
<       if(doSwap)
```

```
<         {
<             float temp = data[i];
<             data[i] = data[j];
<             data[j] = temp;
<         }
---
>         float temp = data[i];
>         data[i] = data[j];
>         data[j] = temp;

55d51
<       while(false);
```

Steps to reproduce:
---
1. Obtain and build piglit, the Mesa OpenGL test suite runner:
https://gitlab.freedesktop.org/mesa/piglit
2. Download the attached archive.
3. From a terminal, execute the supplied tests with the piglit GLES3 shader runner:
$ bin/shader_runner_gles3 reference.shader_test
$ bin/shader_runner_gles3 variant.shader_test

Expected results:

Both images should produce an image like reference.png, and so both tests should pass. If both fail, it's likely due to floating point errors that can affect the precise color values - if this is the case, please manually observe that the shader images have significant differences that are not limited to minor floating point variation.

Actual results:

The variant produces a different image (and fails the test) even though it should run exactly the same as the reference shader. This is because the only difference between the reference and variant shaders is that the variant shader wraps a single-iteration do-while loop around existing code, which makes no semantic difference between the two shaders.

Build & PC specs:

CPU: Intel Core i7-5820k
GPU: nVIDIA GTX 970

OS: Ubuntu 19.04
libdrm: git-5db0f7692d1fdf05f9f6c0c02ffa5a5f4379c1f3 (most recent as of this writing)
Mesa: git-9c19d07b1cdcd22ced0f4e1c147e496b6ff5cf23 (most recent as of this writing)
Xf86-video-nouveau: 1.0.16
Linux kernel version: 5.0.0-16-generic

This bug was found with GraphicsFuzz: https://github.com/google/graphicsfuzz

Ian Romanick 2019-06-20 19:45:17 UTC Comment 1

I'm not able to reproduce this bug on i965. It looks like llvmpipe (from Fedora 30, Mesa 19.0.5) produces the variant.png image. The other bug was in a GLSL IR pass that i965 doesn't use, so this one probably is too. I'll take a look.

Ian Romanick 2019-06-20 19:53:13 UTC Comment 2

The difference in the output of "./glsl_compiler --version 300 --dump-lir" for both shaders is below. It looks like the loop in the second shader is (incorrectly) unrolled twice.

```
--- /tmp/before.txt     2019-06-20 12:48:58.425392405 -0700
+++ /tmp/after.txt      2019-06-20 12:49:05.983437986 -0700
@@ -72,6 +72,14 @@
           )
           ())

+          (if (var_ref checkSwap_retval) (
+            (declare (temporary ) float assignment_tmp@2)
+            (assign  (x) (var_ref assignment_tmp@2)  (array_ref (var_ref data)
(var_ref i) ) )
+            (assign  (x) (array_ref (var_ref data) (var_ref i) )  (array_ref
(var_ref data) (var_ref j) ) )
+            (assign  (x) (array_ref (var_ref data) (var_ref j) )  (var_ref
assignment_tmp@2) )
+          )
+          ())
+
           (assign  (x) (var_ref j)  (expression int + (var_ref j) (constant int
(1)) ) )
         ))

@@ -87,12 +95,12 @@
         (assign  (xyzw) (var_ref _GLF_color)  (var_ref vec_ctor) )
       )
       (
-        (declare (temporary ) vec4 vec_ctor@2)
-        (assign  (w) (var_ref vec_ctor@2)  (constant float (1.000000)) )
-        (assign  (x) (var_ref vec_ctor@2)  (expression float / (array_ref (var_ref
data) (constant int (5)) ) (constant float (10.000000)) ) )
-        (assign  (y) (var_ref vec_ctor@2)  (expression float / (array_ref (var_ref
data) (constant int (9)) ) (constant float (10.000000)) ) )
-        (assign  (z) (var_ref vec_ctor@2)  (expression float / (array_ref (var_ref
data) (constant int (0)) ) (constant float (10.000000)) ) )
-        (assign  (xyzw) (var_ref _GLF_color)  (var_ref vec_ctor@2) )
+        (declare (temporary ) vec4 vec_ctor@3)
+        (assign  (w) (var_ref vec_ctor@3)  (constant float (1.000000)) )
+        (assign  (x) (var_ref vec_ctor@3)  (expression float / (array_ref (var_ref
data) (constant int (5)) ) (constant float (10.000000)) ) )
+        (assign  (y) (var_ref vec_ctor@3)  (expression float / (array_ref (var_ref
data) (constant int (9)) ) (constant float (10.000000)) ) )
+        (assign  (z) (var_ref vec_ctor@3)  (expression float / (array_ref (var_ref
data) (constant int (0)) ) (constant float (10.000000)) ) )
+        (assign  (xyzw) (var_ref _GLF_color)  (var_ref vec_ctor@3) )
       ))

     ))
```

Ian Romanick 2019-06-20 22:11:27 UTC [Comment 3]

Setting debug = true in do_common_optimization
(src/compiler/glsl/glsl_parser_extras.cpp), it seems the bad code appears in
do_dead_code_unlinked.

Ian Romanick 2019-06-20 22:19:44 UTC [Comment 4]

(In reply to Ian Romanick from comment #3)
> Setting debug = true in do_common_optimization
> (src/compiler/glsl/glsl_parser_extras.cpp), it seems the bad code appears in
> do_dead_code_unlinked.

Scratch that. It is unroll_loops, but unroll_loops doesn't log its progress the

way the other optimization passes do.

Ian Romanick 2019-06-20 22:35:39 UTC Comment 5

The problem is this bit of code loop_unroll_visitor::simple_unroll.

```
   if (limit_if != first_ir->as_if() || exit_branch_has_instructions)
      iterations++;
```

The check that the first thing in the loop is an if-statement expects that to only occur when the if-statement is a terminator for the loop (e.g., if (i > 6) break;). This loop starts with an if-statement that is not a terminator, so the loop iteration count is incorrectly incremented.

Ian Romanick 2019-06-21 05:17:34 UTC Comment 6

This MR should fix the bug:

https://gitlab.freedesktop.org/mesa/mesa/merge_requests/1152

And this MR has a minimal test case:

https://gitlab.freedesktop.org/mesa/piglit/merge_requests/88

I'm running both through Intel's CI on G33 hardware. That very, very old hardware has OpenGL ES 2.0, but it does not use the NIR loop unrolling path. It should hit this bug... I'll find out for sure in the morning. I tried the test and the fix on my local machine using softpipe.

@abel: Can you test the fix on nouveau?

Abel Briggs 2019-06-21 14:38:32 UTC Comment 7

(In reply to Ian Romanick from comment #6)
> This MR should fix the bug:
>
> https://gitlab.freedesktop.org/mesa/mesa/merge_requests/1152
>
> And this MR has a minimal test case:
>
> https://gitlab.freedesktop.org/mesa/piglit/merge_requests/88
>
> I'm running both through Intel's CI on G33 hardware. That very, very old
> hardware has OpenGL ES 2.0, but it does not use the NIR loop unrolling path.
> It should hit this bug... I'll find out for sure in the morning. I tried
> the test and the fix on my local machine using softpipe.
>
> @abel: Can you test the fix on nouveau?

I pulled your repo and ran the shader, and I can confirm that the issue is fixed (the reference and variant now render the same image).

Ian Romanick 2019-06-25 18:41:38 UTC Comment 8

Fixed by:

```
commit ee1c69faddb3624ace6548dafaff50549a031380
Author: Ian Romanick <ian.d.romanick@intel.com>
Date:   Thu Jun 20 15:48:48 2019 -0700

    glsl: Don't increase the iteration count when there are no terminators
```

```
    Incrementing the iteration count was intended to fix an off-by-one error
    when the first terminator was superseded by a later terminator.  If
    there is no first terminator or later terminator, there is no off-by-one
    error.  Incrementing the loop count creates one.  This can be seen in
    loops like:

        do {
            if (something) {
                // No breaks or continues here.
            }
        } while (false);

    Reviewed-by: Timothy Arceri <tarceri@itsqueeze.com>
    Tested-by: Abel Briggs <abelbriggs1@hotmail.com>
    Bugzilla: https://bugs.freedesktop.org/show_bug.cgi?id=110953
    Fixes: 646621c66da ("glsl: make loop unrolling more like the nir unrolling
path")
```

Format For Printing - Top of page

Bug 111006 - Adding a uniform-dependent if-statement in shader renders a different image

Status:	RESOLVED MOVED	**Reported:**	2019-06-26 16:39 UTC by Abel Briggs
		Modified:	2019-09-18 20:48 UTC (History)
Alias:	None	**CC List:**	0 users
Product:	Mesa		
Component:	Drivers/DRI/nouveau (show other bugs)	**See Also:**	
Version:	git		
Hardware:	x86-64 (AMD64) Linux (All)		
Importance:	medium normal		
Assignee:	Karol Herbst		
QA Contact:	Nouveau Project		
URL:	https://trello.com/c/ESKBoRGN/215-cod...		
Whiteboard:			
Keywords:			
Depends on:			
Blocks:			

Attachments	
Reproduction shader_test files, images and diff (41.57 KB, application/zip) 2019-06-26 16:39 UTC, Abel Briggs	Details
	View All

Abel Briggs 2019-06-26 16:39:09 UTC Description

Created attachment 144639 [details]
Reproduction shader_test files, images and diff

The attached archive contains two shaders, a reference shader and a variant shader, that by construction should render the same image. On the build and PC specified below, however, these shaders render different images. Images of the shaders' output are also supplied in the archive, as well as the minor diff between the source code of the two shaders.

The difference between the two shaders is that the variant saves the current value of GLF_color, then sets it to a random value. Then, depending on a condition that is always true at runtime, GLF_color is overwritten with its original value.

```
11a12,13
> uniform vec2 injectionSwitch;
>

20,22c22,23
<   float width = 256.0;
<   float c_re = 0.8 * (xCoord - width / 2.0) * 4.0 / width - 0.4;
<   float c_im = 0.8 * (yCoord - 256.0 / 2.0) * 4.0 / width;
---
>   float c_re = 0.8 * (xCoord - 256.0 / 2.0) * 4.0 / 256.0 - 0.4;
>   float c_im = 0.8 * (yCoord - 256.0 / 2.0) * 4.0 / 256.0;
```

```
32a34,40
>       vec4 _GLF_outVarBackup_GLF_color;
>       _GLF_outVarBackup_GLF_color = _GLF_color;
>       _GLF_color = vec4(2799.7886, 1762.3462, - 8612.3800, - 7.8);
>       if(((0.0 < injectionSwitch.y)))
>        {
>         _GLF_color = _GLF_outVarBackup_GLF_color;
>        }

79a88
>

80a90
> uniform vec2 injectionSwitch 0.0 1.0
```

The value of injectionSwitch is set to (0.0, 1.0), ensuring that the conditional (0.0 < injectionSwitch.y) is always true at runtime. Notably, if you remove injectionSwitch and replace the condition with 'true', the variant renders the same image as the reference.

Steps to reproduce:

Obtain and build piglit, the Mesa OpenGL test suite runner:
https://gitlab.freedesktop.org/mesa/piglit
Download the attached archive.
From a terminal, execute the supplied tests with the piglit GLES3 shader runner:
$ bin/shader_runner_gles3 reference.shader_test
$ bin/shader_runner_gles3 variant.shader_test

Expected results:

Both tests should produce an image like reference.png.

Actual results:

The variant produces a different image even though it should run exactly the same as the reference shader. Even though the variant swaps out GLF_color and swaps it back in, the two shaders are semantically equivalent, and there should be no difference in their rendering. Despite this, the variant renders many of the colored pixels in the image as transparent.

Build & PC specs:

CPU: Intel Core i7-5820k
GPU: nVIDIA GTX 970

OS: Ubuntu 19.04
libdrm: git-5db0f7692d1fdf05f9f6c0c02ffa5a5f4379c1f3
Mesa: git-d4575c3071 (most recent as of this writing)
Xf86-video-nouveau: 1.0.16
Linux kernel version: 5.0.0-16-generic

This bug was found with GraphicsFuzz: https://github.com/google/graphicsfuzz

Michel Dänzer 2019-06-27 07:30:19 UTC Comment 1

Can't reproduce with radeonsi or llvmpipe/softpipe => seems to be a nouveau issue.

Paul 2019-06-27 09:29:15 UTC Comment 2

Hi Abel.

Could you provide your configure command, please?

Karol Herbst 2019-06-27 13:43:15 UTC Comment 3

this is caused by a "call/return stack" overflow. We know of another shader causing this issue in a game, so it's something we probably want to fix.

Abel Briggs 2019-06-27 14:46:00 UTC Comment 4

If you're referring to the Meson(In reply to Paul from comment #2)
> Hi Abel.
>
> Could you provide your configure command, please?

If you mean the Meson configure command, the only things I changed from the default is that I enabled ASAN with -Db_sanitize=address and I set the prefix to be a different location(along with the libdrm I compiled) so I wouldn't mess up my normal graphics stack. I compiled with gcc 8.3.0.

I'm not sure of any other configure commands you may mean, so please let me know if this isn't what you're looking for.

GitLab Migration User 2019-09-18 20:48:11 UTC Comment 5

-- GitLab Migration Automatic Message --

This bug has been migrated to freedesktop.org's GitLab instance and has been closed from further activity.

You can subscribe and participate further through the new bug through this link to our GitLab instance: https://gitlab.freedesktop.org/mesa/mesa/issues/1183.
