VLSI design

Netlist generation

P.J. Drongowski
SandSoftwareSound.net

Netlist generation

- irsim netlists are long and complex
- Need to quickly generate simulation model
- Write program to produce model
- Basic ideas
  - C function to generate each circuit block
  - Parameters to generator function
    - Instance name (`char *`)
    - Block input / output names (`char *`)
    - Number of bits (optional, `int`)
    - Number of words (optional, `int`)
  - Internal node name format (when necessary)
    - Instance name
    - Block name
    - Node name
    - Word and/or bit number (optional)
  - Bit numbering: MSB is highest, LSB is zero
  - Standard node names: `Vdd`, `GND`, `PHI1`, `PHI2`
  - Comments to identify instance and block

```
main()
  {
  OpenNetlist("test") ;

  MSlatch("I", 8, "D", "Q", "QBar", "Clock") ;

  CloseNetlist() ;
  }
```

Instance name

Input and output connections

Number of bits

Generator function for inverter

```
void Not(Inst, Bits, A, NotA)
  char *Inst ; int Bits ; char *A, *NotA ;

  /*
   * e Input Output GND
   * p Input Vdd    Output
   */

  {
  register int i ;
  char *In[NODE], *Out[NODE] ;

  Generating(Inst, PlitNot) ;

  VddNode(Vdd) ;
  GroundNode(Ground) ;

  for (i = 0 ; i < Bits ; i++)
    {
    if ((Bits == 1) && (i == 0))
      {
      Input(A, In) ;
      Output(NotA, Out) ;
      }
    else {
      InputB(A, i, In) ;
      OutputB(NotA, i, Out) ;
      }
    Nxistor(In, Out, Ground, 2, 2) ;
    Pxistor(In, Vdd, Out, 2, 4) ;
    }
  }
```

Instance/block comment

Define Vdd and ground nodes

Loop over number of bits

No bit number on single inverter

Generate n- and -pchannel transistors

Power, input, output functions

```
void VddNode(Node) char *Node ;

  {
  }

void GroundNode(Node) char *Node ;

  {
  }

void Input(Name, Node) char *Name, *Node ;

  {
  strcpy(Node, Name) ;
  }

void InputB(Name, Bit, Node)
  char *Name, *Node ; int Bit ;

  {
  sprintf(Node, "%s%d", Name, Bit) ;
  }

void Output(Name, Node) char *Name, *Node ;

  {
  strcpy(Node, Name) ;
  }

void OutputB(Name, Bit, Node)
  char *Name, *Node ; int Bit ;

  {
  sprintf(Node, "%s%d", Name, Bit) ;
  }
```

Internal node and transistor functions

```
void Node(Inst, Comp, Node)
  char *Inst, *Comp, *Node ;

  {
  sprintf(Node, "%s_%s", Inst, Comp) ;
  }

void NodeB(Inst, Comp, Bit, Node)
char *Inst, *Comp, *Node ; int Bit ;

  {
  sprintf(Node, "%s_%s%d", Inst, Comp, Bit) ;
  }

void Nxistor(Gate, Source, Drain, L, W)
 char *Gate, *Source, *Drain ; int L, W;

  {
  fprintf(NetList, "n %s %s %s %d %d\n",
    Gate, Source, Drain, L, W) ;
  }

void Pxistor(Gate, Source, Drain, L, W)
 char *Gate, *Source, *Drain ; int L, W;

  {
  fprintf(NetList, "p %s %s %s %d %d\n",
    Gate, Source, Drain, L, W) ;
  }
```

I/O and comments

```
void Comment(Text) char *Text ;

  {
  fprintf(NetList, "| %s\n", Text) ;
  }

void Generating(Instance, Component)
  char *Instance, *Component ;

  {
  fprintf(NetList,
    "\n| Instance `%s' component `%s'\n\n",
    Instance, Component) ;
  }

void OpenNetlist(Pathname) char *Pathname ;

  {
  if ((NetList=fopen(Pathname,"w"))==NULL)
    {
    fprintf(stderr,"Couldn't create `%s'\n",
      Pathname) ;
    exit() ;
    }
  fprintf(NetList,
    "| Auto-generated netlist\n\n") ;
  }

void CloseNetlist()

  {
  fprintf(NetList, "\n| End of netlist\n") ;
  fclose(NetList) ;
  }
```