CprE / ComS 583 Reconfigurable Computing

Prof. Joseph Zambreno
Department of Electrical and Computer Engineering
Iowa State University

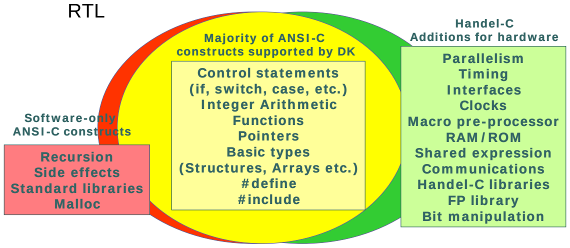
Lecture #26 – Course Wrapup

Quick Points

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
25	26	27 Lect-25	28	29 Lect-26	30	1
2 Dead Week	3	4 Project Seminars (EDE)[1]	5	6 Project Seminars (Others)	7	8
9 Finals Week	10	11	12	13	14	15 Project Write-ups Deadline
16	17	18 Electronic Grades Due	December / November 2007			

Celoxica Handel-C

- Handel-C adds constructs to ANSI-C to enable hardware implementation
  - Synthesizable HW programming language based on C
  - Implements C algorithm direct to optimized FPGA or RTL



Fundamentals

- Language extensions for hardware implementation as part of a system level design methodology
  - Software libraries needed for verification
- Extensions enable optimization of timing and area performance
- Systems described in ANSI-C can be implemented in software and hardware using language extensions defined in Handel-C to describe hardware
- Extensions focused towards areas of parallelism and communication

Variables

- Handel-C has one basic type - integer
- May be **signed** or **unsigned**
- Can be any width, not limited to 8, 16, 32 etc.

Variables are mapped to **hardware registers**

```
void main(void)
{
        unsigned 6 a;
        a=45;
}
```

a = | 1 | 0 | 1 | 1 | 0 | 1 | = 0x2d

MSB ... LSB

Timing Model

- Assignments and delay statements take 1 clock cycle
- Combinatorial Expressions computed between clock edges
  - Most complex expression determines clock period
  - Example: takes 1+n cycles (n is number of iterations)

```
index = 0;                    // 1 Cycle
while (index < length){
      if(table[index] = key)
            found = index;    // 1 Cycle
      else
            index = index+1;  // 1 Cycle
      }
}
```

Parallelism

- Handel-C blocks are by default sequential
- **par{…}** executes statements in parallel
- Par block completes when all statements complete
  - Time for block is time for longest statement
  - Can nest sequential blocks in par blocks
- Parallel version takes 1 clock cycle
  - Allows trade-off between hardware size and performance

Parallel Block

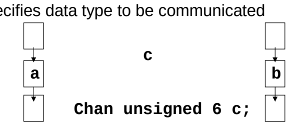
```
// 1 Clock Cycle
   par{
      a=1;
      b=2;
      c=3;
   }
```

Parallel code

```
par(i=0;i<10;i++)
{
   array[i]=0;
}
```


Channels

- Allow communication and synchronization between two parallel branches
  - Semantics based on CSP (used by NASA and US Naval Research Laboratory)
  - Unbuffered (synchronous) send and receive
- Declaration
  - Specifies data type to be communicated




```
Chan unsigned 6 c;
```

```
{
   …
   c!a+1;  //write a+1 to c
   …
}
```

```
{
   …
   c?b;  //read c to b
   …
}
```


Signals

- A signal behaves like a wire - takes the value assigned to it but only for that clock cycle
  - The value can be read back during the same clock cycle
  - The signal can also be given a default value

```
// Breaking up complex expressions
int 15 a, b;
signal <int> sig1;
static signal <int> sig2=0;
a = 7;
par
{
   sig1 = (a+34)*17;
   sig2 = (a<<2)+2;
   b = sig1 + sig2;
}
```


Sharing Hardware for Expressions

- Functions provide a means of sharing hardware for expressions
- By default, compiler generates separate hardware for each expression
  - Hardware is idle when control flow is elsewhere in the program
  - Hardware function body is shared among call sites

```
{…
   x= x*a + b;
   y= y*c + d;
}
```

```
int mult_add(int z,c1,c2){
   return z*c1 + c2; }
{
   …
   x= mult_add(x,a,b);
   y= mult_add(y,c,d);
}
```


Bit-width Analysis

- Higher Language Abstraction
  - Reconfigurable fabrics benefit from specialization
  - One opportunity is bitwidth optimization
- During C to FPGA conversion consider operand widths
  - Requires checking data dependencies
  - Must take worst case into account
  - Opportunity for significant gains for Booleans and loop indices
- Focus here is on specialization

Arithmetic Analysis

- Example

```
int      a;
unsigned b;
a = random();
b = random();
```

a: 32 bits b: 32 bits

```
a = a / 2;
```

a: 31 bits b: 32 bits

```
b = b >> 4;
```

a: 31 bits b: 28 bits

```
a = random() & 0xff;
```

a: 8 bits b: 28 bits

Loop Induction Variable Bounding

- Applicable to *for* loop induction variables.
- Example

```
int  i;                              i: 32 bits

for (i = 0; i < 6; i++) {            i: 3 bits
      …
}
                                     i: 3 bits
```


Clamping Optimization

- Multimedia codes often simulate saturating instructions
- Example

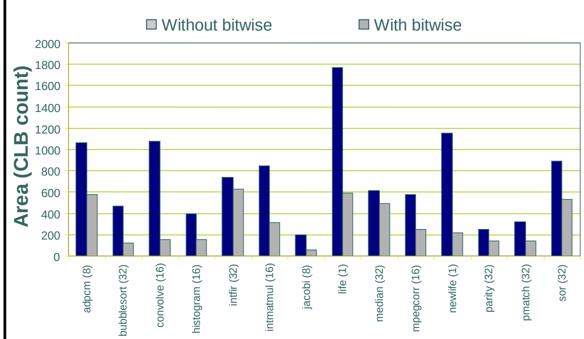
```
int valpred                          valpred: 32 bits

if (valpred > 32767)
  valpred = 32767
else if (valpred < -32768)
  valpred = -32768                   valpred: 16 bits
```


Solving the Linear Sequence

```
a = 0                      <0,0>
for i = 1 to 10
  a = a + 1                <1,460>
  for j = 1 to 10
     a = a + 2             <3,480>
     for k = 1 to 10
       a = a + 3           <24,510>
...= a + 4                 <510,510>
```

- Sum all the contributions together, and take the data-range union with the initial value
- Can easily find conservative range of <0,510>

FPGA Area Savings



Summary

- High-level compilation is still not well understood for reconfigurable computing
- Difficult issue is the parallel specification and verification
- Designers efficiency in RTL specification is quite high. Do we really need better high-level compilation?

Some Emerging Technologies

- Several emerging technologies may make an impact
  - Carbon nanotubes
  - Magnetoelectronic devices
- Technologies are in their infancy

Carbon Nanotubes

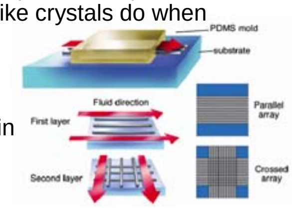
- Extensions of carbon molecules
- Grown as long straight tubes
- "Flow" used to align nanotubes in a specific direction
- Technology still in infancy

SWNT (Single Wall Carbon Nanotubes)

- Nanometer(s) in diameter
- microns long
- good conductors

Bottom-Up Self-Assembly

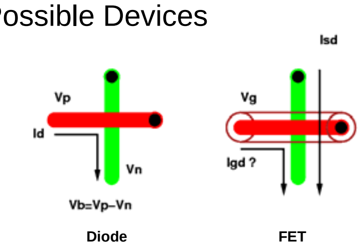
- We can't make nano-circuits *top-down*
  - Lithography can't get to the nano scale
- Make them *bottom-up* with chemical self-assembly
  - Their own physical properties keep them in regular order, much like crystals do when they grow
- Fluid flow self-assembly
  - Crossbar generated in two passes



Nanotubes in Electronics?

- Carbon nanotubes come in two flavors:
  - Metallic
  - Semiconducting
- Metallic nanotubes make great wires
- Semiconducting nanotubes can be made into transistors
- Depending on how nanotubes are formed, range from about 1/3 semiconducting, 2/3 metallic to 2/3 semiconducting, 1/3 metallic
- No good technology at present time for creating nanotubes of just one type

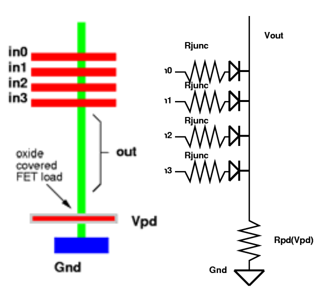
Possible Devices



- Diode connection formed by making connection between upper and lower nanotube
- Nanotubes do not touch when forming a FET
  - Top nanotube covered with oxide
  - Effectively acts as a "gate" to current path

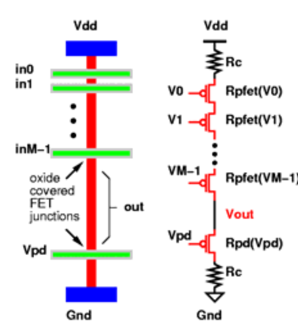
Diode Logic

- Arise directly from touching NW/NTs
- Passive logic
- Non-restoring

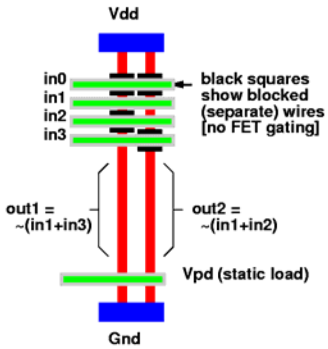


PMOS-like Restoring FET Logic

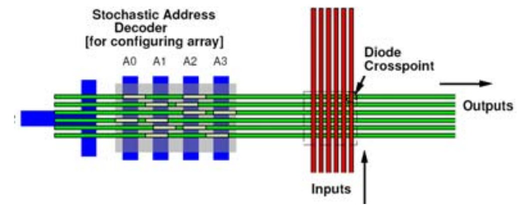
- Use FET connections to build restoring gates
- Static load
  - Like NMOS (PMOS)



Programmed FET Arrays

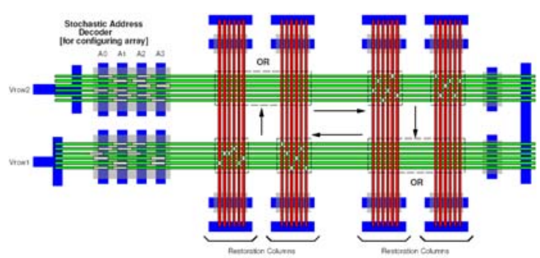


Programmable OR-plane



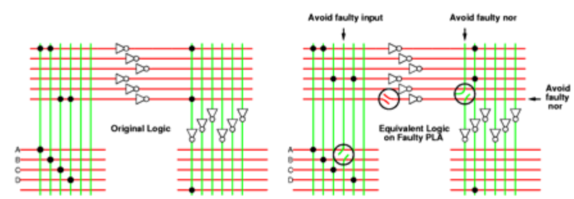
- Addressing is a challenge since order of addresses can't be predetermined
  - Nanotubes can be doped to form different addresses
  - Some redundancy OK
- Diode logic formed at crosspoint

Simple Nanowire-Based PLA



NOR-NOR = AND-OR PLA Logic

Defect Tolerance



All components (PLA, routing) interchangeable;
Allows local programming around faults

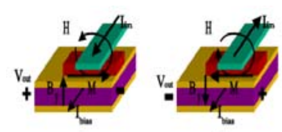
Results [Deh05A]

- Pair of 60-term OR planes roughly same size as 4-LUT
- Special mapping and programming tools needed
- Fault tolerance a big issue

Design	FSM Ins	FSM Outs	FSM S	60-OR-term (s,w) org.	2-in gates	ratio PLA	4-in LUTs	ratio PLA
tbk	6	3	32	(8,1)	213	27	98	12
pma	8	8	24	(18,1)	208	12	83	5
s1	8	6	20	(10,1)	188	19	75	8
ex1	9	19	20	(2,1)	186	93	70	35
keyb	7	2	19	(10,1)	184	18	77	8
s420	19	2	18	1/2 (2,1)	67	67	25	25
s208	11	2	18	1/2 (2,1)	67	67	25	25
sse	7	7	16	(2,1)	111	56	40	20
kirkman	12	6	16	(6,1)	139	23	58	10
bbsse	7	7	16	(2,1)	111	56	40	20
cse	7	7	16	(10,1)	178	18	71	7
dk512	1	3	15	1/2 (2,1)	51	51	16	16
mark1	5	16	15	(4,1)	84	21	38	10
ex4	6	9	14	1/2 (2,1)	60	60	20	20
s386	7	7	13	(2,1)	109	54	39	20
bbara	4	2	10	1/2 (2,1)	46	46	17	17
opus	5	6	10	1/2 (2,1)	79	79	25	25
dk17	2	3	8	1/2 (2,1)	56	56	15	15
shiftreg	1	1	8	1/2 (2,1)	11	11	4	4
ex6	5	8	8	(2,1)	83	42	32	16
dk14	3	5	7	(2,1)	72	36	24	12
beecount	3	4	7	1/2 (2,1)	22	22	9	9
dk27	1	2	7	1/2 (2,1)	20	20	5	5
s27	4	1	6	1/2 (2,1)	23	23	8	8
bbtas	2	2	6	1/2 (2,1)	21	21	6	6
mc	3	5	4	1/2 (2,1)	26	26	7	7
lion	2	1	4	1/2 (2,1)	13	13	3	3
dk15	3	5	4	1/2 (2,1)	61	61	19	19
tav	4	4	4	1/2 (2,1)	27	27	9	9
geometric mean						33		11

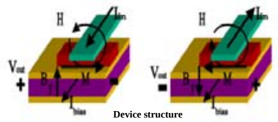
Magnetoelectronic Devices

- Program a cell by setting a directional magnetic field
  - Programming current sets field
- Technique already heavily using in storage devices
- Flexible, reliable
- Advantages:
  - Non-volatile
  - Low power consumption

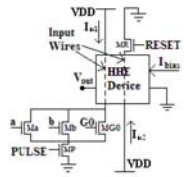


HHE Devices

- Information written as magnetization states by passing a write current through a current line
- HIGH, and LOW output Hall voltage according to direction of magnetization
- Good remanence in the ferromagnet may lead to hysteresis loop and hence memory
- Easily integrated with rest of the CMOS circuit



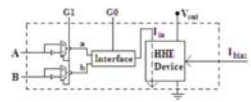
Device structure

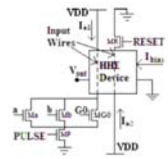


HHE integrated with CMOS logic

Magnetoelectronic Gates

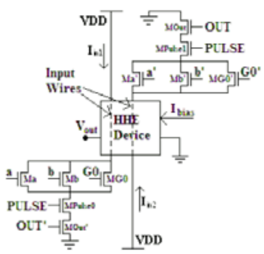
- Use storage cell along with a minimum of external transistors to create logic
- External circuitry induces current which can program cell





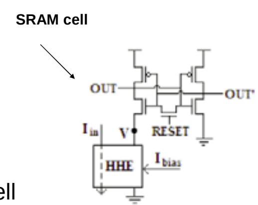
Power Reducing

- Logic only evaluated if the output result will change state
- If change redetected then perform reset
- Otherwise, maintain old value



Magnetoelectronic Look-up Tables

- SRAM storage cell used for high performance
- Initial value of SRAM cell stored in magnetoelectronic cell
- Cell is programmed following reset



Summary

- Difficult to explore without experts in physics and chemistry
- Initial architectural ideas based on perceptions of likely available technology
- Daunting challenges involving CAD and power reduction remain
- Not likely to have much commercial application for 10-15 years
- Active area of research