Software-defined design for systems of chiplets

Dr. Duncan Haldane
JITX
June 19th, 2022
Chiplet design work is snowballing

**DISAGGREGATION ENABLES**
- Increased yields
- 3D stacking
- Reduced time to market
- Mass customizability

<table>
<thead>
<tr>
<th>IFF WE CAN SOLVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Unfamiliar optimization problems</td>
</tr>
<tr>
<td>- Extra power and area for D2D interfaces</td>
</tr>
<tr>
<td>- Packaging cost and NRE</td>
</tr>
<tr>
<td>- Verification complications</td>
</tr>
</tbody>
</table>

**WHILE USING NEW**
- Advanced packages
- D2D interface standards

[Intel 2021]
Chiplet design work is snowballing

<table>
<thead>
<tr>
<th>DISAGGREGATION ENABLES</th>
<th>IFF WE CAN SOLVE</th>
<th>WHILE USING NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increased yields</td>
<td>Unfamiliar optimization problems</td>
<td>Advanced packages</td>
</tr>
<tr>
<td>3D stacking</td>
<td>Extra power and area for D2D interfaces</td>
<td></td>
</tr>
<tr>
<td>Reduced time to market</td>
<td>Packaging cost and NRE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verification complications</td>
<td>D2D interface standards</td>
</tr>
</tbody>
</table>

What should we ask of our design tools?
What should we ask of our design tools?

1. Rigorous design optimization integrating:
   1.1. Si design co-optimization
   1.2. Package design co-optimization
   1.3. Board design co-optimization

2. Automated design verification

3. Design reuse across package technologies

4. Enable what-if? analysis

$ git clone silicon-ip
$ git clone package-design-kits
$ git clone board-ip
$ make hpc42

[Sources Intel, ASE]
Team
3 PhD founders from UC Berkeley

Prior work
(10x design productivity for SoCs)

Funding
DARPA SEQUOIA
A compiler for systems of chiplets

ESIR: Electronic Systems Intermediate Representation (open-source)
A compiler for systems of chiplets

- Code
- ESIR
- Verification
- Generators
- Pkg. floorplanning
- Place and route
- SI / PI
- ...
pcb:module daughter:card (daisy:chain:In):
    port Rs485: Rs485
    port in : pin(daisy:chain)
    port out : pin(daisy:chain)
    inst conn : (ocdb/samtec/qtn/component[30])
    package(conn) at loc(0.0, 0.0) on Top
    net (Rs485.A, conn.p1)
    net (Rs485.P, conn.p2)
    for i in 0 to daisy:chain do :
        net (conn.p1 + i, in[i])
        net (conn.p2 + i, out[i])

pcb:module mother:card :
    inst power:jack : (ocdb/phoenix/combicon-cmc/component[2])
    net (and (power:jack.p1), power:jack.p2)
    set power:source(power:jack.p1, power:jack.p2, 0.0)
    inst pros : (ocdb/stmicroelectronics/STM32F427/module)
    inst ethernet:jack : (ocdb/pulse:electronics/J69-6009NL/component
        require rgmii:rgmii from pros
        require eth:ethernet:j69 from ethernet:jack
        connect phy (rgmii.eth)

SHRENA  OUTPUT  REGisu CONTROL  TERMINAL
Two chiplet boards generated from software

2500 pin
300um pitch
Chiplet compiler prototype

Transformations

Code → ESIR → Exports

Target: Chiplet network using Bunch of Wires

Organic substrate
Bunch of wires - Slice

```clojure
defn bunch-of-wires-slice-bumps [pitch:Double, rx?:True|False]

inst tx-slice   : bunch-of-wires-slice(1000.0e-3, false)
inst rx-slice   : bunch-of-wires-slice(1000.0e-3, true)
inst tiny-slice : bunch-of-wires-slice(40.0e-3, false)
```
Bunch of wires - Bundle and component

bundle bunch-of-wires :
  port D : pin[16]
  pin fec
  pin aux
  port clk : diff-pair

bunch-of-wires-slice (pitch:Double, rx?:True|False) :
  port BoW : bunch-of-wires

inst tx : bunch-of-wires-slice(130.0e-3, false)
inst rx : bunch-of-wires-slice(130.0e-3, true)
net (tx.BoW rx.BoW)
Bunch of wires - Stack

\[\text{bunch-of-wires-stack} \ (\text{depth:}\text{Int}, \ \text{pitch:}\text{Double}, \ \text{rx?:}\text{True|False}):\]

\[\text{inst rx-stack : bunch-of-wires-stack(4, } 130.0\text{e-3, true)}\]
Bunch of wires - Link

```clojure
(defn connect-bunch-of-wires [x:JITXObject, y:JITXObject]:
  inside pcb-module : 
    net (x.rx, y.tx)
    net (y.rx, x.tx)

bunch-of-wires-link-bumps (tx-size:Int, rx-size:Int, depth:Int, pitch:Double, flip?:True|False)

inst ayar-link : bunch-of-wires-link-bumps(16, 8, 4, 130.0e-3, true)
inst intel-link : bunch-of-wires-link-bumps(8, 16, 4, 130.0e-3, false)

place(ayar-link) at loc(0.0, 0.0) on Top
place(intel-link) at loc(0.0, -2.0, 180.0) on Top

connect-bunch-of-wires(ayar-link.io intel-link.io)
```
Bunch of wires - Link
Minimize routing depth and unused pins. Report power.

substrate-pitch = 60.0e-3
link = BoW-link( :
    bit-width = 100
    tx-clk = 1.0e6
    max-edge-width = 3.0
    source-terminated? = true)

optimize-link-parameters(link)

> Power per BoW link : 0.0001152W

substrate-pitch = 60.0e-3
link = BoW-link( :
    bit-width = 150
    tx-clk = 2.0e6
    max-edge-width = 7.5
    source-terminated? = true)

optimize-link-parameters(link)

> Power per BoW link : 0.0003456W
Bunch of wires - Chiplet

\[
\text{inst} \ \text{amd-25} : \ \text{chiplet} ("\text{AMD25}\", \ \text{false}, \ \text{false}, \ [16\ 16], \ [16\ 16], \ 4, \ 130.0\times10^{-3}, \ \text{Rectangle}(5.0,\ 10.0))[2]
\]

\[
\text{place}(\text{amd-25}[0]) \ \text{at} \ \text{loc}(2.5,\ 10.0) \ \text{on} \ \text{Top}
\]
\[
\text{place}(\text{amd-25}[1]) \ \text{at} \ \text{loc}(-2.5,\ 10.0) \ \text{on} \ \text{Top}
\]

\[
\text{connect-bunch-of-wires}(\text{amd-25}[0].\ \text{west-io} \ \text{amd-25}[1].\ \text{east-io})
\]
Next steps

Support more varieties of packages
- Stacking, interposers, EMIB, …

Add more export targets
- Bump map integration
- Glue logic for verification

Hook into upstream architecture optimization
Summary

We need design tools to support optimization for disaggregation

A software-defined approach integrates chiplet, package, and board design

We can now interactively generate and evaluate networks of chiplets

Contact information: Duncan Haldane - d.haldane@jitx.com
Questions?