

Two-Level Microprocessor-Accelerator Partitioning

Scott Sirowy, Yongui Yu, Stefano Lonardi, Frank Vahid*

Department of Computer Science and Engineering
University of California, Riverside
{ssirowy,yonghui,stelo,vahid}@cs.ucr.edu
*Also with the Center for Embedded Computer Systems at UC Irvine

This work was supported in part by the National Science Foundation and the Semiconductor Research Corporation



Introduction

HW/SW Partitioning

- Hw/sw partitioning can speedup software
 - Shown by numerous researchers
 - E.g., Balboni, Fornaciari, Sciuto CODES'96; Eles, Peng, Kuchchinski, Doboli DAES'97; Gajski, Vahid, Narayan, Gong Prentice-Hall 1997; Grode, Knudsen, Madsen DATE'98; many others
 - 1.5 to 10x common
 - Some examples like image processing get 100-800x speedup
 - E.g., Cameron project, FCCM'02
- Can reduce energy too
 - E.g.
 - Henkel, Li CODES'98
 - Wan, Ichikawa, Lidsky, Rabaey CICC'98
 - Stitt, Grattan, Villarreal, Vahid FCCM'02
 - 60-80% energy savings measured on real single-chip uP/FPGA devices

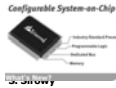
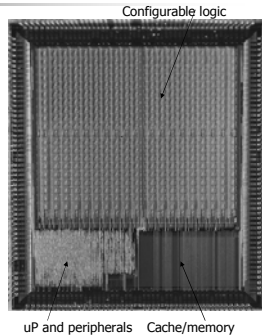
S. Sirowy

2/31

Introduction

Partitioning on Single Chip Platforms

- Numerous single-chip commercial devices with uP and FPGA
 - Triscend E5 (shown)
 - Triscend A7
 - Atmel FPSLIC
 - Xilinx Virtex II Pro
 - Altera Excalibur
 - More sure to come...
- Make hw/sw partitioning even more attractive

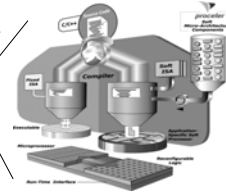


3/31

Introduction

Commercial Tools Evolving

- Commercial products evolving
 - Synopsys' Nimble compiler (2000) attempt
 - Proceler
 - Microprocessor Report's 2001 Technology of the Year Award
 - Others coming...



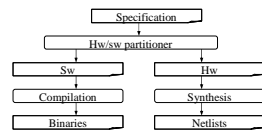
S. Sirowy

4/31

Introduction

Single-Spec Assumption

- Assumption – Start from a single specification
 - Typically sw source
- Partitioning
 - Find critical sw kernels, map some to hw
- This assumption is made in most research efforts as well as commercial tools



S. Sirowy

5/31

Introduction

Hardware/Software Partitioning

- Multi-processing Partitioning
 - Goal: Map a task graph to a number of concurrently executing microprocessors and custom processors.
 - MPEG-2
- Sequential Processing Partitioning
 - Create custom hardware to execute commonly executed functions
 - Critical Regions => Accelerators

S. Sirowy

6/31

Introduction

Architectural Model

- FPGA Technology
 - Hard/soft microprocessors
 - Supports multiple clock domains

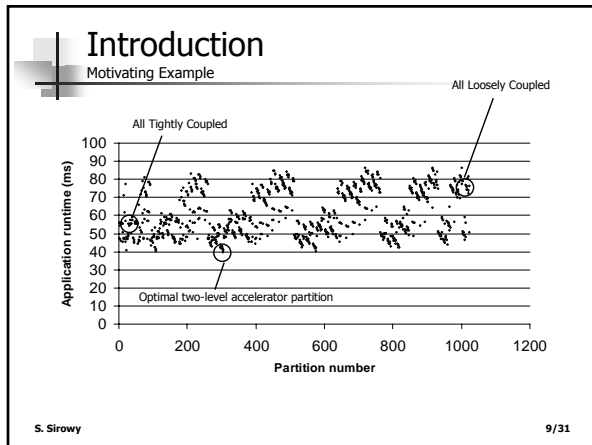
S. Sirowy 7/31

Introduction

Architectural Model

- Multilevel Bus Structures
 - Tightly Coupled(TC) Accelerators
 - Direct access to microprocessor or cache
 - Operate on the same frequency
 - Loosely Coupled(LC) Accelerators
 - Access microprocessor memory through a bridge

S. Sirowy 8/31

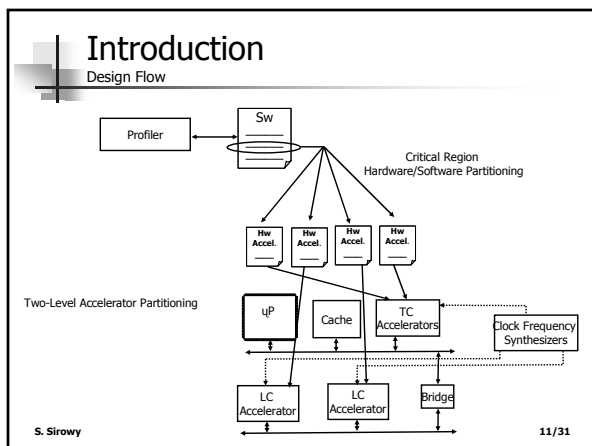


Introduction

Related Work

- Most work does not consider two levels of coupling
- Clock frequency interactions among tightly coupled accelerators
 - Assume single cycle access for all accelerators
 - Assume multiple cycles access
 - Associate an execution time with each accelerator without regard to clock frequency

S. Sirowy 10/31



Contribution

- Identification of the two-level accelerator partitioning problem
- Develop a fast algorithm suitable for integration in larger exploration environments

S. Sirowy 12/31

Problem Definition

- Given a set F of n functions to be implemented as hardware accelerators, for each we are given:
 - Computation Cycles
 - Memory Accesses
 - Clock Frequency
 - Area (LUT's)
- Maximize their performance (subject to an area constraint)
 - Minimize $TC([\sum comp_cycles_i + mem_accesses_i] / min_clock) + LC(d * \sum mem_accesses_i / clk_freq_i) + \sum (comp_cycles_i / clk_freq_i)$

S. Sirowy

13/31

Partitioning Solutions

- We need a fast, optimal solution that can be incorporated into a larger design space exploration
 - Greedy Heuristic
 - Dynamic Programming

S. Sirowy

14/31

N-Knapsacks Dynamic Programming

Algorithm

NKDP (S, n, d, A)

- $A \leftarrow$ Sort the accelerators in CP in the decreasing order of their frequencies
- $min_t \leftarrow \sum A_i.mem_accesses * d + A_i.clock_cycles / A_i.clock$
- $opt_sol \leftarrow \{\emptyset\}$
- for $i \leftarrow 1$ to n
 - $freq = A_i.freq$
 - for $j \leftarrow 1$ to $(i-1)$

$$V_j \leftarrow ((A_i.mem_accesses * f + A_j.comp_cycles) / A_j.clk_freq) - ((A_i.mem_accesses + A_j.comp_cycles) / freq);$$
 $W_j \leftarrow A_j.size;$
 - $S' = S - A_i.size$
 - $tmp_sol, tmp_t \leftarrow Knapsack01(V, W, S')$
 - $tmp_sol \leftarrow tmp_sol \cup A_i;$
 $tmp_t \leftarrow tmp_t + (A_i.mem_accesses + A_i.comp_cycles) / freq$
 - if $tmp_t < min_t$
 $min_t \leftarrow tmp_t, opt_sol \leftarrow tmp_sol$
- return opt_sol

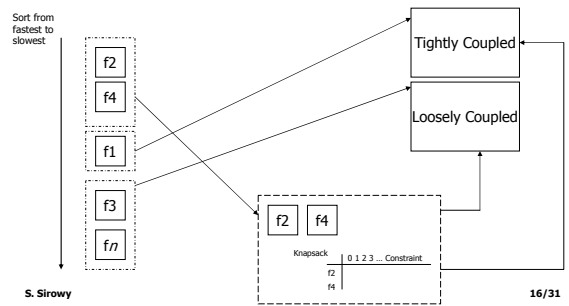
S. Sirowy

15/31

N-Knapsacks Dynamic Programming

NKDP

- 0-1 Knapsack Variant



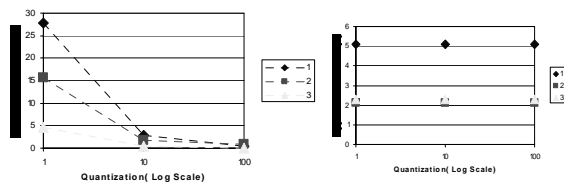
S. Sirowy

16/31

N-Knapsacks Dynamic Programming

NKDP Quantization

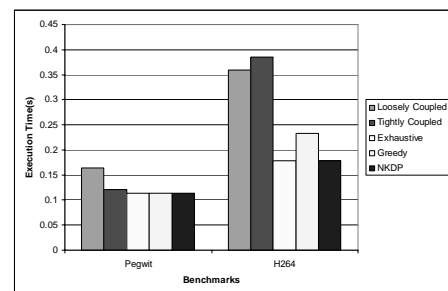
- NKDP scales to the area constraint parameter



S. Sirowy

17/31

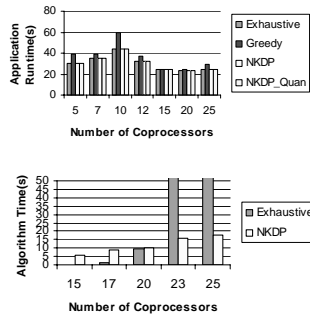
NKDP



S. Sirowy

18/31

Experimental Results



S. Sirowy

19/31

But...

- We assumed that the loosely coupled accelerators had access to an infinite amount of distinct clock signals
- Current FPGA platforms support at most 4-8 clock frequencies
- Can we partition the accelerators to the available number of clocks?

S. Sirowy

20/31

Clock-Frequency Assignment for Multiple Clock Domain Systems-on-a-Chip

Scott Sirowy, Yonghui Yu, Stefano Lonardi, Frank Vahid*

Department of Computer Science and Engineering
University of California, Riverside
{ssirowy, yonghui, stelo, vahid}@cs.ucr.edu

*Also with the Center for Embedded Computer Systems at UC Irvine

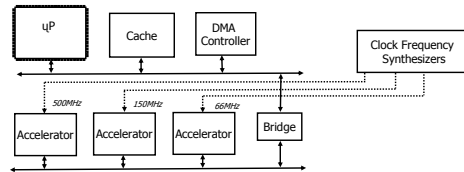
This work was supported in part by the National Science Foundation and the Semiconductor Research Corporation



Introduction

Architectural Model

- FPGA Technology
 - Hard/soft microprocessors
 - Supports multiple clock domains



S. Sirowy

22/31

H.264 Decoder

- Highly optimized video decoder developed by Freescale
- Stitt identified 42 critical functions that could benefit from accelerator speedup
 - Assumptions made
 - One clock frequency for all accelerators
 - One clock frequency per accelerator

S. Sirowy

23/31

H.264 Decoder

Function	SW Time(s)	# hw cycles	hw max clk freq (MHz)
<MotionComp_00>:	0.040733	1	281
<InvTransform4x4>:	0.034787	8	194
<FindHorizontalBS>:	0.025026	1	140
<GetBits>:	0.024681	1	200
<FindVerticalBS>:	0.02366	1	140
<MotionCompChromaFullXFully>:	0.023577	1	285
<FilterHorizontalLuma>:	0.023559	4	134
<FilterVerticalLuma>:	0.020008	4	138
<FilterHorizontalChroma>:	0.018803	4	134
<CombineZerosInvQuantScan>:	0.018438	1	120
<MotionCompensate>:	0.016822	10	40
<FilterVerticalChroma>:	0.016035	4	138
<MotionChromaFracXFracy>:	0.016023	32	78
<ReadLeadingZerosAndOne>:	0.015665	1	106

S. Sirowy

24/31

Problem Definition

- Given:
 - a set of accelerators $A = \{a_1, a_2, \dots, a_n\}$ each with the following attributes:
 - Clock Cycles, Max Clock Frequency, Clock Frequency
 - A maximum number of unique clock frequencies F
- Find a positive integer value for every a_i , freq, s.t each a_i , freq is less than a_i , max_freq for every I , the number of distinct a_i , freq values is less than or equal to F , and the execution time E is minimized

S. Sirowy

25/31

Problem Definition

Dynamic Programming Formulation

- Let $X(A,C)$ = total execution time of the first A accelerators using the first C clock frequencies.

```

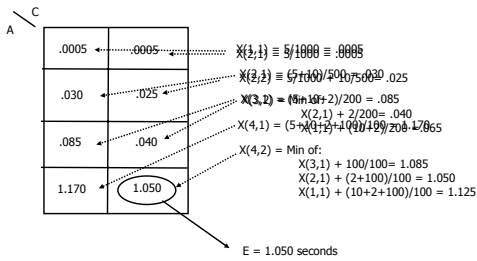
If (A = 0)
  then X(A,C) = 0
Else if (C=0)
  then X(A,C) = ∞
Else
  X(A,C) = MINi=1..A ((Σaj.cycles)/ai.max_freq) + X(i-1, C-1)
    
```

S. Sirowy

26/31

A Simple Example

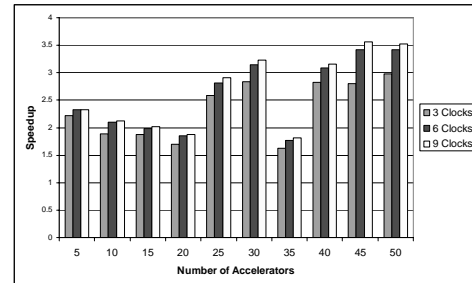
	a_1	a_2	a_3	a_4	Number of Available Clocks = 2
Max Freq:	1000	500	200	100	
Cycles:	5	10	2	100	



S. Sirowy

27/31

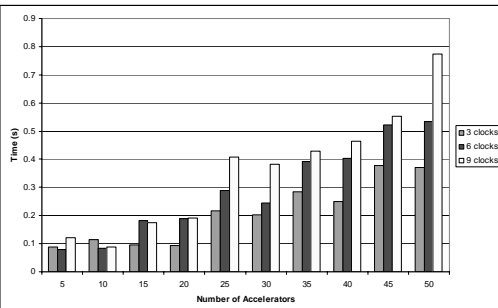
Results- Application Speedups



S. Sirowy

28/31

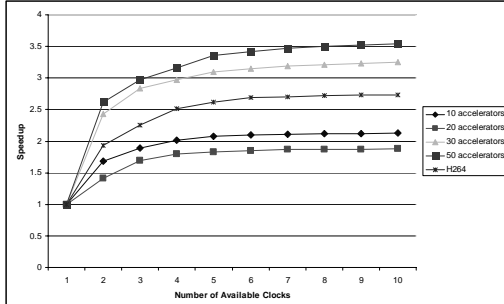
Results- Algorithm Runtimes



S. Sirowy

29/31

Results



S. Sirowy

30/31

Conclusions and Future Work

- Conclusions
 - Application speedups from 1.5x-4x for applications with 5-50 accelerators
 - Novel dynamic programming algorithm
- Future Work
 - Integration of clock frequency assignment with other exploration methods
 - Wider range of clock domain usage scenarios