Using Vivado-HLS for Structural Design: a NoC Case Study

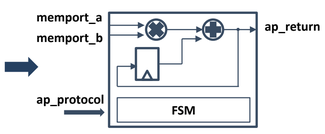
Zhipeng Zhao <zzhao1@andrew.cmu.edu>, **James C. Hoe** <jhoe@ece.cmu.edu>

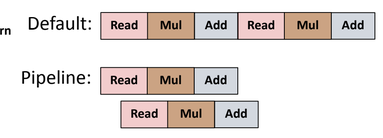
Should we use HLS for structural design?

- **Xilinx Vivado High Level Synthesis (HLS)**

```
int dot_product(
  int a[2], int b[2]){
  int i;
  int c = 0;

  for(i=0;i<2;i++){
#pragma HLS PIPELINE
    c += a[i] * b[i];
  }

  return c;
}
```

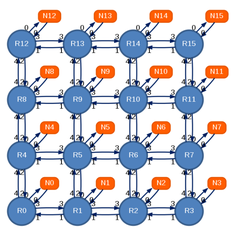
→ Vivado-HLS →



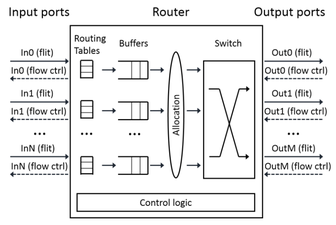


  - Algorithm to hardware synthesis
  - Automatic parallelization and scheduling
  - Pragma to steer mapping decisions

- **Structural Design:** ***Precise Cycle- and Bit-level Control***



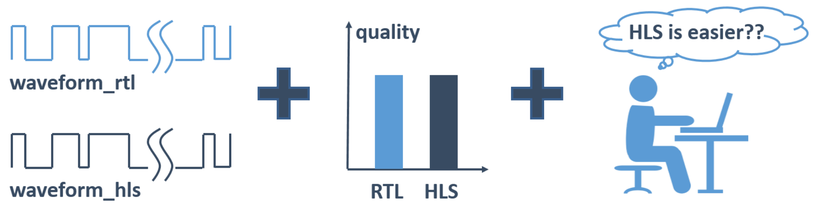
CONNECT NoC*



CONNECT Router Template*

  - **Examples:**
    Network-on-Chip (NoC), Routers…
  - **Challenge:**
    Use *untimed* and *sequential* C code to describe *cycle-accurate* and *concurrent* hardware.

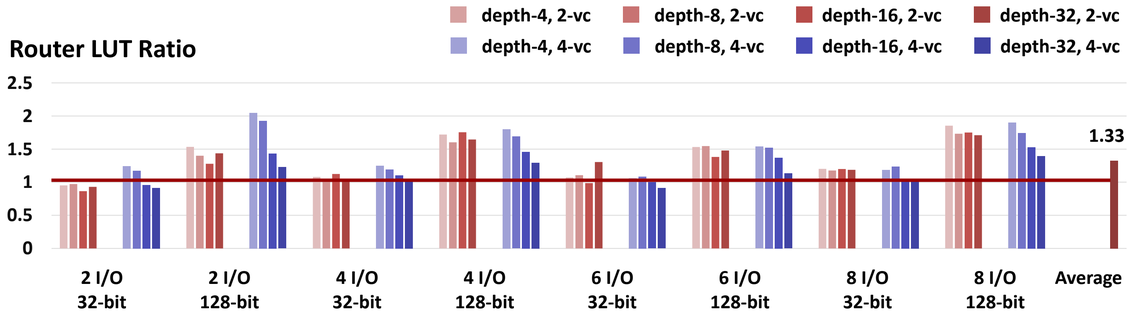
- ***Can We* and *Should We* Use Vivado-HLS for Structural Design?**

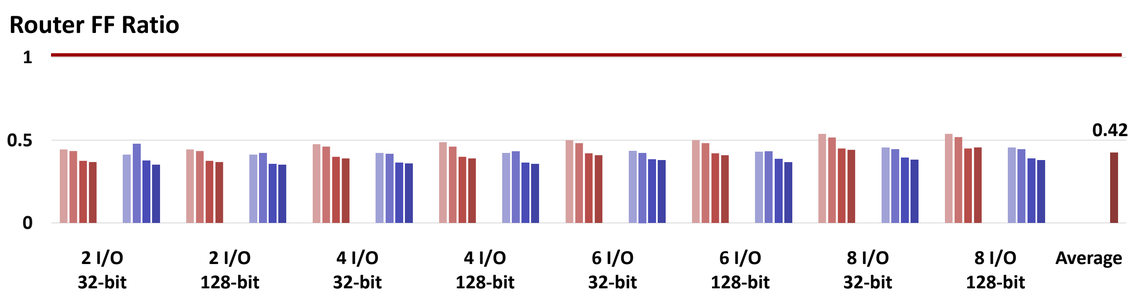


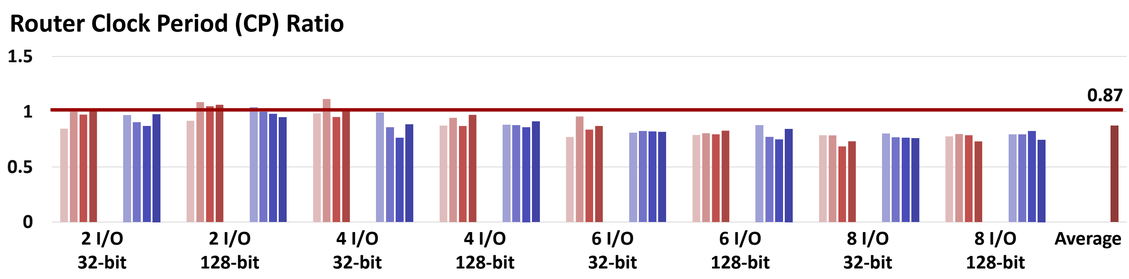
  - Exact structure and timing?
  - Same quality as RTL?
  - Improve productivity?

Can we? Yes; Should we? Depends.

- **Case Study Overview**
  - RTL Reference: CONNECT Parameterized NoC (ww.ece.cmu.edu/calcm/connect)
  - Productivity gain from separation of *functionality* and *structural details*. But offer little advantage in pure netlisting
  - Produced exact cycle- and bit-accurate replacements
  - Achieved comparable hardware cost and critical path
- **Productivity Improvement**
  - Natural C's sequential reading is maintained. Utilize C's facilities to capture more maintainable and scalable designs. *1126* lines of C++ VS. *2605* lines of Bluespec System Verilog
  - Still need to specify structural details to get desired hardware
  - Separation of functionality and structural details enables fast design iteration
  - Ordering discipline could be cumbersome when creating a netlist to compose submodules
- **Hardware Quality (HLS VS. CONNECT)**





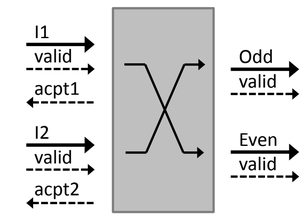


What worked very well

- **Illustrative example here; see website for complete router source**

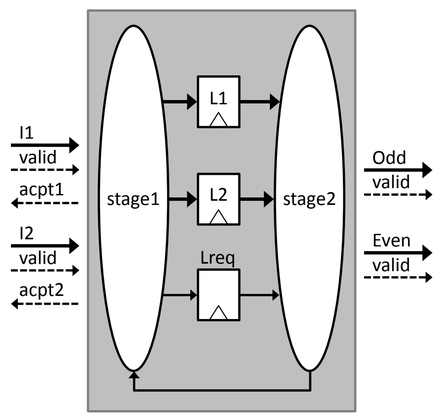
```
typedef struct vDat {bool v; int d;}

void switch_comb(vDat I1, vDat I2, vDat *Odd, vDat *Even,
                 bool *acpt1, bool *acpt2){
   *acpt1=*acpt2=(*Odd).v=(*Even).v=false;

   //For Odd
   if(I1.v && (I1.d%2))
      {(*Odd).v=true;(*Odd).d=I1.d;*acpt1=true;}
   else if (I2.v && (I2.d%2))
      {(*Odd).v=true;(*Odd).d=I2.d;*acpt2=true;}
   … repeat for Even …
}
```

switch_comb



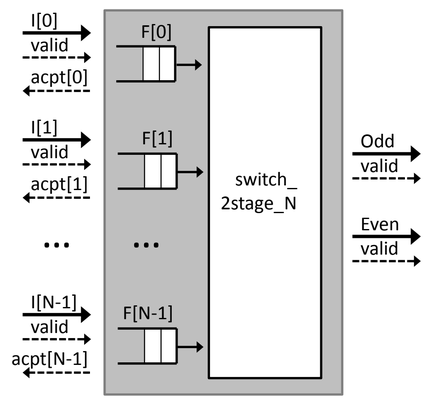

```
void switch_2stage(vDat I1, vDat I2, vDat *Odd,
                   vDat *Even, bool *acpt1, bool *acpt2){
   //Internal states
   static vData L1,L2;//Latch for I1 and I2
   static Path Lreq;//Latch for four possible connections

   /* --- stage 2 --- */
   Path grnt=allocate(Lreq);

   if (grnt.L1xOd) *Odd=L1;
   else if (grnt.L2xOd) *Odd=L2;
   else (*Odd).v=false;
   … repeat for Even …

   /* --- stage 1 --- */
   if (grnt.L1xOd||grnt.L1xEv)
      L1.v=false;
   if (I1.v && (!L1.v))
      {*acpt1=true; L1=I1;}
   else *acpt1=false;
   … repeat for I2 and L2 …

   Lreq=decode(L1,L2);
}
```

switch_2stage




```
void switch_buffered_N(vData I[N], vData *Odd,
                       vData *Even, bool acpt[N]){
#pragma HLS ARRAY_PARTITION variable=acpt complete dim=1
#pragma HLS ARRAY_PARTITION variable=I complete dim=1
   static FIFO<int> F[N];
   bool okX[N];
   vData frontX[N];

   for(int i=0;i<N;i++){
#pragma HLS UNROLL
      frontX[i].v=!F[i].empty(); frontX[i].d=F[i].front();
   }

   switch_2stage_N(frontX, Odd, Even, okX);

   for(int i=0;i<N;i++){
#pragma HLS UNROLL
      if(!F[i].full() && I[i].v)
         {F[i].push(I[i].d);acpt[i]=true;}
      else acpt[i]=false;
      if(okX[i]) F[i].pop();
   }
}
```

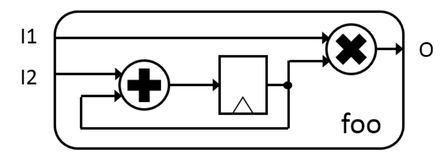
switch_buffered_N



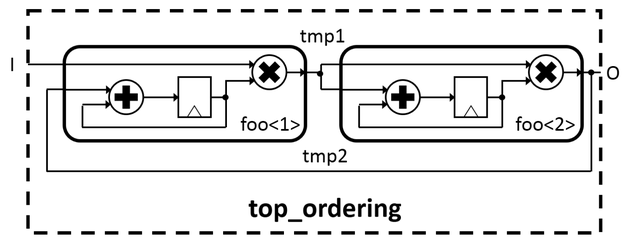
What didn't work so well

- ***Sequential* C Language => *Concurrent* Hardware?**

```
void foo(int I1, int I2, int *O){
   static int L = INIT_VAL; //latch

   *O = I1 * L; //read current-L
   L = I2 + L; //assign next-L
}
```




```
void fxn_ordering_try(int I, int *O){
   int tmp1 = 0;
   int tmp2 = 0;

   foo<1>(I,tmp2,&tmp1);
   foo<2>(tmp1,tmp1,&tmp2);
   *O = tmp2;
}
```



- **See paper for how to get around this using C++ objects**

Tech report and source code available

http://www.ece.cmu.edu/calcm/connect_hls

Carnegie Mellon University

Electrical & Computer ENGINEERING