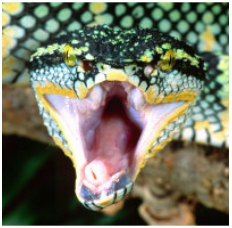


# Tools and Debugging

ocpihdl, ocpixml, and ocpi\_debug

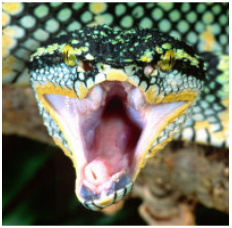
# Outline

1. Collect HDL device/worker information: `ocpihdl` and `ocpixmap`
2. Review worker lifecycle
3. Control worker state with `ocpihdl`
4. Change and view worker properties
5. Introduce `ocpi_debug`



# Using ocpihdl

- We can use ocpihdl to:
  - Probe the xml of the currently loaded bitstream
  - Display currently loaded workers
  - View/change worker status
  - Set/change worker properties
  - Reset workers
  - Step through an application
  - ...
- Without options, ocpihdl prints out all of its uses



# Available HDL devices on Matchstiq-Z1



## \$ ocpihdl search

- Provides information for available HDL devices
  - Name
  - Date of bitstream creation
  - Platform
  - Part
  - UUID

```
% ocpihdl search
```

```
OpenCPI HDL device found: 'PL:0': bitstream date Mon May 1 08:59:51 2017, platform "matchstiq_z1",  
part "xc7z020", UUID 10226754-2e6e-11e7-88f7-573bc8e8124c
```

## \$ ocpihdl probe

- Similar to “search”, but you can specify a device

# Loading/Probing the Bitstream



- Examine the XML packaged into *any* artifact (including bitstreams):  
`$ ocpixml get <.so or .bitz artifact filename>`
- Load a bitstream (*e.g.* on the Matchstiq-Z1):  
`$ ocpihdl load <path to bitstream>`
- Examine the XML packaged into a *currently loaded* bitstream:  
`$ ocpihdl getxml <output-file>`

# Loading/Probing the Bitstream



```
$ ocpihdl getxml <output-file>
```

```
$ grep controlOperation <output-file>
```

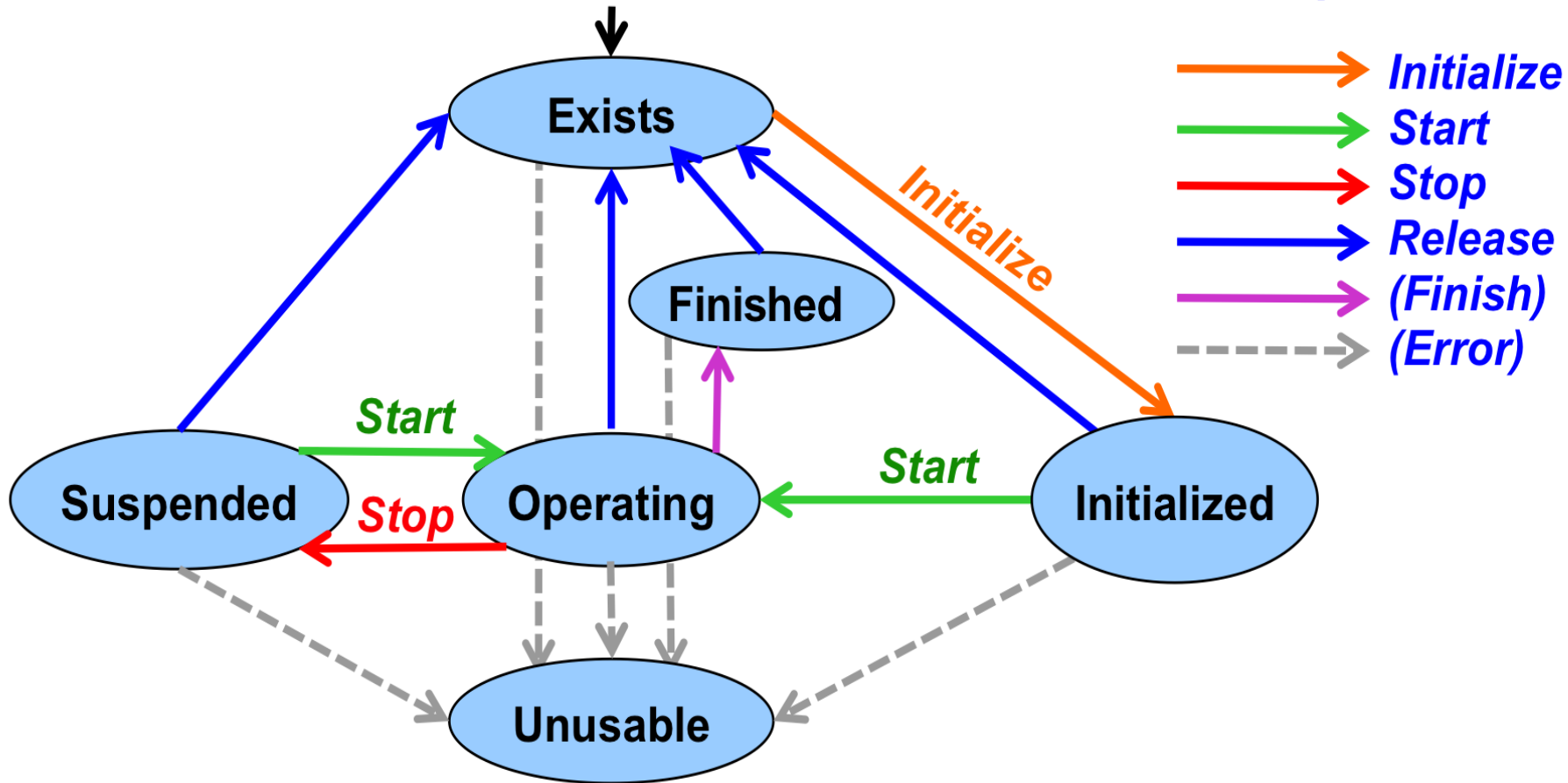
- What controlOperations are implemented by each worker?

```
% grep controlOperation output.xml | tail -4
<worker name="pattern" model="hdl" package="ocpi" specname="ocpi.pattern" sizeOfConfigSpace="21
47483712" controlOperations="initialize">
<worker name="bias" model="hdl" package="ocpi" specname="ocpi.bias" sizeOfConfigSpace="42949672
96" controlOperations="initialize" Timeout="444">
<worker name="capture" model="hdl" package="ocpi" specname="ocpi.capture" sizeOfConfigSpace="21
47487744" controlOperations="initialize">
<worker name="ocscp" model="hdl" package="ocpi" specname="ocpi.ocscp" controlOperations="stop">
```

- If you ever want to remove the bitstream from the FPGA:

```
$ ocpihdl unload
```

# Worker LifeCycle



# Controlling Workers

- Find worker index and other information

\$ ocpihdl get

- Can specify <index/worker-name> (and even <property>) here as well

- Reset or Unreset workers

\$ ocpihdl wreset/wunreset <index>

- Perform other control operations on workers

\$ ocpihdl wop <index> start/initialize/stop

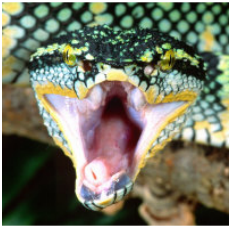
- Set worker properties

\$ ocpihdl set <index/worker-name> <property> <value>

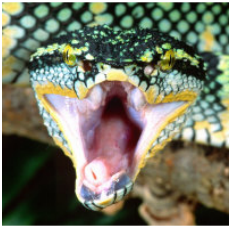
- Observe worker state

\$ ocpihdl status <index/worker-name>

- Add -v to see worker properties







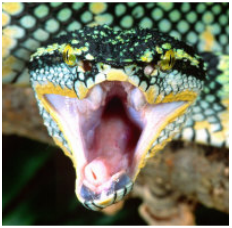
# View Worker Information from Bitstream

## \$ ocpihdl get

```
% ocpihdl get
HDL Device: 'PL:0' is platform 'matchstiq_z1' part 'xc7z020' and UUID '10226754-2e6e-1
1e7-88f7-573bc8e8124c'
Platform configuration workers are:
  Instance p/matchstiq_z1 of io worker matchstiq_z1 (spec ocp.platform) with index 0
  Instance p/time_server of io worker time_server (spec ocp.devices.time_server) with
index 1
Container workers are:
  Instance c/ocscp of normal worker ocscp (spec ocp.ocscp)
  Instance c/unoc_term0_0 of io worker sdp_term (spec ocp.devices.sdp_term)
  Instance c/unoc_term1_0 of io worker sdp_term (spec ocp.devices.sdp_term)
  Instance c/unoc_term2_0 of io worker sdp_term (spec ocp.devices.sdp_term)
  Instance c/unoc_term3_0 of io worker sdp_term (spec ocp.devices.sdp_term)
  Instance c/tb_bias_wti0_time_client of normal worker time_client (spec ocp.time_cli
ent)
  Instance c/metadata of normal worker metadata (spec ocp.metadata)
Application workers are:
  Instance a/pattern of normal worker pattern (spec ocp.pattern) with index 2
  Instance a/bias of normal worker bias (spec ocp.bias) with index 3
  Instance a/capture of normal worker capture (spec ocp.capture) with index 4
```

- We can identify them by their indices

# Controlling Workers



- If the worker is 'RESET'

\$ ocpihdl wunreset <index>

```
% ocpihdl status 2
```

```
Status of instance 'a/pattern' of worker 'pattern' is 'RESET'
```

```
Worker 2 on device pl:0
```

```
Status: 0x00008000
```

```
Control: 0x00000000 not enabled (reset asserted); timeout value is 1
```

```
ConfigAddr: 0x00000000
```

```
PageWindow: 0x00000000
```

```
Instance a/pattern of normal worker pattern (spec ocpi.pattern) with index 2
```

```
% ocpihdl wunreset 2
```

```
Worker 2 on device pl:0: reset deasserted, was asserted
```

```
% ocpihdl status 2
```

```
Status of instance 'a/pattern' of worker 'pattern' is 'EXISTS'
```

```
Worker 2 on device pl:0
```

```
Status: 0x00008000
```

```
Control: 0x80000000 enabled (reset not asserted); timeout value is 1
```

```
ConfigAddr: 0x00000000
```

```
PageWindow: 0x00000000
```

```
Instance a/pattern of normal worker pattern (spec ocpi.pattern) with index 2
```

# Controlling Workers



- If the worker implements the 'initialize' controlOperation:

\$ ocpihdl wop <index> initialize

```
% ocpihdl wop 2 initialize
Worker 2 on device pl:0: the 'initialize' control operation was performed with result 'success'
01)
% ocpihdl status 2
Status of instance 'a/pattern' of worker 'pattern' is 'INITIALIZED'
Worker 2 on device pl:0
Status:      0x00048000 opValid:0x0:init
Control:     0x80000000 enabled (reset not asserted); timeout value is 1
ConfigAddr: 0x00000000
PageWindow: 0x00000000
Instance a/pattern of normal worker pattern (spec ocp.pattern) with index 2
```

- Ultimately, to bring the worker to 'OPERATING':

\$ ocpihdl wop <index> start

# Controlling Workers



- Writable worker properties can be changed:  
\$ ocpihdl set <index/worker-name> <property> <value>
- Readable (the explicit flag, not the concept) / Volatile properties can be read:

\$ ocpihdl status -v <index/worker-name>

```
% ocpihdl set 2 step true
```

Setting the step property to 'true' on instance 'a/counter'

```
% ocpihdl status 2 ; ocpihdl get -v 2
```

Status of instance 'a/counter' of worker 'counter-1' is 'OPERATING'

Worker 2 on device pl:0

Status: 0x091f0000 addrValid beValid:0x1 opValid:0x1: start wrtValid:1

Control: 0x80000004 enabled (reset not asserted); timeout value is 16

ConfigAddr: 0x00000008

PageWindow: 0x00000000

Instance a/counter of normal worker counter-1 (spec local.counter) with index 2

0	counter: 2
1	max: 9
2	ocpi_debug: true
3	ocpi_endian: little
4	step: true

# Debugging



- `ocpi_debug`
  - Worker parameter that can enable debugging
  - Set to *true* in OWD before building workers for debugging
  - Add debugging functionality to worker that depends on `ocpi_debug`
- In VHDL: “`ocpi_debug`”

```
debug_gen : if its(ocpi_debug) generate
```
- In C++: “`<WORKER_NAME>_OCPI_DEBUG`”

```
if (COUNTER_OCPI_DEBUG) {
```

# Execution and Debug Utilities



- Log level
  - export OCPI\_LOG\_LEVEL=8
  - OCPI\_LOG\_LEVEL=10 make tests
  - ocpirun -l 8 (lowercase L)
- “ocpirun -C” finds containers, including simulators
- “ocpiview” is used to view simulation waveform results
- printf/cout
- ocp\_debug OWD parameter
- gdb