1 April, 2004
UNCA CSCI 311, NCSU ECE 212

Control paths

Squaring a number

Input n-bit number representing k
Output $2n$-bit number representing k^2

A purely combinational solution?

Start with some C++
Generate some espresso input

```
.i 8
.o 16
.p 256
00000000 0000000000000000
00000001 0000000000000001
……
11111110 1111110000000100
11111111 1111111000000001
.e
```

and some espresso output

```
.i 8
.o 16
.p 187
00010011 0000000101100000
01001111 0000100000000000
……
----01-1 0000000000010000
111----- 1100000000000000
------10 0000000000000100
-------1 0000000000000001
.e
```

compute the gate delays and number of transistors

for 12-bit input

217873 bytes of `espresso` output

3026 AND-gates -- average ~10 inputs
23 OR-gates -- from 1 to 726 inputs
gate delays – $\log_3(10) + \log_3(726) \sim 7$

Another combinational approach

The "grade school" algorithm in an array of gates
Again, compute delays and transistors

~ n * 5 delays
~ n^2 * 25 transistors

Make the last approach sequential

- *n* cycles of addition, shift, and store
- MSI components needed (maximal?)
  - two *n*-bit registers
  - one 2*n*-bit register
  - one 2*n*-bit shifter
  - one 2*n*-bit adder

But you really only add *n* bits

- Improved sequential approach

The interface

n data bits	input
2*n* data bits	output
Clock control	input
Start control	input
Done control	output

The "internal" variables

Save register	holds original value
Work register	holds evolving result

The Algorithm

- Loop until Start asserted
  - Do nothing
- Deassert Done
- Load internal registers (both work and save register)
- For *n* clock cycles
  - Test low bit
  - Add into upper half, if appropriate
  - Shift work register
- Assert Done

In Perl

```
my $r = $n ;
my $h = $n<<15 ;
for (my $i=16; $i; --$i) {
    $r = ($r>>1) + ( ($r & 1) ? $h : 0 ) ;
}
```

The register

Assume data inputs, CLK, and Load control

Remember: Data is loaded at the beginning of the clock cycle

Actions during a clock cycle

Load registers if LD asserted

If Start is asserted

Set Done flip-flop input to 0
Route external input to R and H registers
Assert LD controls of R and H registers
Assert CLR control of C (counter)
Set Working flip-flop input to 1

If Working is asserted and C has not asserted OVR

Set Done flip-flop input to 0
Addin = 0, if R is even; H, if is odd
Route R>>1 + H to R
Assert LD control of R
Assert DCR control of C
Set Working flip-flop input to 1

If Working is not asserted or C has asserted OVR

Set Done flip-flop input to 1
Set Working flip-flop input to 0
Assert CLR control of C

Relevant inputs and outputs

Register	Inputs	Outputs
C	CLR, DCR	OVR
H	LD, data	data
R	LD, data	data

Note the CLR and DCR are always complements