Saving Energy with Architectural and Frequency Adaptations for Multimedia Applications

Chris Hughes

w/ Jayanth Srinivasan and Sarita Adve

Department of Computer Science

University of Illinois at Urbana-Champaign

MICRO 34, December 2001

Motivation (1 of 3)

Multimedia and communication will be critical workloads

- Traditionally used ASICs, DSPs
- Now critical for general-purpose processors

General-purpose processors have high energy consumption

Motivation (2 of 3)

Examine soft real-time multimedia applications

- Each frame must be processed within deadline
  - Soft real-time $\Rightarrow$ can miss a few deadlines
- Leftover processing time = *slack*
  - Presence of slack $\Rightarrow$ can slow processor to save energy

Motivation (3 of 3)

Many proposals for adaptive hardware to save energy

- **D**ynamic **v**oltage and frequency **s**caling (**DVS**)
- Architectural adaptation
  - Instruction window size, issue width, functional units, …

How to control adaptation?

Contributions

Adaptation control algorithm for multimedia [outlined in ISCA ’01]

- DVS + architectural adaptation
- Reduces slack to save most energy possible

Interaction between DVS and architectural adaptation

- Which is better and when?
- Is combining them effective?

Results

Adaptation control algorithm effective

- Eliminates most slack with few missed deadlines

DVS + architectural adaptation most energy efficient

- DVS gives majority of gains for our suite
- But other cases possible

Best architecture depends on presence of DVS

- No DVS $\Rightarrow$ simple architectures
- With DVS $\Rightarrow$ complex architectures
  - Significant implications for architecture design

Outline

Adaptation Control Algorithm

Experimental Methodology

Results

Conclusions

Control Algorithm Outline [ISCA '01]

(1) When to adapt?

Execution time variability at frame level

$\Rightarrow$ Adaptation at frame granularity

(2) What to adapt?

Must predict time, energy of next frame for *all configurations*

Pick lowest energy configuration that can meet deadline

Use ISCA '01 findings for prediction

- IPC almost constant
- Little time in memory stalls
- Instruction count changes slowly

Execution Time Prediction for a Frame

$$\text{Execution cycles} = \frac{1}{\text{IPC}} \times \text{Instruction count}$$

IPC constant ⇒

Get by profiling initial frame

Memory time small ⇒

Profile only one frequency

IC changes smoothly ⇒

Can use simple predictor

One prediction for all hardware

⇒ Frame execution time dynamically predictable

Dynamic predictor needed only for frame *instruction count*

Energy prediction analogous

Adaptation Control Algorithm

Profiling Phase

For each hardware, H

$Imax_H$ = Maximum instructions H can execute in deadline

EPI_H = **E**nergy **p**er **I**nstruction

Adaptation Phase

Predict instruction count for next frame

Choose hardware with Imax ≥ prediction and least EPI

Adaptation Control Algorithm

Profiling Phase

For each hardware, H, with architecture A

$Imax_H$ = Maximum instructions H can execute in deadline

= Deadline $\times$ $Frequency_H$ $\times$ IPC_A

EPI_H = Energy per Instruction

Adaptation Phase

Predict instruction count for next frame

Choose hardware with Imax $\geq$ prediction and least EPI

Adaptation Control Algorithm

Profiling Phase

For each architecture, A, measure IPC_A at one voltage/freq

For each hardware, H, with architecture A

$Imax_H$ = Maximum instructions H can execute in deadline

= Deadline $\times$ $Frequency_H$ $\times$ IPC_A

EPI_H = Energy per Instruction

Adaptation Phase

Predict instruction count for next frame

Choose hardware with Imax $\geq$ prediction and least EPI

EPI = Energy per Instruction

For each hardware, H, with architecture A

$$EPI_H = \frac{Energy_H}{\#\,Instructions}$$

$$= \frac{Time_H \times Power_H}{Time_H \times f_H \times IPC_A} \qquad C_A\, V_H^{\,2}\, f_H$$

$C_A \propto P_A$ = power at some base voltage, frequency

$$EPI_H \propto \frac{P_A \times V_H^{\,2}}{IPC_A}$$

Adaptation Control Algorithm

Profiling Phase

For each arch, A, measure IPC_A and P_A at one voltage/freq

For each hardware, H, with architecture A

$Imax_H = Deadline \times Frequency_H \times IPC_A$

EPI_H = Energy per Instruction $\propto P_A V_H^2 / IPC_A$

Adaptation Phase

Predict instruction count for next frame

Choose hardware with Imax ≥ prediction and least EPI

Adaptation Control Algorithm

Profiling Phase

For each arch, A, measure IPC_A and P_A at one voltage/freq

For each hardware, H, with architecture A

$Imax_H = Deadline \times Frequency_H \times IPC_A$

EPI_H = Energy per Instruction $\propto P_A V_H^2 / IPC_A$

Adaptation Phase

Predict instructions: Max of past 5 frames

Choose hardware with Imax $\geq$ prediction and least EPI

Choosing Correct Hardware

Choose hardware with Imax ≥ prediction and least EPI

In profile phase

Build EPI-Imax table - order hardware in increasing EPI

Hardware (increasing EPI)	Imax
⋮	⋮

In adaptation phase

Choose first entry in table with Imax ≥ predicted instructions

Adaptation Control Algorithm

Profiling Phase

For each arch, A, measure IPC_A and P_A at one voltage/freq

For each hardware, H, with architecture A

$Imax_H = Deadline \times Frequency_H \times IPC_A$

EPI_H = Energy per Instruction $\propto P_A V_H^2 / IPC_A$

Order hardware by increasing EPI in EPI-Imax table

Adaptation Phase

Predict instruction count: Max of past 5 frames

Choose hardware with Imax $\geq$ prediction and least EPI

= First hardware in EPI-Imax table with Imax $\geq$ prediction

Modifications for Continuous DVS

At least one system with continuous DVS

$\Rightarrow$ EPI-Imax table too big

Modified version of algorithm for continuous DVS systems

See paper

Outline

Adaptation Control Algorithm

Experimental Methodology

Results

Conclusions

Workload

Speech codecs

- GSMenc, GSMdec
- G728enc, G728dec

Video codecs

- H263enc, H263dec
- MPGenc, MPGdec

Audio (Music) codecs

- MP3dec

Base Architecture Studied

1GHz out-of-order processor

- 8-issue, 128 entry instruction window
- 64KB L1 data (2 cycles)
- 1MB L2 data (20 cycles)
- 102 cycles main memory

Aggressive clock gating

Experimental Methodology (1 of 2)

Two sets of deadlines

- Maximum and tighter

RSIM + Wattch for time and energy simulations

Experimental Methodology (2 of 2)

Processors evaluated

NoAdapt, Arch, CDVS, DDVS, CDVS+Arch, DDVS+Arch

Architectural adaptations

Issue width and instruction window size, functional units

Architecture	Mean IPC	Mean Power
Base	2.64	12.3W
.5x IW	2.17	9.1W
.5x IW, .5x FU	1.86	7.3W
.25x IW, .5x FU	1.45	5.6W

DVS adaptations: Frequency from 100 MHz to 1GHz

Continuous (CDVS) or discrete (DDVS) with 100 MHz steps

Outline

Adaptation Control Algorithm

Experimental Methodology

Results

Conclusions

How Good is the Algorithm?

Missed deadlines

For all deadlines and processors, very few deadlines missed

Average across all apps = 2.2%

Maximum for a single app = 4.3%

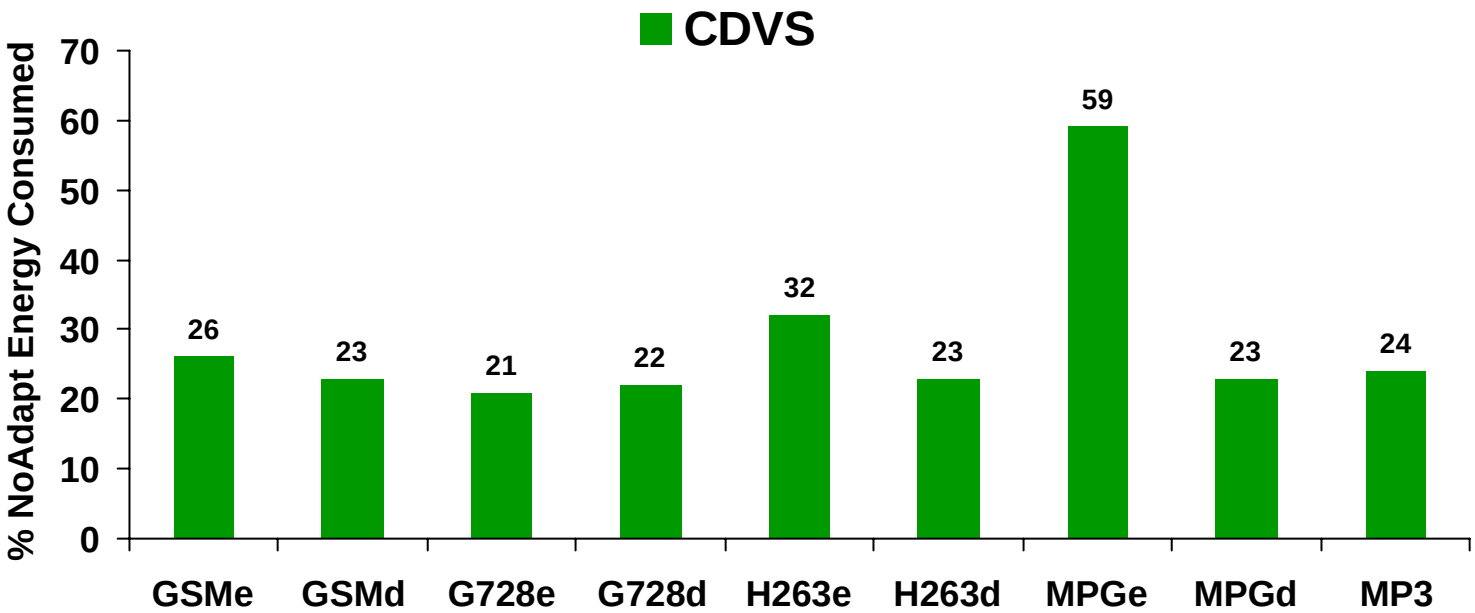
Slack removed

Slack = Idle time between end of processing until deadline

Most slack removed

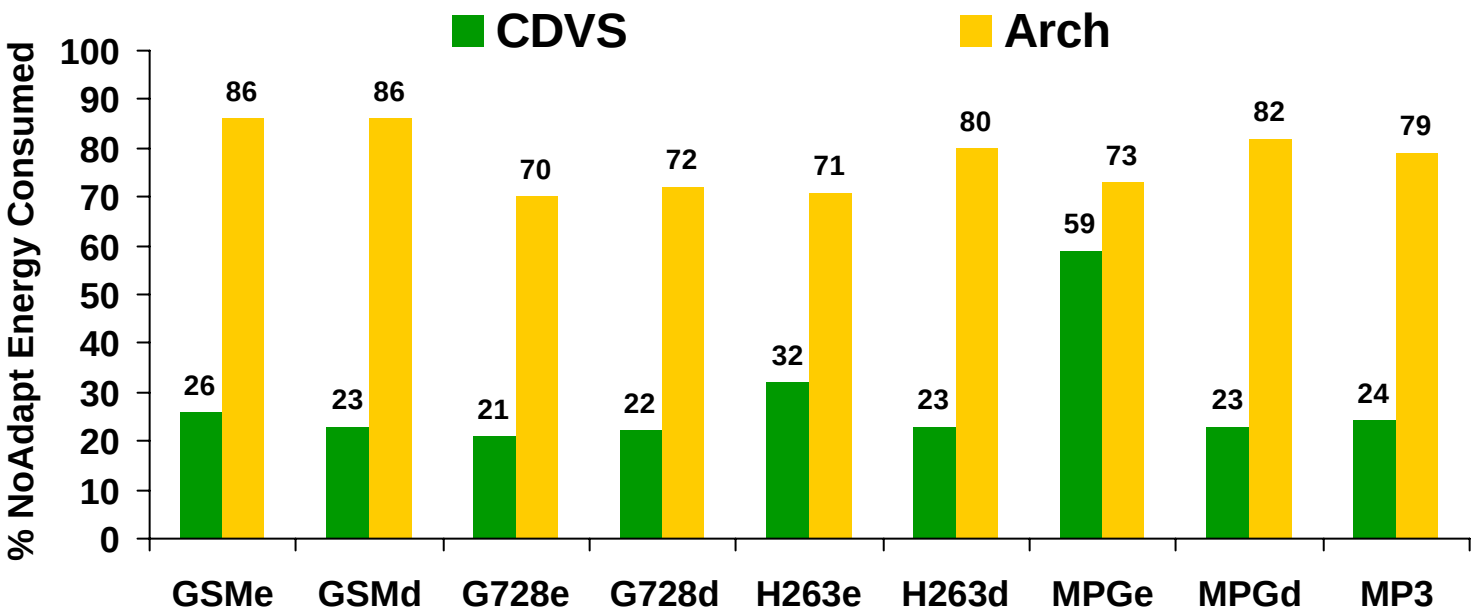
Remaining slack mostly from system limitations

Energy Savings



DVS very effective - average savings 68% to 78%

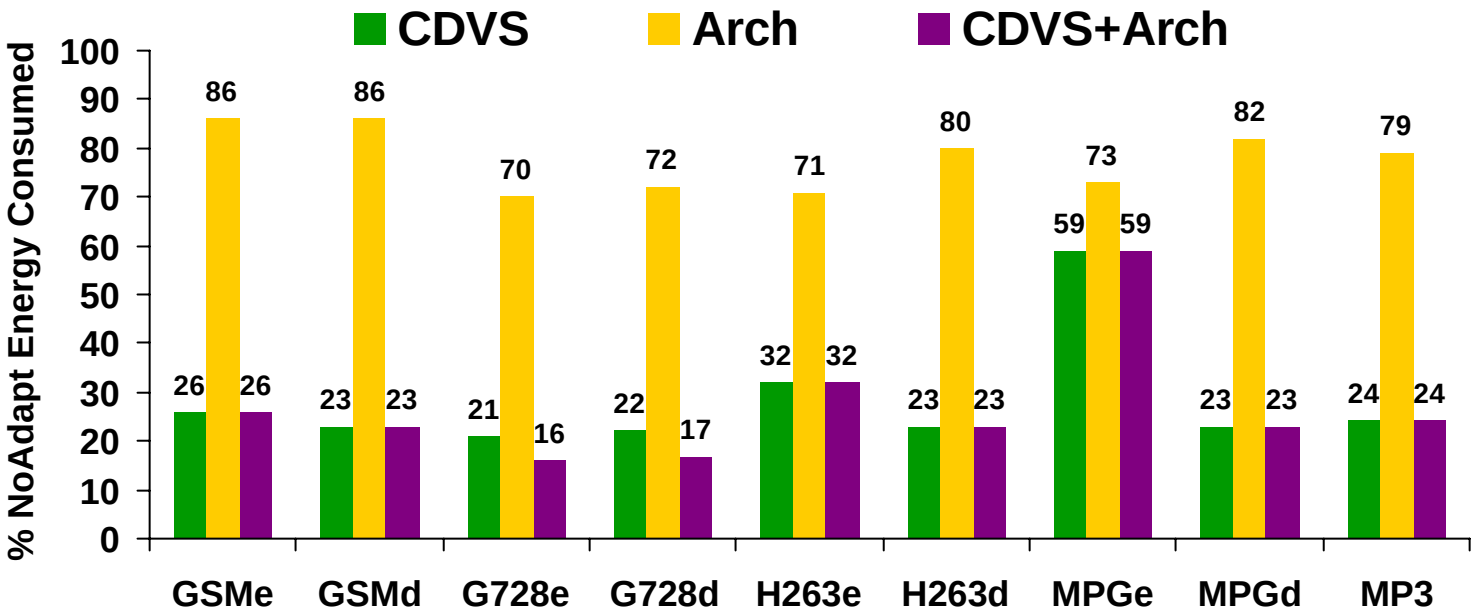
Energy Savings



DVS very effective - average savings 68% to 78%

Architectural adaptation effective, but much less than DVS

Energy Savings



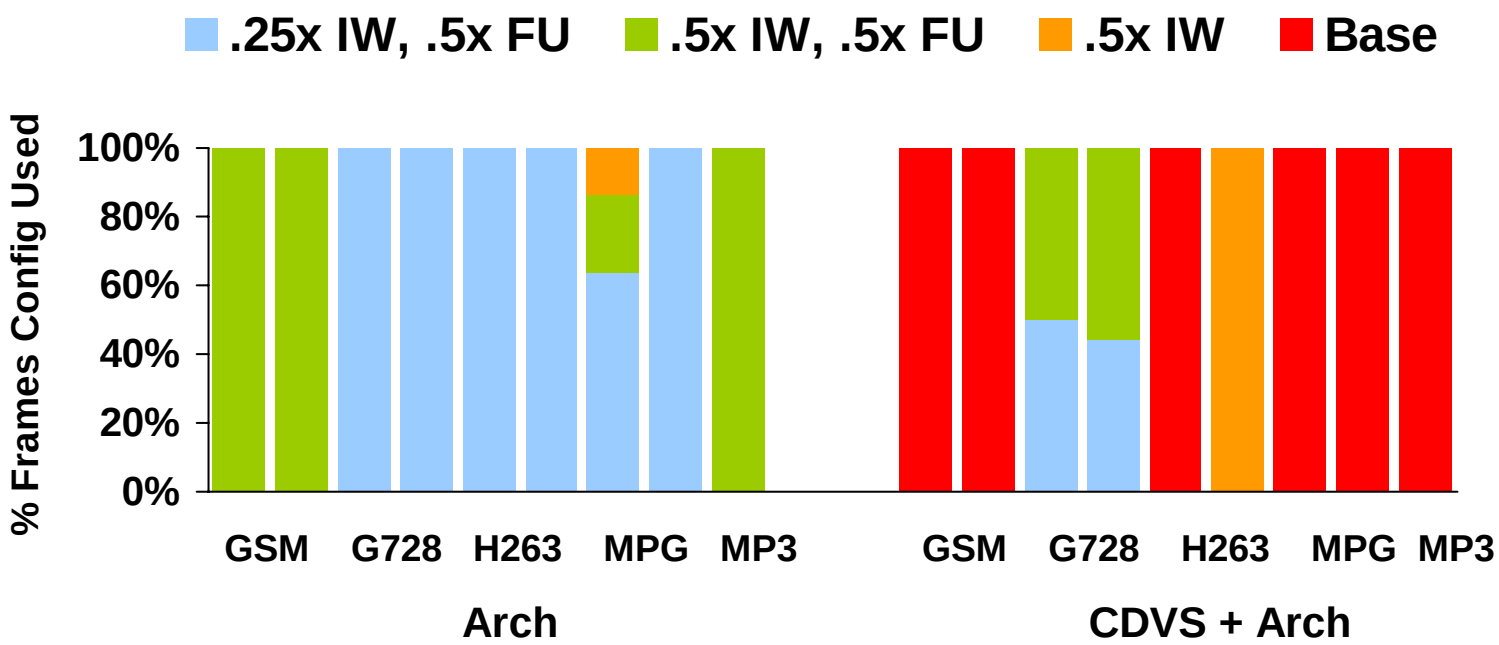
DVS very effective - average savings 68% to 78%

Architectural adaptation effective, but much less than DVS

DVS + Arch best

- Average 5%-17%, max 30%
- Overall, DVS gives majority of gains for our suite

Architectural Configurations Exercised



Most energy efficient architecture depends on presence of DVS

Without DVS, simple configurations (low IPC) chosen

With DVS, more aggressive configurations (high IPC) chosen

High IPC allows running at low frequency

Significant implications for architecture design

When Is Architectural Adaptation Beneficial?

When is it effective to have architectural adaptation with DVS?

Energy efficiency for given computation $\propto \frac{P_A}{IPC_A^3}$

$\Rightarrow$ Lower $\frac{P_A}{IPC_A^3}$

e.g., simpler arch useful if app has little parallelism

Application has slack at lowest frequency

$\Rightarrow$ Simpler arch can exploit this slack

Optimal frequency not supported by D-DVS

May have other opportunities with intra-frame adaptation

Conclusions

Adaptation control algorithm at frame granularity for DVS+Arch

- Effectively reduces slack with few missed deadlines

DVS + architectural adaptation most energy efficient

- DVS gives majority of gains for our suite
- But other cases possible

Best architecture depends on presence of DVS

- No DVS $\Rightarrow$ simple architectures
- With DVS $\Rightarrow$ complex architectures
  - Significant implications for architecture design

Frame Sizes

Application	Frame Size
Speech Codecs	
GSMenc, GSMdec	20ms (160 samples@8KHz)
G728enc, G728dec	625μs (5 samples@8KHz)
Video Codecs	
H263enc, H263dec	176x144, 40ms (25fps)
MPGenc, MPGdec	176x144, 33.3ms (30fps)
Audio (Music) Codecs	
MP3dec	26.1ms (1151 samples@44.1KHz)

Modifications for Continuous DVS

At least one processor has continuous DVS

$\Rightarrow$ table too long

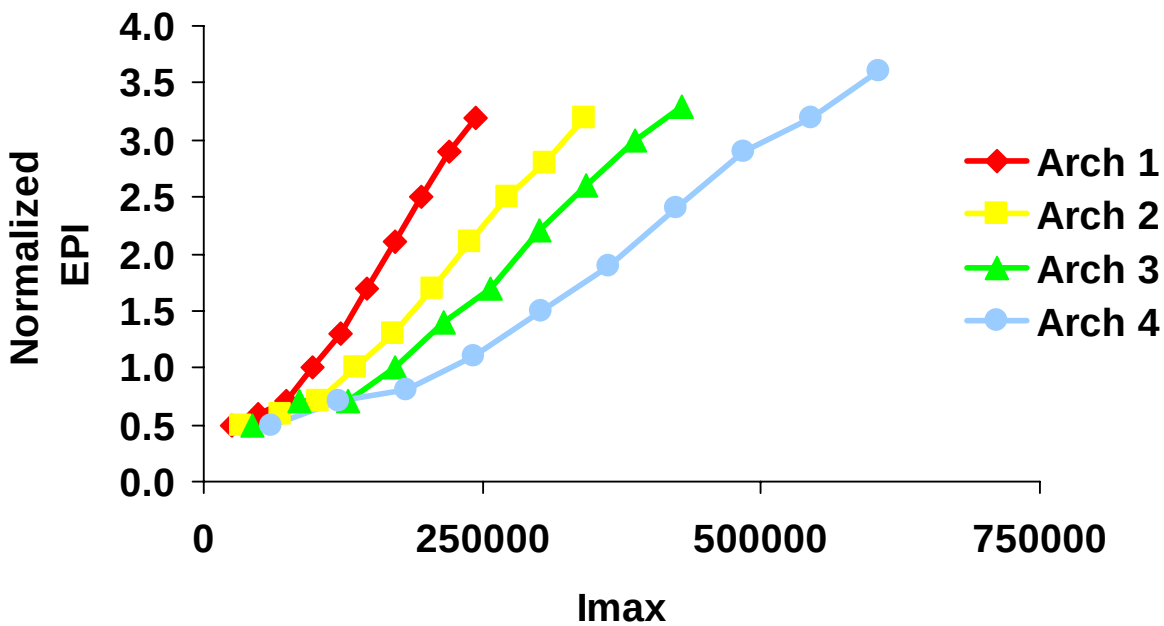
For $V_H \gg V_{threshold}$, $V_H \propto f_H = \dfrac{Imax_H}{Deadline \times IPC_A}$

$$EPI_H \propto \frac{P_A \times V_H^2}{IPC_A} \propto Imax^2 \frac{P_A}{IPC_A^3}$$

$\Rightarrow$ At each Imax, same architecture has least EPI

Continuous DVS

$$EPI_H \propto Imax^2 \frac{P_A}{IPC_A^3}$$



Curves connect same architecture at different frequencies

Algorithm with Continuous DVS

Profiling Phase

For each arch A, measure IPC_A and P_A at one voltage/freq

Choose architecture with smallest P_A / IPC_A^3

This has smallest EPI for most values of Imax

Adaptation Phase

Predict instruction count for next frame

Use chosen architecture with frequency = $\frac{\text{Predicted instructions}}{\text{Deadline} \times IPC_A}$

Some modifications at minimum and maximum frequency

Instruction Count Predictor

Many predictors for execution time/processor utilization for DVS

- Based on various averages of various past frames

Metrics

- Prediction accuracy (most previous work)
- Minimize missed deadlines - under-predictions

Best compromise (< 5% under-predictions)

- MAXPAST(5) – Maximum of last 5 frames
  - Add leeway, hysteresis

Adaptation Overheads

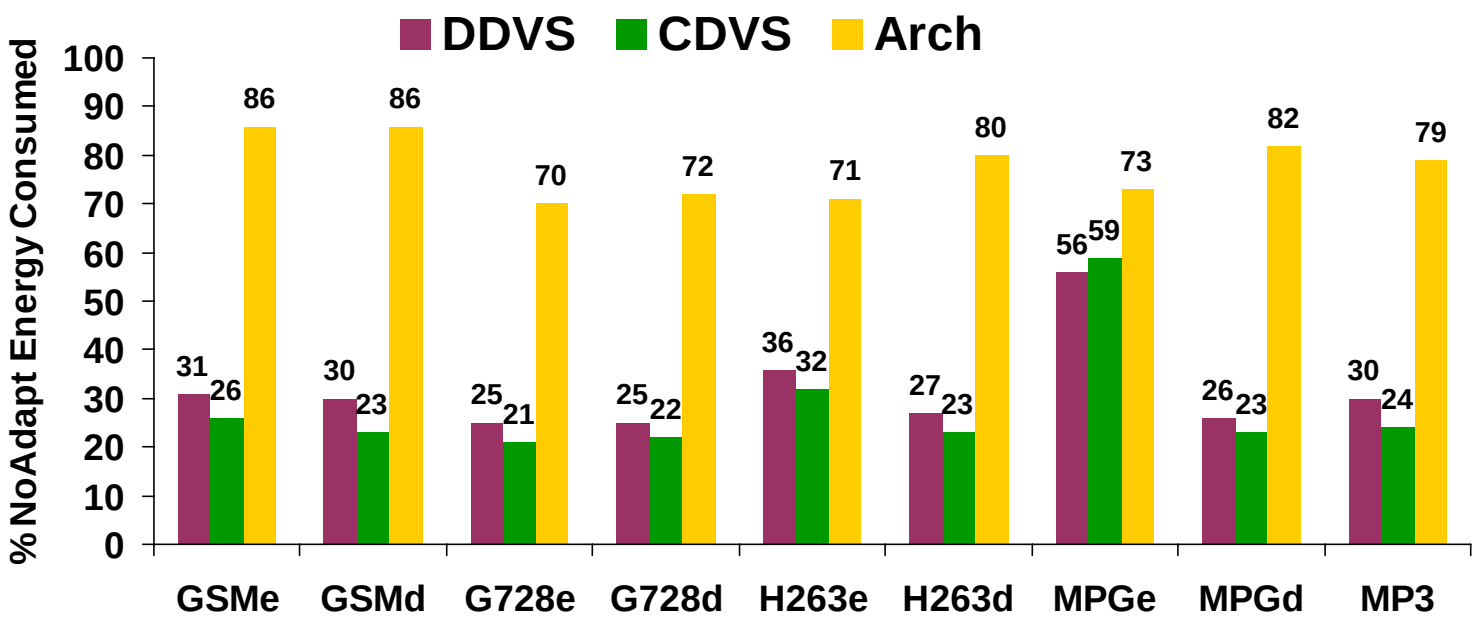
Frame granularity implies

- Adaptation overheads negligible for most cases

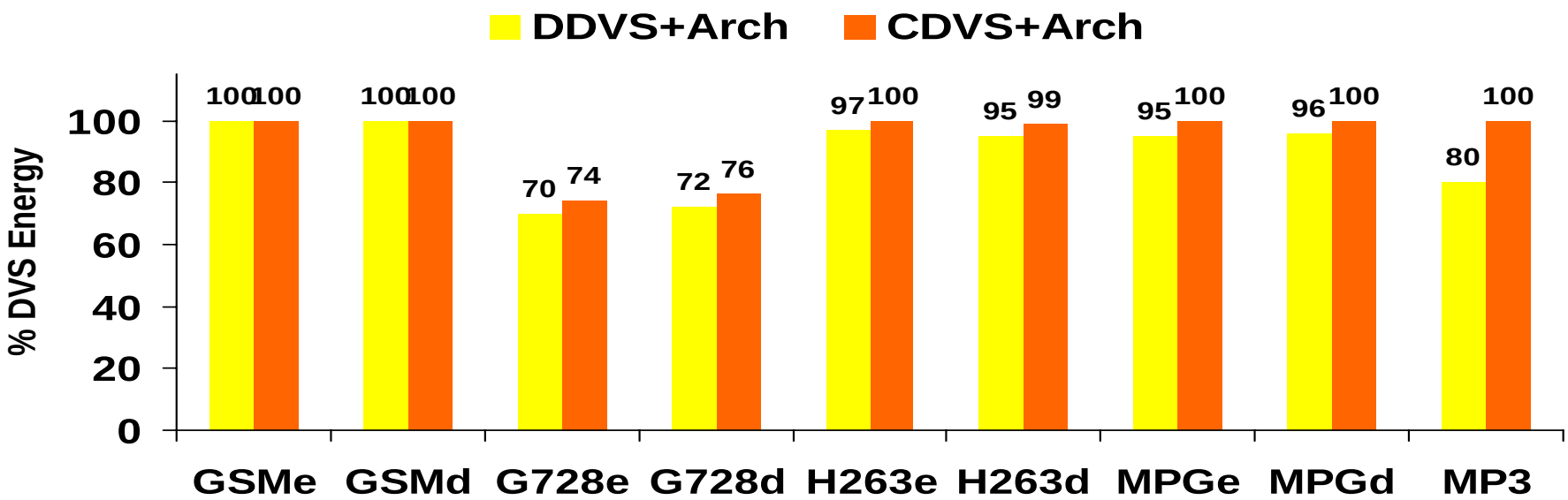
Can modify algorithm to include overheads

- But not done in this study

Energy Savings



DVS + Architectural Adaptation vs. DVS Alone



DVS + Arch most energy efficient

But DVS gives majority of gains for our suite

Savings from DVS + Arch vs. DVS alone

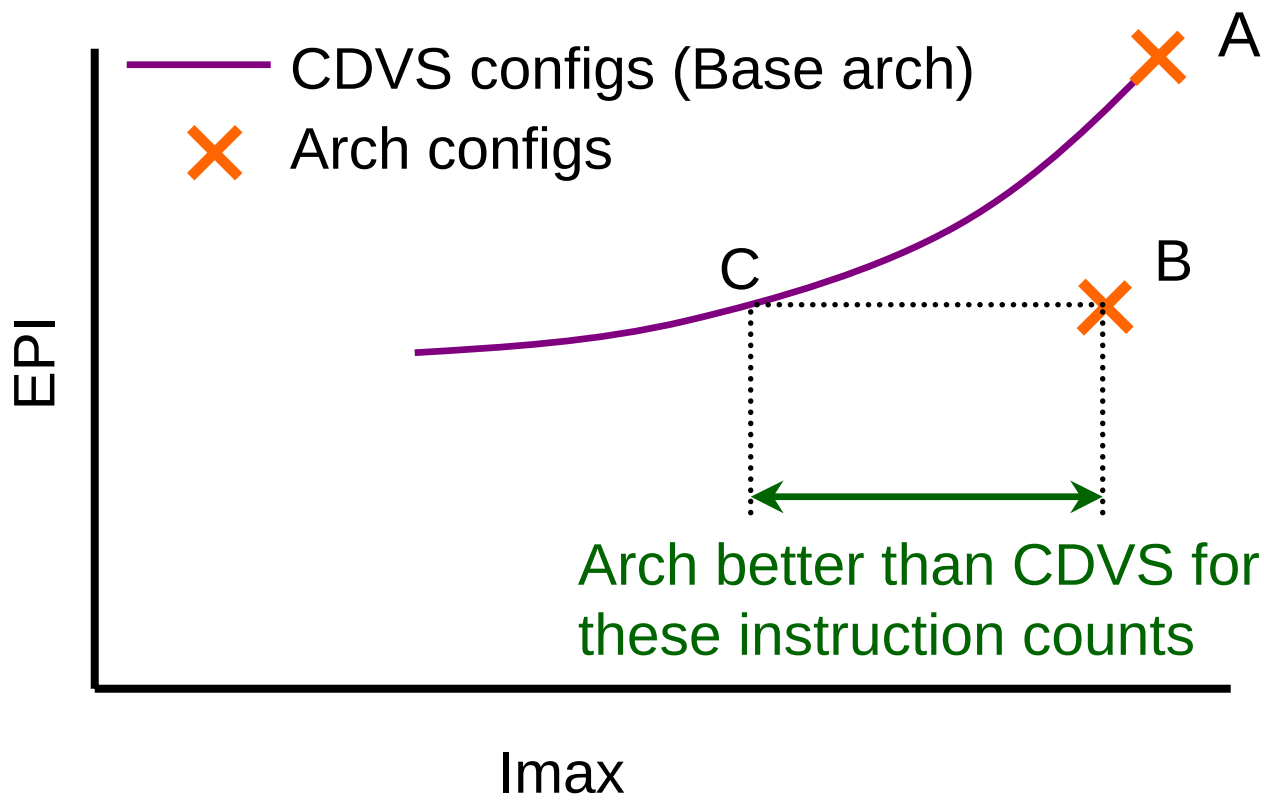
Average 5% to 17% (depend on deadline, C vs. D-DVS)

Max savings for a single app – 30%

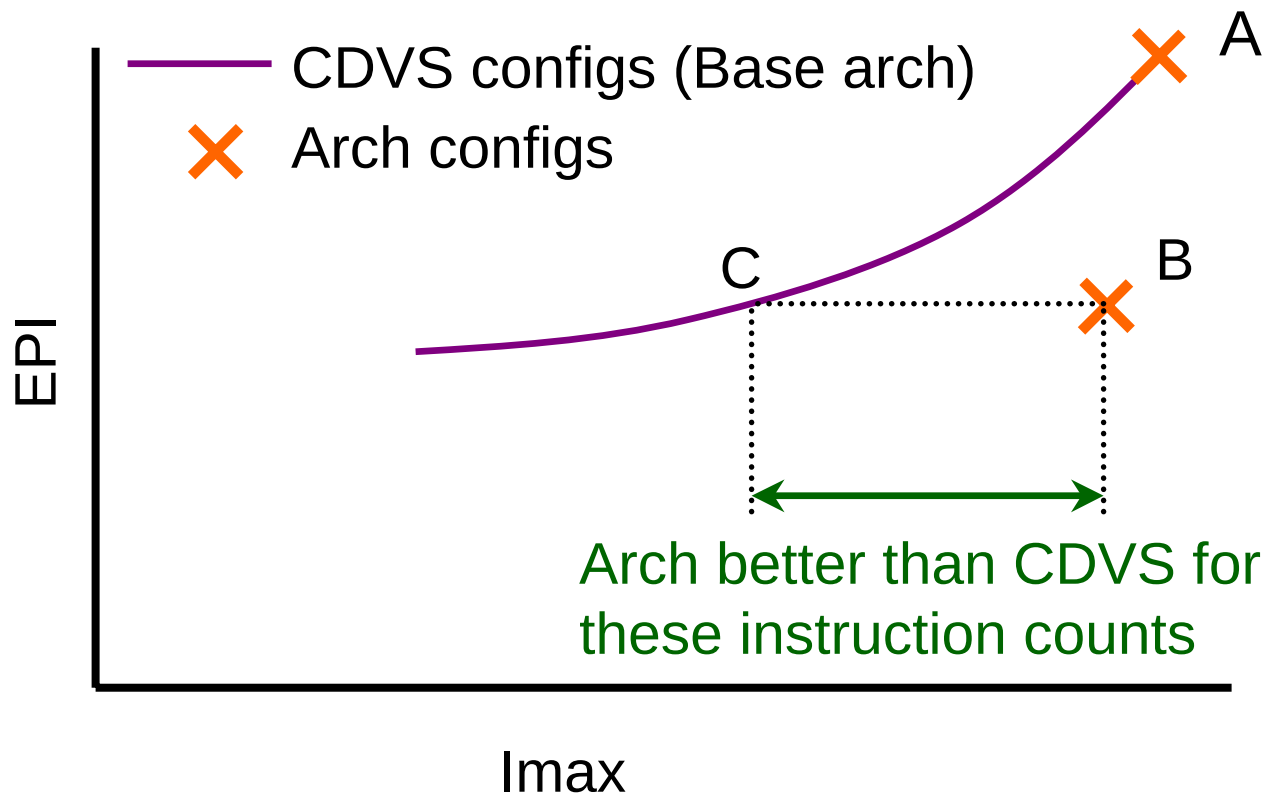
Arch vs. CDVS

When can Arch be better than CDVS?

EPI of Arch adaptation < Base *at all candidate frequencies*



Arch vs. CDVS



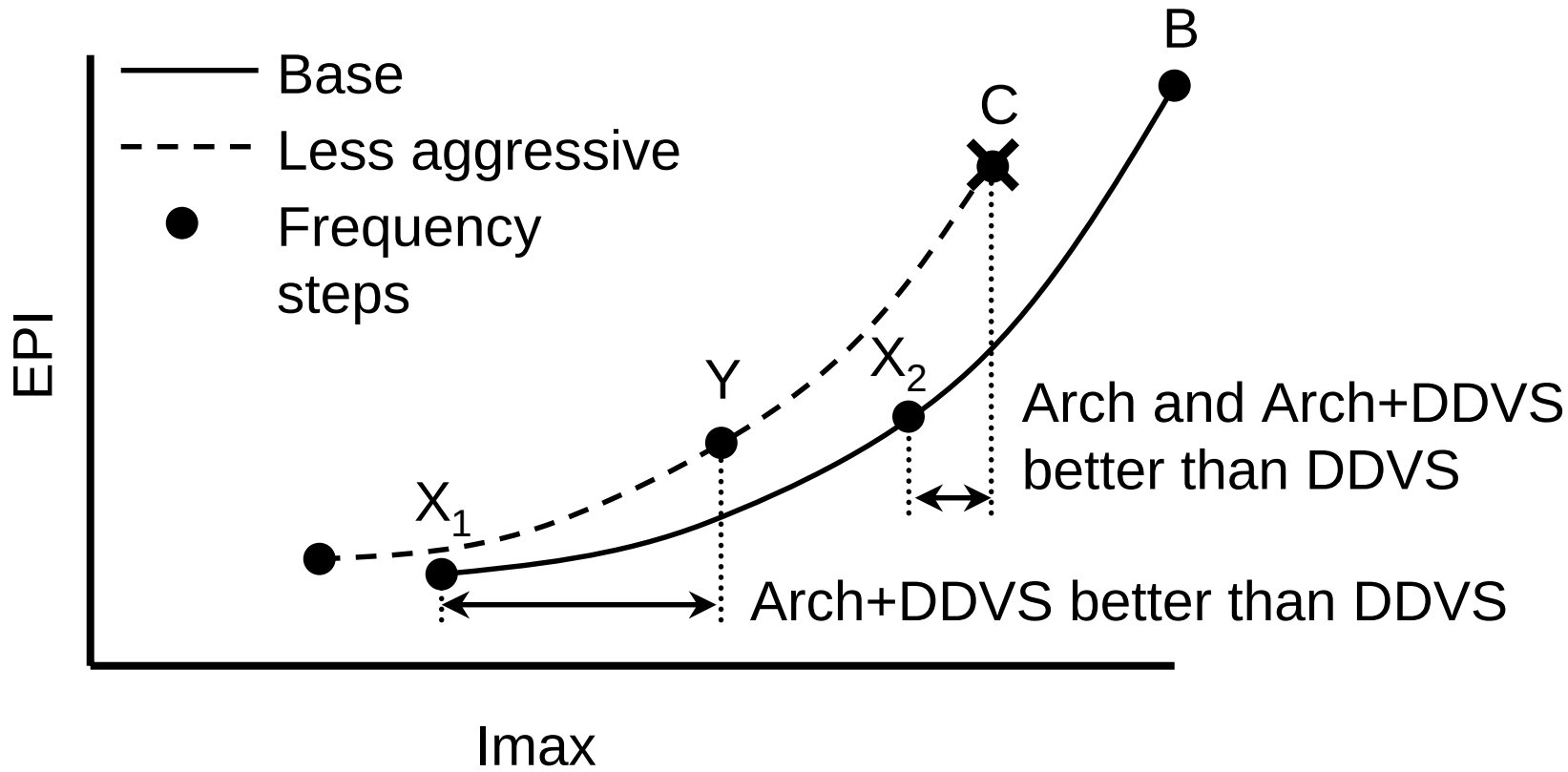
For G728, EPI of adapted architecture < Base for base frequency

But instruction count low ⇒ CDVS better

Even tighter deadlines could make Arch better

Arch + DDVS vs. DDVS

When is it effective to add Arch to DDVS?



Mean IPCs

Application	Mean IPC
GSMenc	4.09
GSMdec	3.41
G728enc	1.27
G728dec	1.32
H263enc	2.38
H263dec	2.69
MPGenc	2.70
MPGdec	3.48
MP3dec	1.63