ESE534: Computer Organization

Day 23: April 19, 2010
Control

Previously

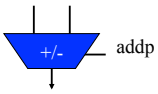
- Looked broadly at instruction effects
- Explored structural components of computation
  - Interconnect, compute, retiming
- Explored operator sharing/time-multiplexing
- Explored branching for code compactness

Today

- Control
  - data-dependent operations
- Different forms
  - local
  - instruction selection
- Control granularity as another parameter in our architecture space

Control

- **Control**: That point where the data affects the instruction stream (operation selection)
  - Typical manifestation
    - data dependent branching
      - if (a!=0) OpA else OpB
      - bne
    - data dependent state transitions
      - new => goto S0
      - else => stay
    - data dependent operation selection

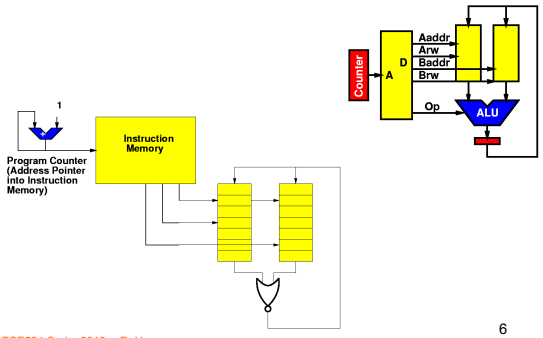


Control

- **Viewpoint**: can have instruction stream sequence without control
  - *I.e.* static/data-independent progression through sequence of instructions is control free
    - C0→C1→C2→C0→C1→C2→C0→…
  - Similarly, FSM w/ no data inputs
  - *E.g.* HW3…non-branching multiplier

Day 6

Programmable Architecture



Terminology (reminder)

- **Primitive Instruction (*pinst*)**
  - Collection of bits which tell a bit-processing element what to do
  - Includes:
    - select compute operation
    - input sources in space (interconnect)
    - input sources in time (retiming)
- **Configuration Context**
  - Collection of all bits (*pinsts*) which describe machine's behavior on one cycle

Back to "Any" Computation

- Design must handle all potential inputs (computing scenarios)
- Requires sufficient generality
- However, computation for **any** given input may be *much* smaller than general case.
- ***Instantaneous*** compute << potential compute

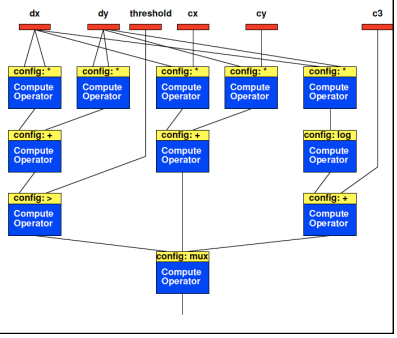
Preclass 1

```
if ((dx*dx+dy*dy)>threshold)
  z=cx*dx+cy*dy
else
  z=log(dx*dy)+c3
```

- How many operations performed?
- Cycles?
- Compute Blocks needed?

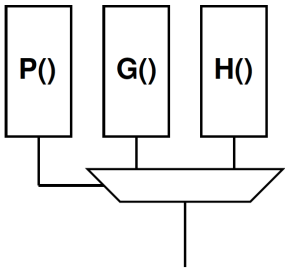
Preclass 1

- Delay?
- Operations?
- How do?



If-Conversion

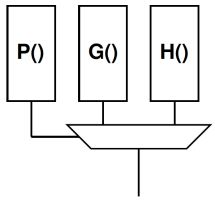
```
if (P())
   G()
else
   H()
```



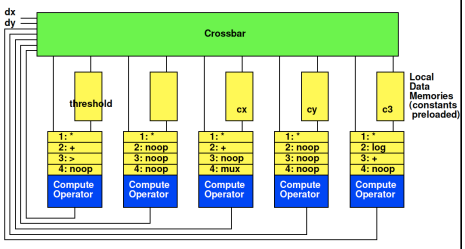
If-Conversion

- Trade-off:
  - Latency
  - Work

```
if (P())
   G()
else
   H()
```



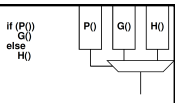
Preclass



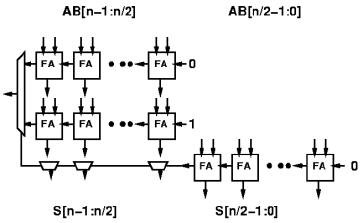
- Operations
- Cycles

Day 3

Idea

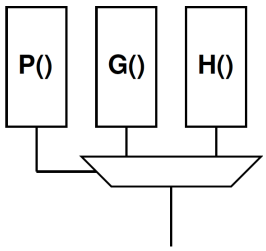


- Compute both possible values and select correct result when we know the answer



If-Conversion ~= Predicated Operations

```
if (P())
   G()
else
   H()
```



P1=P()
P2=!P1
P1 G()
P2 H()

P1=t1>t2
P2=!P1
…
P1 z=t4
P2 z=t5

Why important?

Instruction Control Latency

- For time-multiplexed (data-independent) sequencing
  - Can pipeline instruction distribution
  - Instruction memory read
- Now decision→PC→distribution→read latency becomes part of critical path

Screwdriver Analogy

- Need capability to handle
  - Slothead
  - Phillips
  - Torq
  - Hex…
- But only need one at a time…

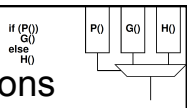
Video Decoder

- *E.g.* Video decoder [frame rate = 33ms]
  - if (packet==FRAME)
    - if (type==I-FRAME)
      - I-FRAME computation
    - else if (type==B-FRAME)
      - B-FRAME computation

Packet Processing

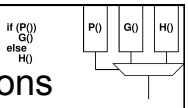
- If IP-V6 packet
  - ….
- If IP-V4 packet
  - …
- If VoiP packet
  - …
- If modem packet
  - ….

Two Control Options



1. Local control
   - unify choices
     - build all options into spatial compute structure and select operation → Mux-conversion
2. Instruction selection
   - provide a different instruction (instruction sequence) for each option
   - selection occurs when chose which instruction(s) to issue

Two Control Options



1. Local control
2. Instruction selection

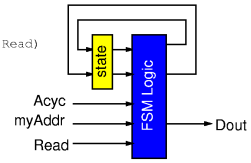
May use both within an application

- Local control in critical path, inner-loops, where latency rather than parallelism limited
- Instruction-selection coarse-grain selection, where have plenty of task parallelism so latency not limit computation

FSM Example (local control)

FSM Description

```
Idle  (00):
      if (Acyc & myAddr & Read)
          goto Wait1
      else
          goto Idle
Wait1 (01):
      goto Data
Data  (10):
      Assert Dout
      goto Wait2
Wait2 (11):
      goto Idle
```



FSM Logic

```
Dout = S1*/S0
NS0 =  /S1*/S0*Acyc*myAddr*Read + S1*/S0
NS1 = /S1*S0 + S1*/S0
```

4 4-LUTs
2 LUT Delays

FSM Example

Context 0 (S1=0)

```
Dout = 0
NS0 = /S0*Acyc*myAddr*Read
NS1 = S0
```

3 4-LUTs
1 LUT Delay

Context 1 (S1=1)

```
Dout = /S0
NS0 = /S0
NS1 = /S0
```


Local Control

- LUTs used ≠ LUT evaluations produced
- → Counting LUTs not tell cycle-by-cycle LUT needs

FSM Example (Instruction)

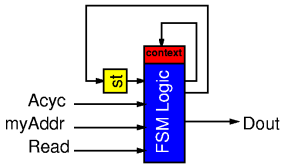
Context 0 (S1=0)

```
Dout = 0
NS0 = /S0*Acyc*myAddr*Read
NS1 = S0
```

3 4-LUTs
1 LUT Delay

Context 1 (S1=1)

```
Dout = /S0
NS0 = /S0
NS1 = /S0
```



Local vs. Instruction

- If can decide early enough
  - and afford schedule/reload
  - instruction select → less computation
- If load too expensive
  - local instruction
    - faster
    - maybe even less capacity (AT)

Slow Context Switch

- When context selection is slow, Instruction selection profitable only at coarse grain
  - Xilinx ms reconfiguration times
- *E.g.* Video decoder [frame rate = 33ms]
  - if (packet==FRAME)
    - if (type==I-FRAME)
      - IF-context
    - else if (type==B-FRAME)
      - BF-context

Local vs. Instruction

- For multicontext device
  - *i.e.* fast (single) cycle switch
  - factor according to available contexts
- For conventional FPGAs
  - factor only for gross differences
  - and early binding time

FSM Control Factoring Experiment

FSM Example

- FSM -- canonical "control" structure
  - captures many of these properties
  - can implement with deep multicontext
    - instruction selection
  - can implement as multilevel logic
    - unify, use local control
- Serve to build intuition

Full Partitioning Experiment

- Give each state its own context
- Optimize logic in state separately
- Tools
  - `mustang`, `espresso`, `sis`, `Chortle`
- Use:
  - one-hot encodings for single context
    - smallest/fastest
  - dense for multicontext
    - assume context select needs dense

Comparison

- Assume stay in context for a number of LUT delays to evaluate logic/next state
- Pick delay from worst-case
- Assume single LUT-delay for context selection?
  - savings of 1 LUT-delay => comparable time
- Count LUTs in worst-case state

Full Partition (Area Target)

FSM	States	Single Context			Context per State			Ratio	Delta
		Levels	N_{dLUT}	Area ($M\lambda^2$)	Levels	N_{dLUT}	Area ($M\lambda^2$)	$\frac{A_{multi}}{A_{single}}$	Levels
bbara	10	6	25	22.0	1	6	9.5	0.43	5
bbsse	16	4	50	43.9	3	12	24.6	0.56	1
bbtas	6	3	7	6.1	1	5	6.34	1.0	2
beecount	7	4	14	12.3	1	7	9.4	0.77	3
cse	16	6	83	72.9	2	15	30.7	0.42	4
dk14	7	4	58	50.9	1	8	10.8	0.21	3
dk15	4	12	25	22.0	1	7	7.8	0.35	11
dk16	27	5	80	70.2	1	8	23.2	0.33	4
dk17	8	6	19	16.7	1	6	8.5	0.51	5
dk512	15	2	20	17.6	1	7	13.8	0.79	1
donfile	24	2	46	40.4	1	6	16.0	0.40	1
ex1	20	7	120	105.4	2	26	61.4	0.58	5
ex4	14	7	21	18.4	1	13	24.6	1.33	6
ex6	8	5	57	50.0	1	11	15.7	0.31	4
keyb	19	7	112	98.3	4	14	32.0	0.32	3
mc	4	2	8	7.0	1	7	7.8	1.10	1
modulo12	12	6	12	10.5	1	5	8.7	0.82	5
planet	48	6	150	131.7	1	25	113.6	0.86	5
pma	24	6	82	72.0	2	15	40.1	0.56	4
s1	20	5	137	120.3	5	25	59.0	0.49	0
s1488	48	6	152	133.5	3	27	122.7	0.92	3
s1a	20	5	72	63.2	7	21	49.6	0.78	-2
s208	18	4	38	33.4	1	7	15.4	0.46	3
s27	6	2	5	4.4	1	4	5.1	1.20	1
s386	13	5	42	36.9	2	12	21.8	0.59	3
s420	18	3	40	35.1	1	7	15.4	0.44	2
s510	47	5	54	47.4	1	13	58.1	1.22	4
s8	5	4	12	10.5	1	4	4.7	0.45	3
s820	25	6	92	80.8	3	30	82.5	1.02	3
sand	32	7	178	156.3	5	30	98.9	0.63	2
sse	16	4	50	43.9	3	12	24.6	0.56	1
styr	30	7	186	163.3	4	21	65.9	0.40	3
tbk	32	8	340	298.5	6	33	108.8	0.36	2
Average								0.64	3

Full Partition (Delay Target)

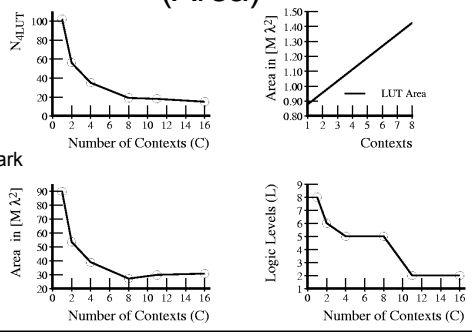
FSM	States	Single Context			Context per State			Ratio	Delta
		Levels	N_{dLUT}	Area ($M\lambda^2$)	Levels	N_{dLUT}	Area ($M\lambda^2$)	$\frac{A_{multi}}{A_{single}}$	Levels
bbara	10	3	40	35.1	1	6	9.5	0.27	2
bbsse	16	3	60	52.7	2	14	28.7	0.54	1
bbtas	6	2	9	7.9	1	5	6.3	0.80	1
beecount	7	2	19	16.7	1	7	9.4	0.57	1
cse	16	4	97	85.2	2	15	30.7	0.36	2
dk14	7	3	67	58.8	1	8	10.8	0.18	2
dk15	4	3	37	32.5	1	7	7.8	0.24	2
dk16	27	3	83	72.9	1	8	23.2	0.32	2
dk17	8	2	26	22.8	1	6	8.5	0.37	1
dk512	15	2	20	17.6	1	7	13.8	0.79	1
donfile	24	2	46	40.4	1	6	16.0	0.40	1
ex1	20	4	151	132.6	2	26	61.4	0.46	2
ex4	14	2	25	22.0	1	13	24.6	1.12	1
ex6	8	3	62	54.4	1	11	15.7	0.29	2
keyb	19	4	150	131.7	3	26	59.3	0.45	1
mc	4	2	8	7.0	1	7	7.8	1.10	1
modulo12	12	1	13	11.4	1	5	8.7	0.76	0
planet	48	4	172	151.0	1	25	113.6	0.75	3
pma	24	4	139	122.0	2	15	40.1	0.33	2
s1	20	4	195	171.2	3	30	70.8	0.41	1
s1488	48	4	183	160.7	2	28	127.2	0.79	2
s1a	20	3	107	93.9	4	30	70.8	0.75	-1
s208	18	3	40	35.1	1	7	15.4	0.44	2
s27	6	2	5	4.4	1	4	5.0	1.16	1
s386	13	4	54	47.4	2	12	21.8	0.46	2
s420	18	3	40	35.1	1	7	15.4	0.44	2
s510	47	3	76	66.7	1	13	58.1	0.87	2
s8	5	2	13	11.4	1	4	4.8	0.42	1
s820	25	3	137	120.3	3	30	82.5	0.69	0
sand	32	4	224	196.7	3	43	141.7	0.72	1
sse	16	3	60	52.7	2	14	28.7	0.54	1
styr	30	5	285	250.2	3	23	72.2	0.29	2
tbk	32	5	510	447.8	4	42	138.4	0.31	1
Average								0.56	1.36

Full Partitioning

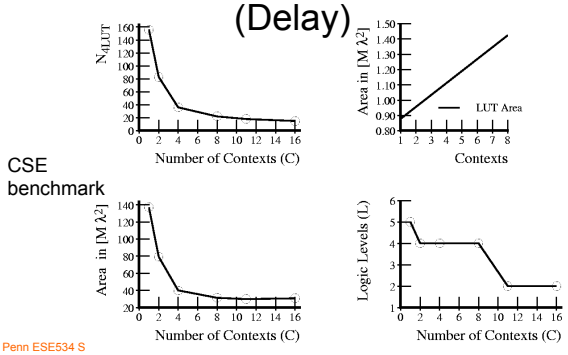
- Full partitioning comes out better
  - ~40% less area
- Note: full partition may not be optimal area case
  - *e.g.* intro example,
    - no reduction in area or time beyond 2-context implementation
    - 4-context (full partition) just more area
      - (additional contexts)

Partitioning versus Contexts (Area)

CSE benchmark



Partitioning versus Contexts (Delay)



CSE benchmark

Partitioning versus Contexts (Heuristic)

- Start with dense `mustang` state encodings
- Greedily pick state bit which produces
  - least greatest area split
  - least greatest delay split
- Repeat until have desired number of contexts

Partition to Fixed Number of Contexts

FSM	States	Best Single Context	Area Ratio by Number of Context (Dense Encodings) 1	2	4	8	16	32	64
Area Target									
average ratio		1.00	1.51	0.86	0.63	0.56	0.70	1.09	1.92
average delta		0.00	-0.27	0.33	1.27	2.18	2.70	3.03	3.06
Delay Target									
average ratio		1.00	1.45	1.05	0.59	0.50	0.62	0.95	1.67
average delta		0.00	-0.91	-0.48	0.06	0.64	0.91	1.15	1.21

N.B. - more realistic, device has fixed number of contexts.

Extend Comparison to Memory

- Fully local => compute with LUTs
- Fully partitioned => lookup logic (context) in memory and compute logic
- How compare to fully memory?
  - Simply lookup result in table?

Memory FSM Compare (small)

FSM	states	ins	outs	**Min area** ($M\lambda^2$)	Integral Addr. & Data Organization	**Memory area** ($M\lambda^2$)	**FPGA area** ($M\lambda^2$)	**8-ctx DPGA area** ($M\lambda^2$)
bbtas	6	2	2	0.1	$2^5\times5$	0.2	6.1	7.1
dk15	4	3	5	0.3	$2^5\times7$	0.3	21.9	10.0
dk17	8	2	3	0.2	$2^5\times6$	0.2	16.7	8.5
dk512	15	1	3	0.3	$2^5\times7$	0.3	17.6	10.0
mc	4	3	5	0.3	$2^5\times7$	0.3	7.0	10.0
modulo12	12	1	1	0.1	$2^5\times5$	0.2	10.5	7.1
beecount	7	3	4	0.5	$2^5\times7$	0.5	12.3	10.0
dk14	7	3	5	0.5	$2^5\times8$	0.6	50.9	11.4
dk16	27	2	3	1.0	$2^7\times8$	1.3	70.2	11.4
donfile	24	2	1	0.7	$2^7\times6$	0.9	40.4	8.5
s27	6	4	1	0.5	$2^7\times4$	0.6	4.4	5.7
s8	5	4	1	0.4	$2^7\times4$	0.6	10.5	5.7
bbara	10	4	2	1.2	$2^8\times6$	1.8	21.9	11.4
ex6	8	5	8	3.4	$2^8\times11$	3.4	50.0	15.7
ex4	14	6	9	14.0	$2^{10}\times13$	16.0	18.4	18.5
bbsse	16	7	7	27.0	$2^{11}\times11$	27.0	43.9	21.4
cse	16	7	7	27.0	$2^{11}\times11$	27.0	72.9	27.1
tbk	32	6	3	19.7	$2^{11}\times8$	19.7	298.5	68.4

Memory FSM Compare (large)

FSM	states	ins	outs	**Min area** ($M\lambda^2$)	Integral Addr. & Data Organization	**Memory area** ($M\lambda^2$)	**FPGA area** ($M\lambda^2$)	**8-ctx DPGA area** ($M\lambda^2$)
sse	16	7	7	27.0	$2^{11}\times11$	27.0	43.9	21.4
s386	13	7	7	22.9	$2^{11}\times11$	27.0	36.9	18.5
keyb	19	7	2	20.4	$2^{12}\times7$	34.4	98.3	31.3
planet	48	7	19	184.3	$2^{13}\times25$	245.8	131.7	54.1
pma	24	8	8	95.8	$2^{13}\times13$	127.8	72.0	34.2
s1	20	8	6	67.6	$2^{13}\times11$	108.1	120.3	62.7
s1a	20	8	6	67.6	$2^{13}\times11$	108.1	63.2	54.1
ex1	20	9	19	294.9	$2^{14}\times24$	471.9	105.4	55.5
s1488	48	8	19	368.6	$2^{14}\times25$	491.5	133.5	74.0
styr	30	9	10	276.5	$2^{14}\times15$	294.9	163.3	57.0
s208	18	11	2	309.7	$2^{16}\times7$	550.5	33.4	12.8
sand	32	11	9	1101.0	$2^{16}\times14$	1101.0	156.3	62.7
s820	25	18	19	188743.7	$2^{23}\times24$	241591.9	80.8	64.1
s420	18	19	2	79272.3	$2^{24}\times7$	140928.6	35.1	14.2
s510	47	19	7	384408.9	$2^{25}\times13$	523449.1	47.4	25.6

Memory FSM Compare (notes)

- Memory selected was "optimally" sized to problem
  - in practice, not get to pick memory allocation/organization for each FSM
  - no interconnect charged
- Memory operate in single cycle
  - but cycle slowing with inputs
- Smaller for <11 state+input bits
- Memory size not affected by CAD quality (FPGA/DPGA is)

Control Granularity

Architectural Parameter(s)

Instruction Selection

Preclass

```
WAIT:  if (in.type==header)
         cnt=in.header_payload_size;
         checksum=0;
         goto RECEIVE;
       else goto WAIT;
RECEIVE: checksum=checksum xor in;
       data[packet][cnt]=in;
       cnt--;
       if (cnt==0) goto CHECK;
       else goto RECEIVE;
CHECK: if (in==checksum)
         packet++;
       goto WAIT
```


Preclass

- How support two ports on VLIW with single controller?

```
WAIT:  if (in.type==header)
         cnt=in.header_payload_size;
         checksum=0;
         goto RECEIVE;
       else goto WAIT;
RECEIVE: checksum=checksum
  xor in;
       data[packet][cnt]=in;
       cnt--;
       if (cnt==0) goto CHECK;
       else goto RECEIVE;
CHECK: if (in==checksum)
         packet++;
       goto WAIT
```

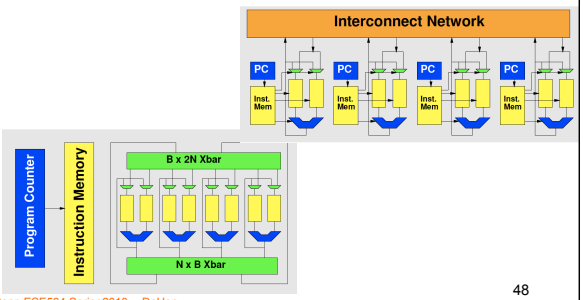

Instruction Control

- If FSMs advance orthogonally
  - (really independent control)
  - context depth => product of states
    - for full partition
  - *I.e.* w/ single controller (PC)
    - must create product FSM
    - which may lead to state explosion
      - N FSMs, with S states => S^N product states

Day 10

Architectural Differences

- What differentiates a VLIW from a multicore?



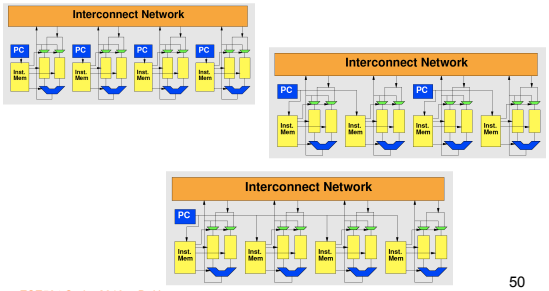
Preclass

- How support on two VLIWs with separate controllers?

```
WAIT:  if (in.type==header)
           cnt=in.header_payload_size;
           checksum=0;
           goto RECEIVE;
       else goto WAIT;
RECEIVE: checksum=checksum
  xor in;
       data[packet][cnt]=in;
       cnt--;
       if (cnt==0) goto CHECK;
       else goto RECEIVE;
CHECK: if (in==checksum)
           packet++;
       goto WAIT
```


Architectural Questions

- How many pinsts/controller?



Day 12

Architecture Taxonomy

PCs	Pints/PC	depth	width	Architecture
0	1	1	1	FPGA
1	1	1024	32	Scalar Processor (RISC)
1	N	D	W	VLIW (superscalar)
1	1	Small	W*N	SIMD, GPU, Vector
N	1	D	W	MIMD
4	4	2048	64	Quad core

Architectural Questions

- How many pinsts/controller?
- Fixed or Configurable assignment of controllers to pinsts?
  - …what level of granularity?

Architectural Questions

- Effects of:
  - Too many controllers?
  - Too few controllers?
  - Fixed controller assignment?
  - Configurable controller assignment?

Architectural Questions

- Too many:
  - wasted space on extra controllers
  - synchronization?
- Too few:
  - product state space and/or underuse logic
- Fixed:
  - underuse logic if when region too big
- Configurable:
  - cost interconnect, slower distribution

Admin

- Final Exercise
  - update with sparing-based design
  - discussion period end 4/26
- Remember online course feedback
- Reading for Wednesday online

Big Ideas [MSB Ideas]

- **Control**: where data effects instructions (operation)
- Two forms:
  - local control
    - all ops resident → fast selection
  - instruction selection
    - may allow us to reduce instantaneous work requirements
    - introduce issues
      - depth, granularity, instruction load and select time

Big Ideas [MSB-1 Ideas]

- If-Conversion
  - Latency vs. work tradeoff
- Intuition → explored canonical FSM case
  - few context can reduce LUT requirements considerably (factor dissimilar logic)
  - similar logic more efficient in local control
  - overall, moderate contexts (*e.g.* 8)
    - exploits both properties … better than extremes