OCAPI

http://www.imec.be/ocpai/

Maged Attia(mattia@ucla.edu)

Introduction to OCAPI

Outline
- OCAPI vs. ART Designer
- How does OCAPI work?
- OCAPI Classes
- Design Cycle in OCAPI
- Compiling C++/OCAPI
- Introduction to the case study.

Agenda

- Introduction to OCAPI (Wednesday)
- Case Study: Complete Design Flow of FIR (Friday)
- Homework: Design of LFSR (Friday)

OCAPI vs. ART Designer

<table>
<thead>
<tr>
<th>OCAPI</th>
<th>Art Designer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) C++ based</td>
<td>1) C/C++ based</td>
</tr>
<tr>
<td>2) Bit Parallel Arch.</td>
<td>2) Bit Parallel</td>
</tr>
<tr>
<td>3) \texttt{csh} \texttt{g++ file.cxx ...} \texttt{csh}&gt;</td>
<td>3) \texttt{...}</td>
</tr>
<tr>
<td>4) More flexible</td>
<td>4) More friendly</td>
</tr>
</tbody>
</table>

4) More flexible
How does OCAPI Work?

C++ Compiler

Add Mult Ram Sub

Link

C++ Compiler

howlib.h

OCAPI Classes

Description

include

libqlib.a

Run

OCAPI Classes: 1- Fixed Point

dfix a;

dfix a(0.25);

dfix a(0.25, 10, 8);

dfix a(0.25, 10, 8, tc, st, rd);

Default

{ tc: two’s complement
wp: for wrap-around
fl: for truncation
}

ns: unsigned
st: saturation
rd: rounding

OCAPI Classes: 2- IN/OUT

dfix a;

FB I("I");

src a("a", I, "in.dat");

_dfix b;

FB O("O");

snk b("b", O, "out.dat");

OCAPI Classes: 3- Internal Signal

_sig a("a");

plain signal with fl-p inside it

_sig b("b", dfix(0, 10, 8))

plain signal with fx-p inside it

_sig k(0.5);

Clk ck;

_registered signal C with initial value k
OCAPI Classes: 4- FSM

ctlfsm f; Creat FSM
F "my_fsm" Give it a name
State rst; Creat 2 states
State active;
Rst "my_rst" Give them name
Active "my_active" Tell which one is the default
F deflt(rst); Active always active;
Active _cnd(a) rst Transitions

Design Cycle in OCAPI

1) Un-Timed Design
- Easy and fast to design
- Non implementable
- Real description

2) Timed Design
- Implementable

Design cycle in OCAPI

C++ C++/OCAPI C++/OCAPI
Refine Refine Non implementable
Fixed point EASY
Function Description
Word-length

Design cycle in OCAPI

C++

Void add::add( int & in1
int & in2
int & out )
{ out = in1 + in2
return 0;
}

Void main()
{
Add Add(a, b, c);
}
Design cycle in OCAPI

```
void add::add(dfix & in1, dfix & in2, dfix & out) {
    if ((in1.getsize() < 1) || (......))
        return 0;
    out = in1.get() + in2.get()
    return 1;
}
```
Compiling C++/OCAPI

C++

```c
file:myfile.cxx
1: void main() {
2: }
```

```bash
csh> gcc myfile.cxx -o myfile
csh> myfile
```

Introduction to the case study

**Step 1:** Specifications -> Coefficients  
**Step 2:** Create floating point description for the whole system.  
**Step 3:** Create fixed point description for the UPS and the FIR and refine for best word length  
**Step 4:** Use that WL to create fx-p FSM description and run timed simulation to generate the VHDL code.