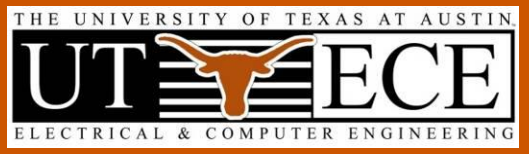


PACT 2010

SYstem-level Max POwer (SYMPO) - A Systematic Approach for Escalating System-level Power Consumption using Synthetic Benchmarks

K. Ganesan, J. Jo, W. L. Bircher, D. Kaseridis, Z. Yu and L. K. John
University of Texas at Austin

Worst-case Power Consumption

- **Understanding worst-case power characteristics**
  - Power management features, designing cooling system, heat sinks, voltage regulators
- **Practically attainable maximum power**
  - If set too high => wastage of resources & If set too low => reliability issues
  - Design of “power viruses”
- **Not just the cores, system-level power virus**
  - Trend towards integrating more components into chip

Industry-grade Max-power Viruses

- **Hand crafting code snippets for power viruses**
  - Very tedious process, complex interactions inside the processor
  - Cannot be sure if it is the maximum case
- **We automatically generate power viruses**

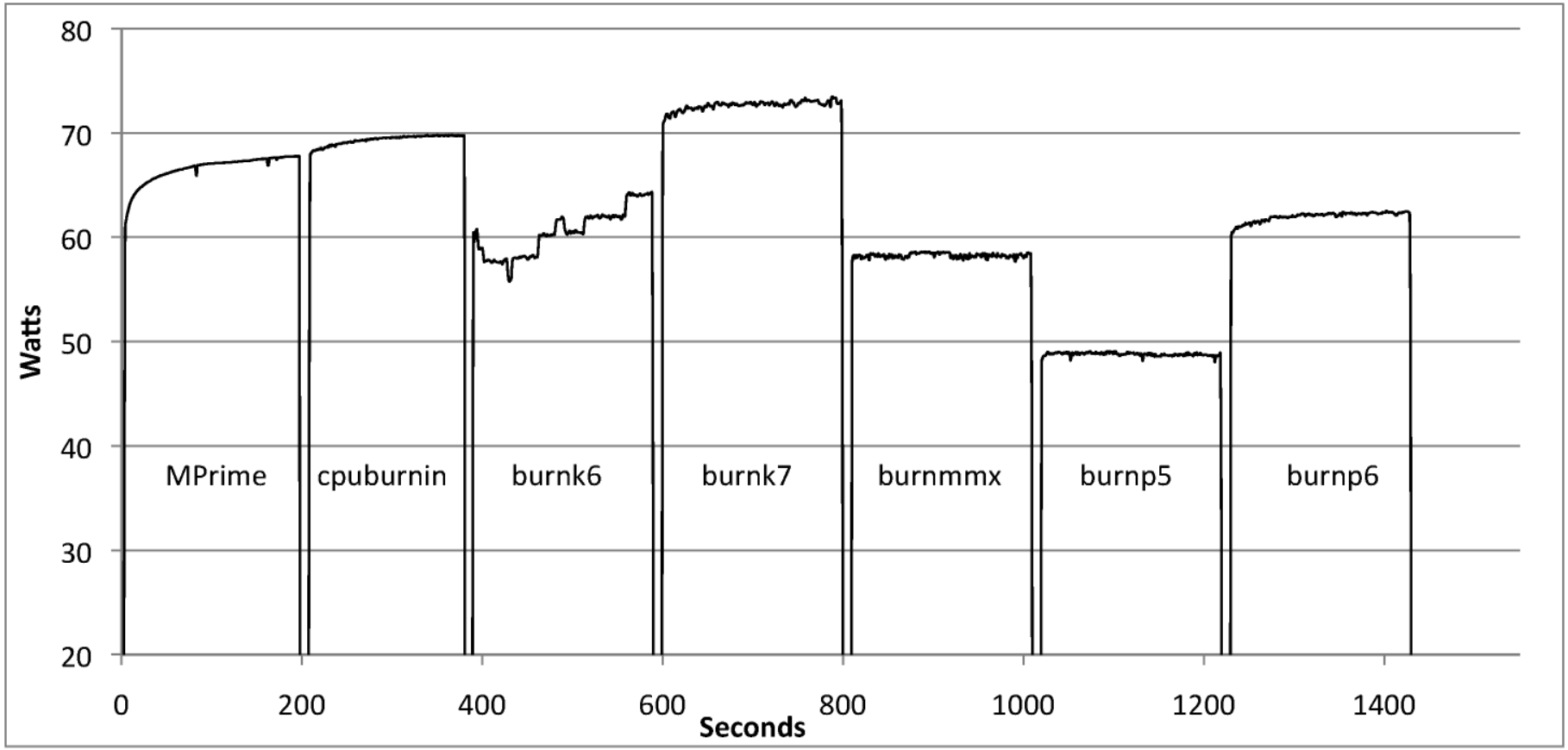
Power Virus	Optimized to stress	Language
MPrime	All CPUs, all ISAs	C
CPUburn-in	X86 machines	x86 assembly
BurnP5	Intel Pentium w&w/o MMX processors	x86 assembly
BurnP6	Intel PentiumPro, Pentium II, Pentium III and Celeron CPUs	x86 assembly
BurnK6	AMD K6 processors	x86 assembly
BurnK7	AMD Athlon/Duron processors	x86 assembly
BurnMMX	cache/memory interfaces on all CPUs with MMX	x86 assembly

Measurement on Hardware

- **Power characteristics on AMD Phenom II X4 (K10)**
- **AMD-designed system board**
  - Fine-grain power instrumentation for CPU core
  - Hall effect current sensor provides 0-5 V signal
  - National Instruments PCI-6255 data logger samples current and voltage

Cores	4
Clock	3.0 GHz
Technology	45 nm
IL1, DL1 Cache	2-way, 64 B line, 64 KB per core
L2 Cache	16-way, 64B line, 512 KB per core
L3 Cache	32-way, 64B line, 6MB shared
Fetch size	32 bytes
Branch Predictor	12 global history, 16 K bi-mod predictor 2 K target buffer
ROB	72 entries in 3 micro-op groups
ALU	3 symmetry Int ALUs, 3 FP ALUs

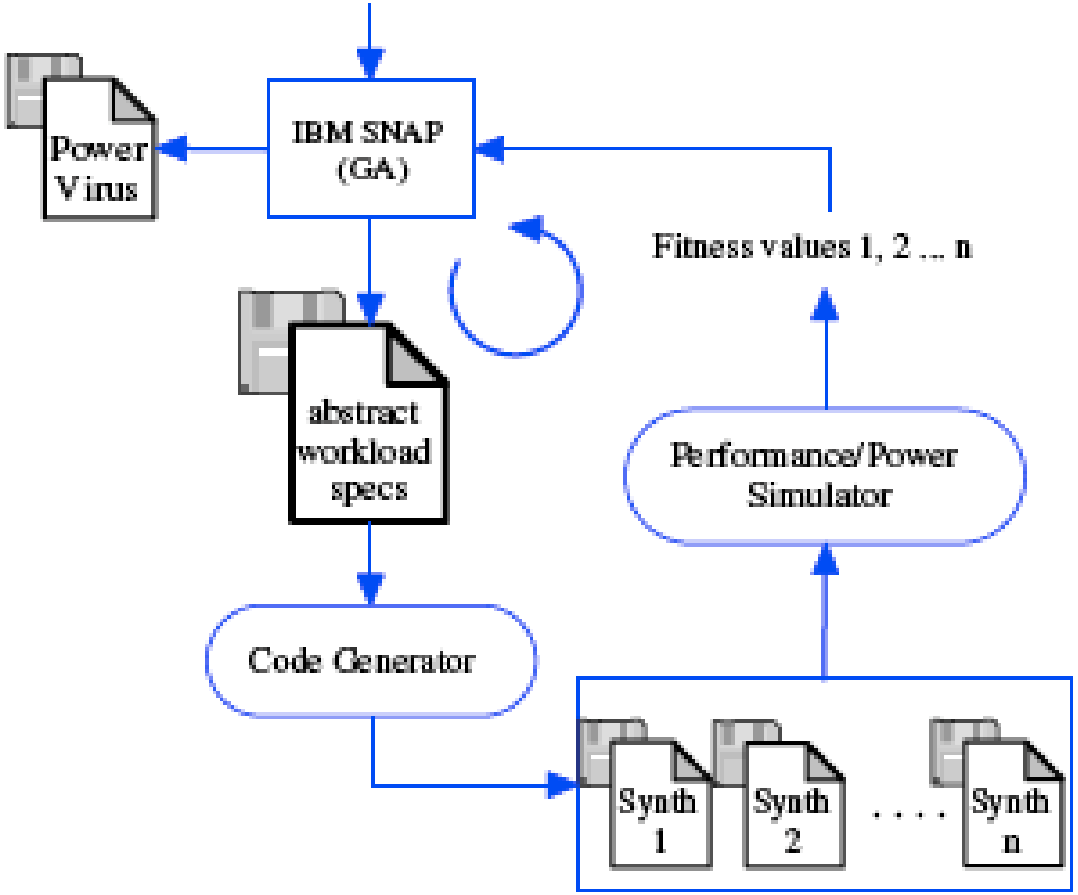
Power measurement on Hardware



- **BurnK7 – 72.1 Watts**
- **SPEC CPU2006: 416.gamess and 453.povray consume highest power of 63.1 and 59.6 Watts**

SYMPO Framework

- **Automatically search for power viruses using an abstract workload model and machine learning**



- **GA: search heuristic to solve optimization problems**
- **Choose a random population, evaluate fitness, apply GA operators to generate next population**
- **Evolve until required fitness achieved**

Abstract Workload Model

! " #$	%&' ()*$	+, - . &$	/, '& " (0$
1$	! 23 4&($ 5$4, 6)*$47' *86$	19:$ 9:$<9:$199:=99$	/" - '(" 7$57' G $ B(&C)*', 4)7)' 0$
=$	$>?&(, . &$4, 6)*$47' *8$6)@&$	19:$ 9:$<9:$199$	
; $	$>?&(, . &$' ?&(, 7$4(, - *A$ B(&C)*', 4)7)' 0$	9#D:$9#DE:$9#F=:$$ 9#FE:$9#FD:$9#FF:$1#9$	
"!	!#$%&' &(!) *+!,$-%. /%0$!1 &,' 2%	3!4"!	#$-%. /%0$!! 6 ,F!
5!	!#$%&' &(!6 . 7.,$-%. /%0$!1 &,' 2%	3!4"!	
8!	!#$%&' &(!9,: !,$-%. /%0$!1 &,' 2%	3!4"!	
; !	!<=!>99!,$-%. /%0$!1 &,' 2%	3!4"!	
?!	!<=!6 . 7.,$-%. /%0$!1 &,' 2%	3!4"!	
@!	!<=!9,: !,$-%. /%0$!1 &,' 2%	3!4"!	
A3!	!<=!6 0: !,$-%. /%0$!1 &,' 2%	3!4"!	
AA!	!<=!-B(%$-%. /%0$!1 &,' 2%	3!4"!	
AC!	!*0>9!,$-%. /%0$!1 &,' 2%	3!4"!	
AD!	!E%0(&!,$-%. /%0$!1 &,' 2%	3!4"!	
A"!	G&' ,-%&(!9&H&$9&$/I !9,-%$/&! 9,-%,J . %0$!K$. 6 J &(!0L! ,$-%. /%0$-M!	AN:CN:N' N? N:ACN:A8N:C3N: DCN:N' ? N:8"!	#$-%. /%0$! 7&: &7. H>(>7&7,-6 !
A5!	*0/>7.-%,9&: >7. &H&(!-%%/! 70>9O-%0(&P!A3!J . /Q&%!&>/2!1 ,%2! &B. >7.H(0J >J ,7,%!	!3N:N' N? N:ACN:A8N:DCN: 8"!,$!&>/2!J . /Q&%	R>%!70/>7,%!T! 6 &6 0(I !7&: &7. H>(>7&7,-6 !
A8!	R>%!L00%H(,$%K$. 6 J &(!0L! ,%&(>%0$-!J &L0(&!(&-&%%0!! J &',$$,$' !0L%2&!>((>I M!	AN:A3N:C3N:N' 3N:A33N: C33!!	
A;!	S &>$!0L!6 &6 0(I !7&: &7.H>(>7&7,-6 !	AN:CN:DN:8!	

Abstract Workload Model

! " #$	%&' ()*$	+, - . &$	/, '& " (0$
" !	#$%&' (!) *!&+, -. !&/) . 0, !	" 12312412" 1125111!	D) : >() /!*/) E ! <(' =-. >+&-/->?!
5!	!67' (+8' !&+, -. !&/) . 0!, -9' !	" 12312412" 11!	
3!	!67' (+8' !) 7' (+//!&(+: . ; ! <(' =-. >+&-/->?!	1@21@B21@52! 1@B21@A21@C2" @!	
1$	$2- '& &($345$- 6'(7*')" - $3 &). 9'$	: $$1$	2- 6'(7*')" - $$ =)L$
<$	$2- '& &($= 7>$- 6'(7*')" - $3 &). 9'$	: $$1$	
?$	$2- '& &($@A$- 6'(7*')" - $3 &). 9'$	: $$1$	
B$	CD @@$- 6'(7*')" - $3 &). 9'$	: $$1$	
E$	CD= 7>$- 6'(7*')" - $3 &). 9'$	: $$1$	
F$	CD@A$- 6'(7*')" - $3 &). 9'$	: $$1$	
G: $	CD= "A$- 6'(7*')" - $3 &). 9'$	: $$1$	
GG$	CD3H('$- 6'(7*')" - $3 &). 9'$	: $$1$	
GI $	$4", @$- 6'(7*')" - $3 &). 9'$	: $$1$	
GJ $	$K"(&$- 6'(7*')" - $3 &). 9'$	: $$1$	
" F!	G' 8-, >' (!=' <' : =' : . ?!=-, >+: . ' ! =-, >(-&$>) : !H $%&' (!) *!! -: , >($. >) : , I!	" 252F2A2" 52" B25122 352FA2BF!	J: , >($. >) : ! / 7' /! <+(+// /-, %!
" 4!	K) . +/!, >(-=' !7+/$' !<' (!, >+>. ! /) +=L, >) (' @' 1!&$. 0' >, !' +. ; !E ->, ! ' M$+/!<() &+&-/->?!	!12F2A2" 52" B23522! BF!-: !' +. ; !&$. 0' >!	N+>+!/) . +/->?!Q! %' %) (?!/ 7' /! <+(+// /-, %!
" B!	N+>+!*)) >(-: >!!H $%&' (!) *!! ->' (+>) : , !&' *) (' !(' , ' >!>) !! &' 8-: : -: 8!) *!>, ' !+((+?I!	" 2" 12512F12" 1125111!!	
" O!	P ' +: !) *!%' %) (?!/ 7' /!<+(+// /-, %!	" 252322B!	

Abstract Workload Model

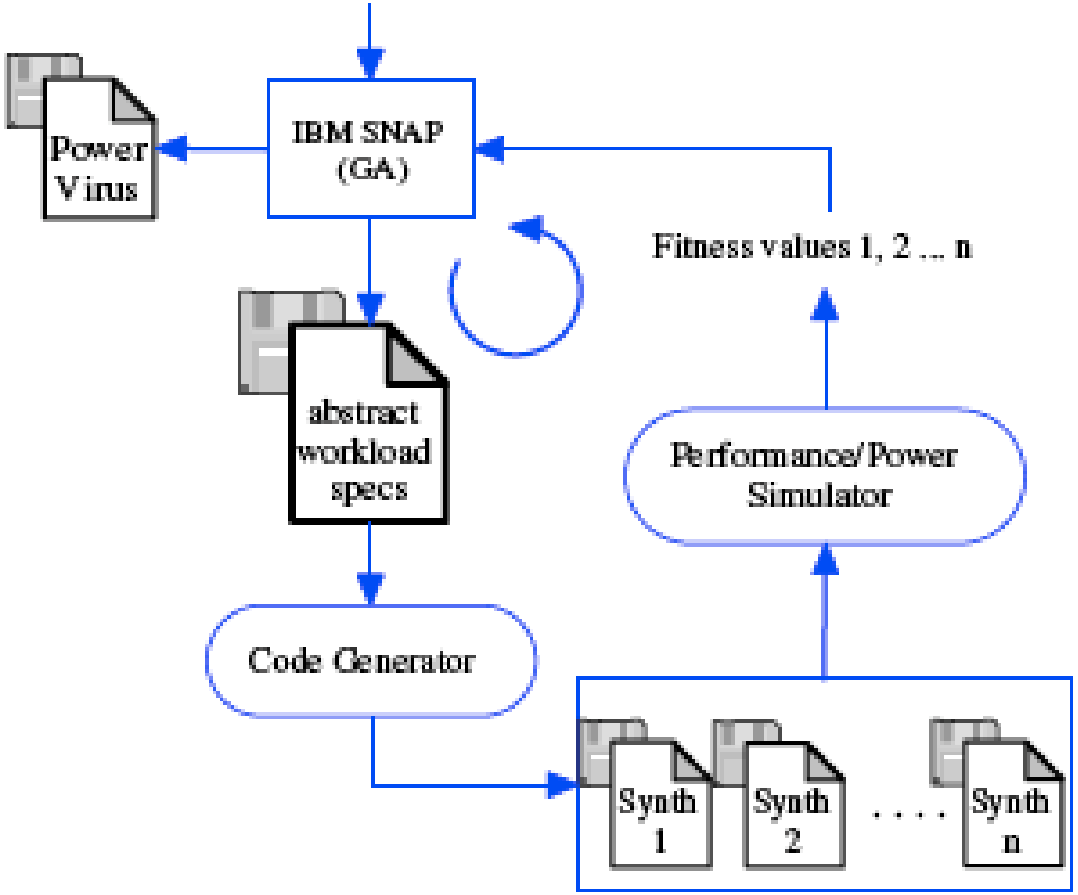
! " #$	%&' ()*$	+, - . &$	/, '& " (0$
" !	#$%&' (!) *!&+, -. !&/) . 0, !	" 12312412" 1125111!	
5!	!67' (+8' !&+, -. !&/) . 0!, -9' !	" 12312412" 11!	D) : >() /!*/) E !
3!	!67' (+8' !) 7' (+//!&(+: . ; ! <(' =-. >+&-/->?!	1@21@B21@52! 1@B21@A21@C2" @!	<(' =-. >+&-/->?!
F!	!G >' 8' (!6HI !-: , >($. >) : !E ' -8; >!	1!J!F!	
4!	!G >' 8' (!%$/!-: , >($. >) : !E ' -8; >!	1!J!F!	
B!	!G >' 8' (!=-7!-: , >($. >) : !E ' -8; >!	1!J!F!	
K!	!LM+==!-: , >($. >) : !E ' -8; >!	1!J!F!	
A!	!LM%$/!-: , >($. >) : !E ' -8; >!	1!J!F!	G , >($. >) : !!
C!	!LM=-7!-: , >($. >) : !E ' -8; >!	1!J!F!	%-P!
" 1!	!LM%) 7!-: , >($. >) : !E ' -8; >!	1!J!F!	
" "!	!LM, N(>!-: , >($. >) : !E ' -8; >!	1!J!F!	
" 5!	!H) +=!-: , >($. >) : !E ' -8; >!	1!J!F!	
" 3!	!O>) (' !-: , >($. >) : !E ' -8; >!	1!J!F!	
12$	**+&)3' &($1&5& 4&- *0$1)3', - *&$ 4)3' ()67')" - $3- 79 6&($':$)- 3' (7*')" - 3;$**	**1<$-<$2<$><$1=<$1?<$=@$ A=<$2><$?2$**	**B 3' (7*')" - $ C&D&C$ 5, (, C&C)39 $**
" 4!	H) . +/!, >(-=' !7+/$' !<' (!, >+>. ! /) +=Q, >) (' @?' 1!&$. 0' >, !' +. ; !E ->, ! ' N$+/!<() &+&-/->?!	!12F2A2" 52" B2352! BF!-: !' +. ; !&$. 0' >!	R+>+!/) . +/->?!V! %' %) (?!/ 7' /! <+(+// /-, %!
" B!	R+>+!*)) ><(-: >!S: $%&' (!) *!! ->' (+>) : , !&' *) (' !(' , ' >!>) !! &' 8-: : -: 8!) *!>, ' !+((+?T!	" 2" 12512F12" 1122 511!!	
" K!	U ' +: !) *!%' %) (?!/ 7' /!<+(+// /-, %!	" 2523 2B!	

Abstract Workload Model

! " #$	%&' ()*$	+, - . &$	/, '& " (0$
" !	#$%&' (!) *!&+, -. !&/) . 0, !	" 12312412" 1125111!	
5!	!67' (+8' !&+, -. !&/) . 0!, -9' !	" 12312412" 11!	D) : >() /!*/) E ! <(' =-. >+&-/->?!
3!	!67' (+8' !) 7' (+//!&(+: . ; ! <(' =-. >+&-/->?!	1@A21@AB21@52! 1@B21@A21@C2" @!	
F!	!G >' 8' (!6HI !-: , >($. >) : !E ' -8; >!	1!J!F!	G , >($. >) : !! %-P!
4!	!G >' 8' (!%$/!-: , >($. >) : !E ' -8; >!	1!J!F!	
B!	!G >' 8' (!=-7!-: , >($. >) : !E ' -8; >!	1!J!F!	
K!	!LM+==!-: , >($. >) : !E ' -8; >!	1!J!F!	
A!	!LM%$/!-: , >($. >) : !E ' -8; >!	1!J!F!	
C!	!LM=-7!-: , >($. >) : !E ' -8; >!	1!J!F!	
" 1!	!LM%) 7!-: , >($. >) : !E ' -8; >!	1!J!F!	
" "!	!LM, N(>!-: , >($. >) : !E ' -8; >!	1!J!F!	
" 5!	!H) +=!-: , >($. >) : !E ' -8; >!	1!J!F!	
" 3!	!O>) (' !-: , >($. >) : !E ' -8; >!	1!J!F!	
" F!	Q' 8-, >' (!=' <' : =' : . ?!=-, >+: . ' ! =-, >(-&$>) : !R $%&' (!) *!! -: , >($. >) : , S!	" 252F2A2" 52" B2512 352FA2BF!	G , >($. >) : ! / 7' /! <+(+// /-, %!
12$	3" *, 45' ()6&$7, 48&$9&($', ')*$ 4', 6: 5' " (&#$1; $<8*=&' 5$&, *>$?')' >$ &@8, 49(" <, <)4)' 0$	$ ABACA$1DA$1EAFDA EB$)- $&, *>$<8*=&' $	G, ', $#' *, 4)' 0$M$ J &J " (0$1&7&4$ 9, (, 41&4)5J $
1E$	G, ', $H' " ' 9()- ' $1- 8J <&($' H$)' &(, ')" - 5$<&H' (&$(&5&' $" $ <&)- -)- . $' H$>&$) ((, 0K$	1A$1; A$D; A$B; A$1; ; A$ D; ; $	
1L$	%&, - $' HJ &J " (0$1&7&49, (, 41&4)5J $	1ADAFAE	

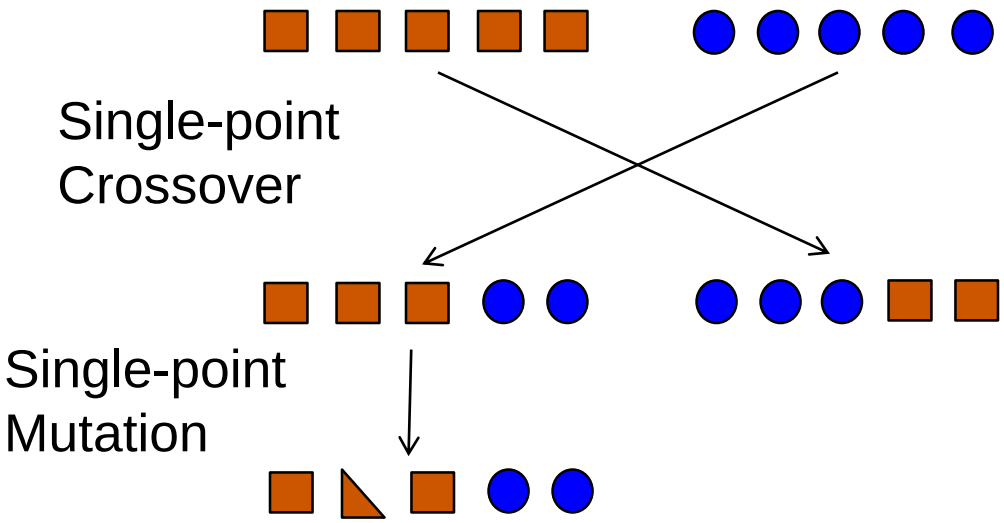
SYMPO Framework

- **Automatically search for power viruses using an abstract workload model and machine learning**



- **GA: search heuristic to solve optimization problems**
- **Choose a random population, evaluate fitness, apply GA operators to generate next population**
- **Evolve until required fitness achieved**

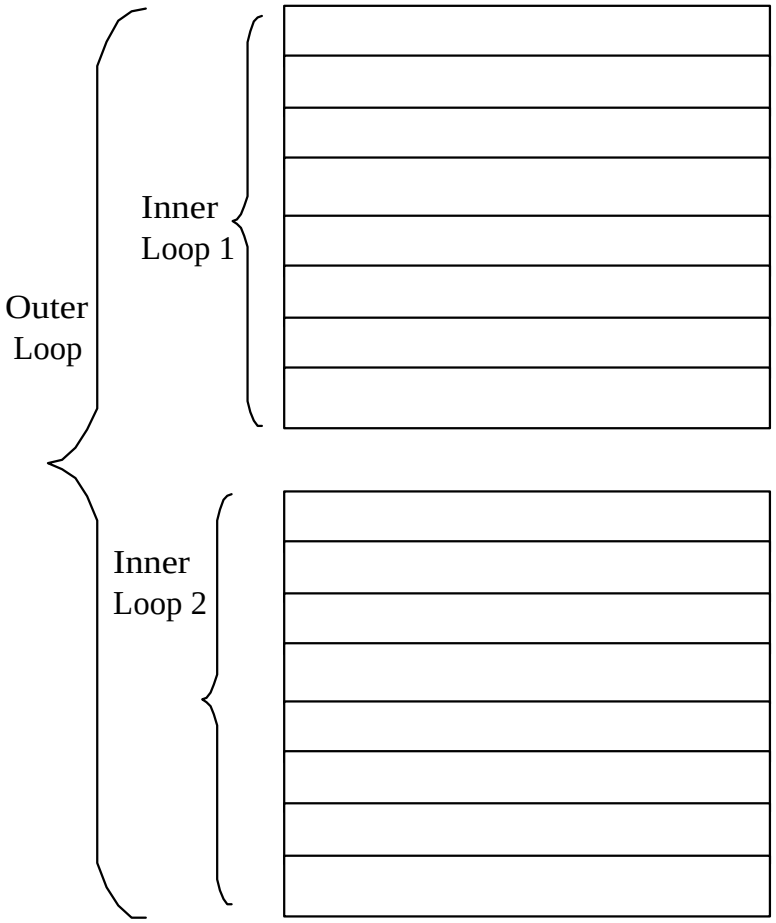
SYMPO Framework – Genetic Algorithm, IBM SNAP



- Individuals -> synthetic workloads,
- Fitness function -> power on the design under study
- Mutation rate, reproduction rate, crossover rate

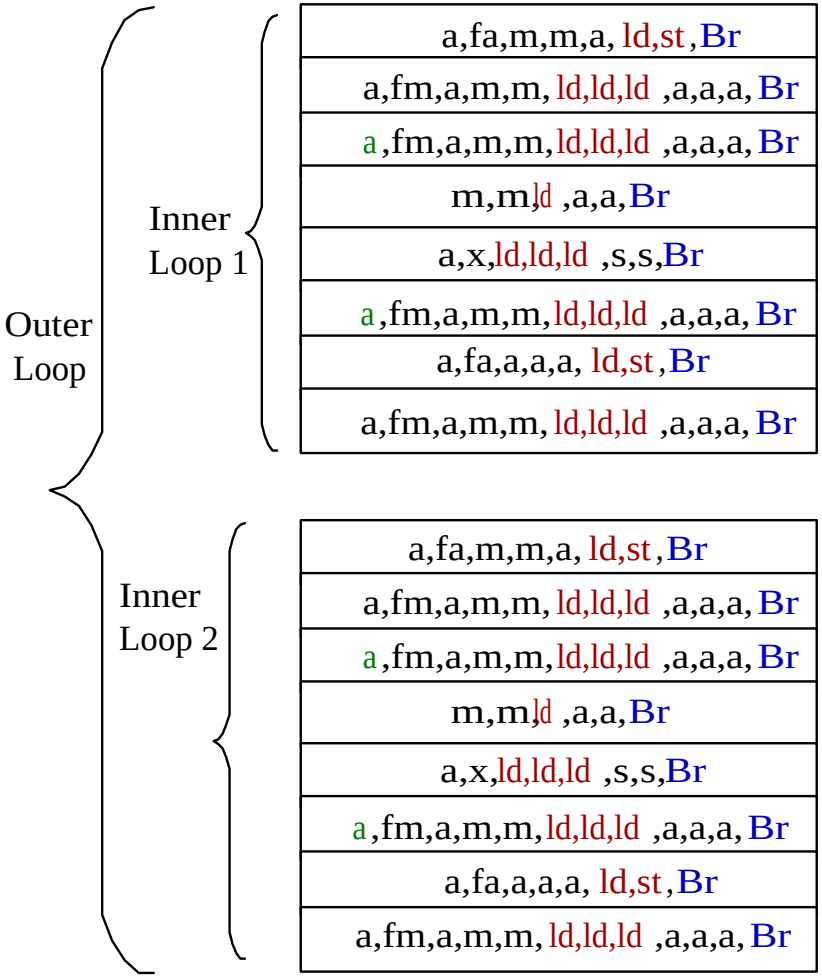
Code Generation

- **Step 1:**
  - Fix the number of basic blocks in the synthetic



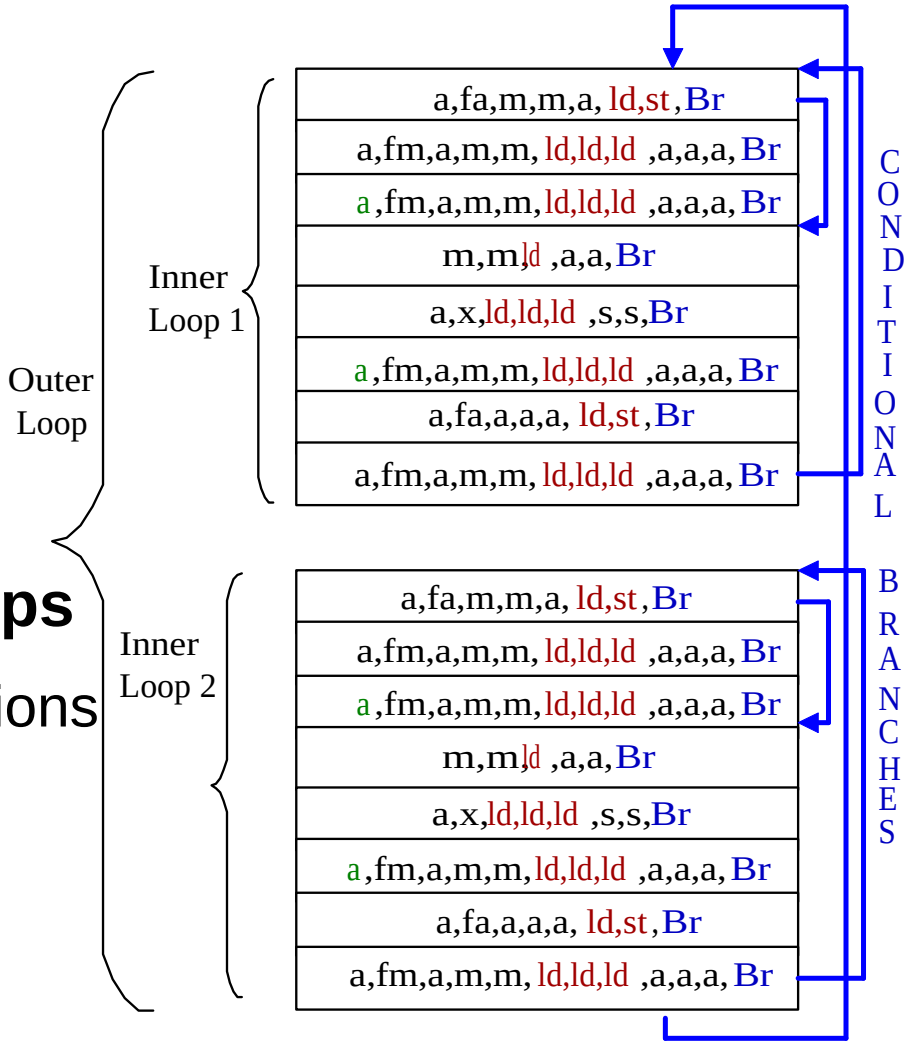
Code Generation

- **Step 1:**
  - Fix the number of basic blocks in the synthetic
- **Step 2: For each Basic Block**
  - Choose the instruction type for every instruction using the global Instruction mix



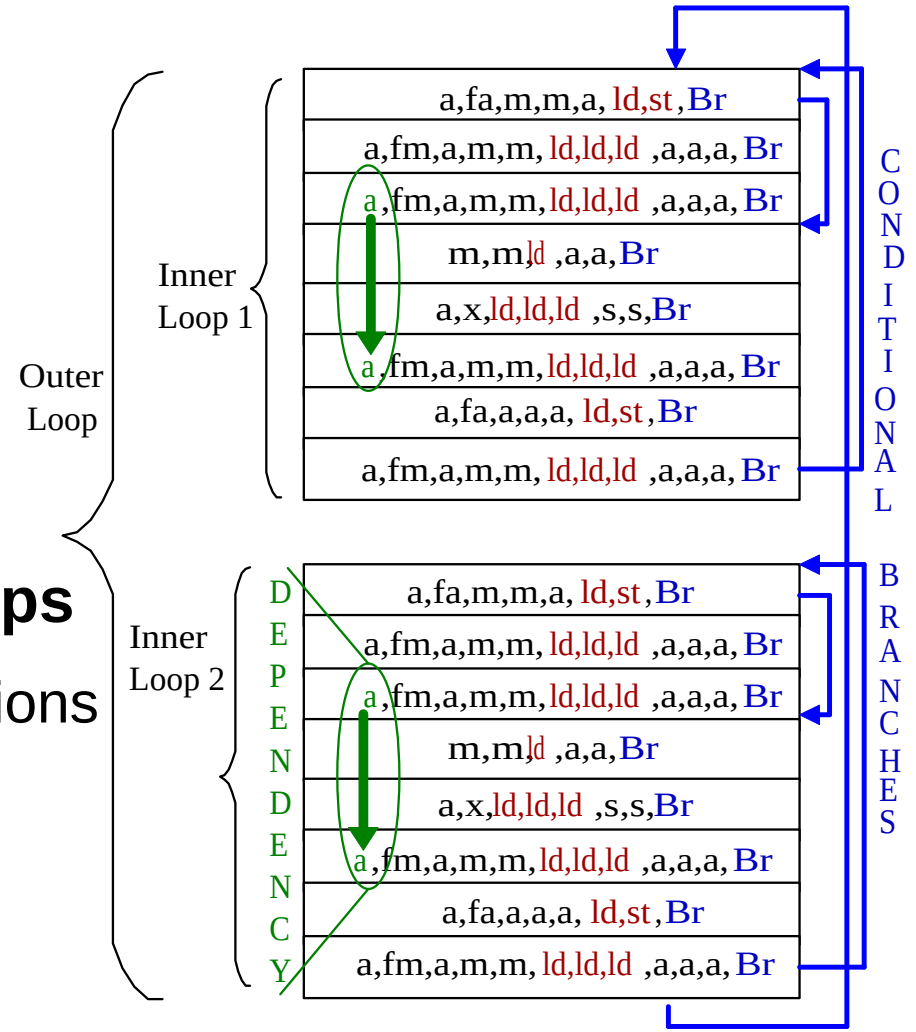
Code Generation

- **Step 1:**
  - Fix the number of basic blocks in the synthetic
- **Step 2: For each Basic Block**
  - Choose the instruction type for every instruction using the global Instruction mix
- **Step 3: Bind the basic blocks together using conditional jumps**
  - Group into pools & modulo operations



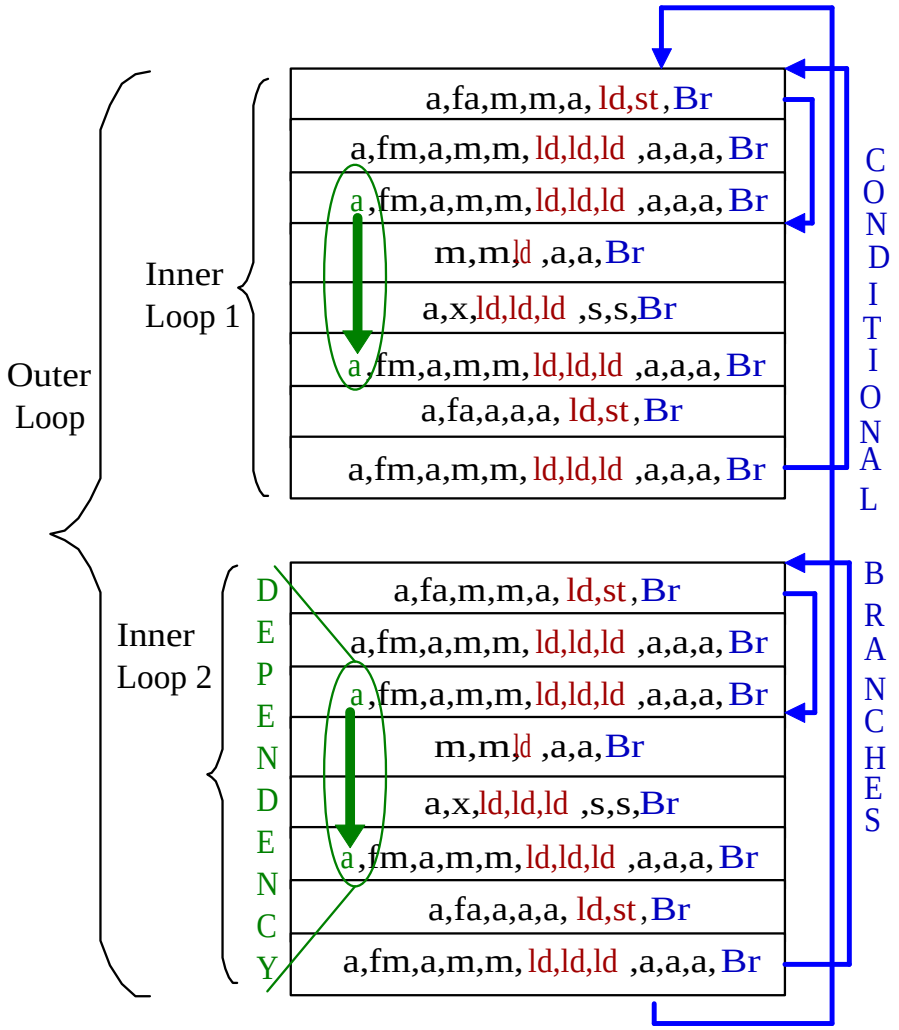
Code Generation

- **Step 1:**
  - Fix the number of basic blocks in the synthetic
- **Step 2: For each Basic Block**
  - Choose the instruction type for every instruction using the global Instruction mix
- **Step 3: Bind the basic blocks together using conditional jumps**
  - Group into pools & modulo operations
- **Step 4: For each instruction**
  - Find a producer inststruction to assign a register dependency
  - Not compatible? Move up/down



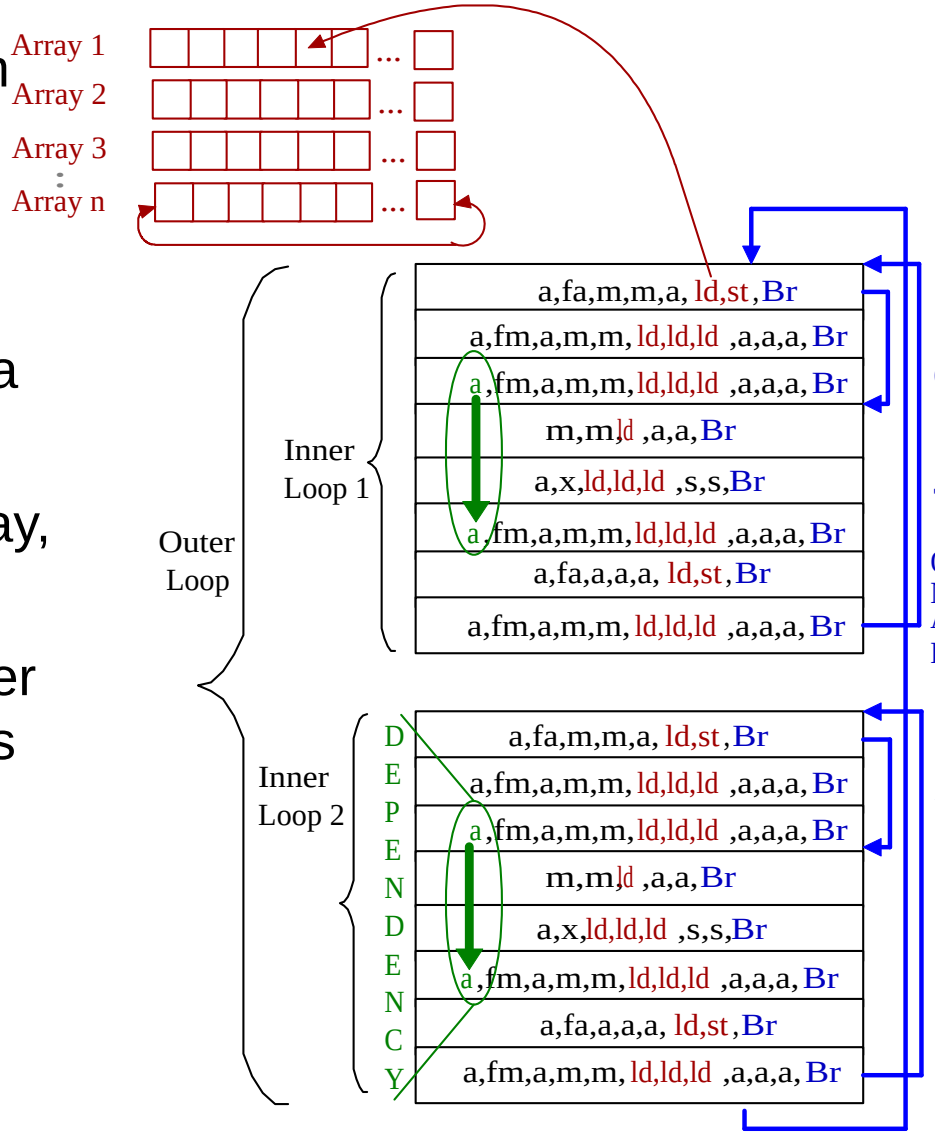
Code Generation

- **Step 5: Assign registers**
  - Destination registers – RoundRobin
  - Source registers – based on dependency



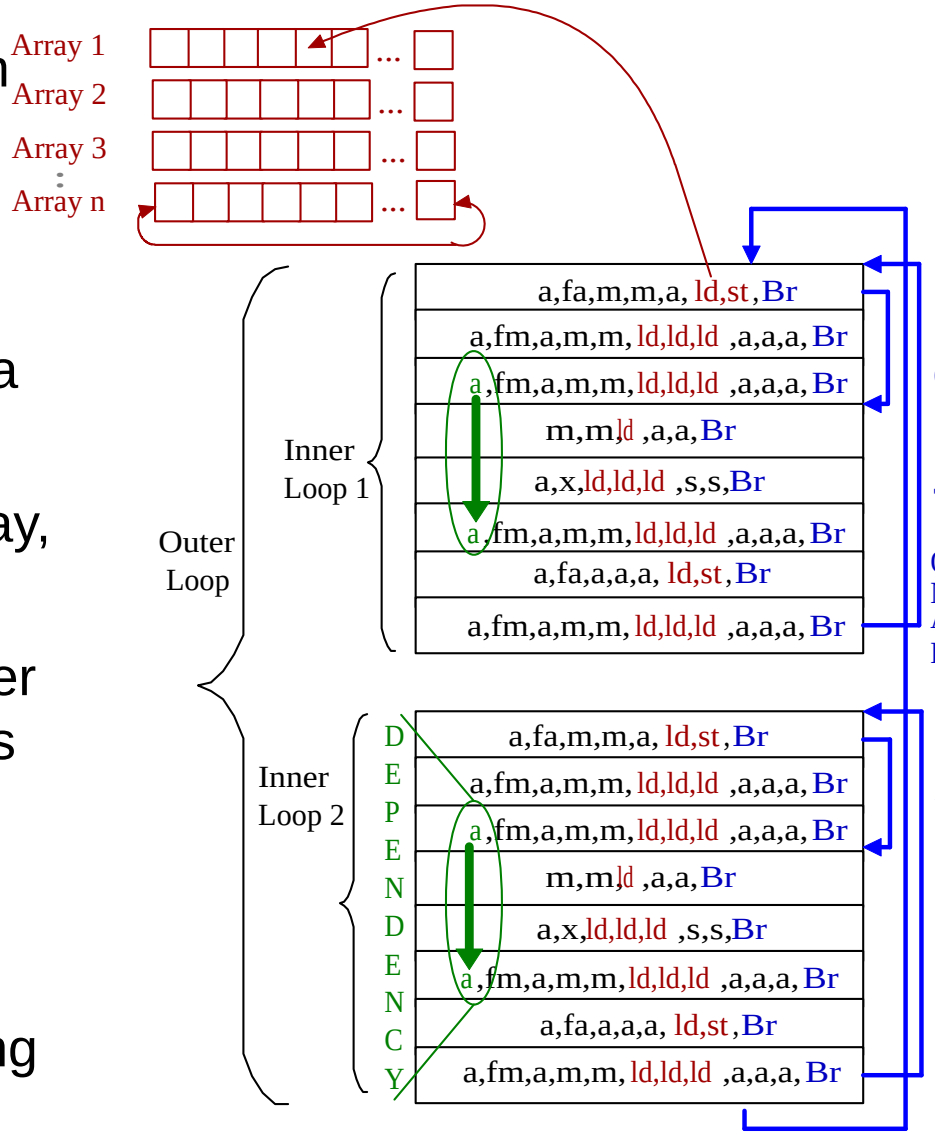
Code Generation

- **Step 5: Assign registers**
  - Destination registers – RoundRobin
  - Source registers – based on dependency
- **Step 6: Memory access model**
  - Ld/st access a set of 1-D arrays in a strided fashion
  - Ld/St - group into pools, assign array, 1 address calc instruction
  - Pointers - top of array at end of inner loop when required data foot print is touched

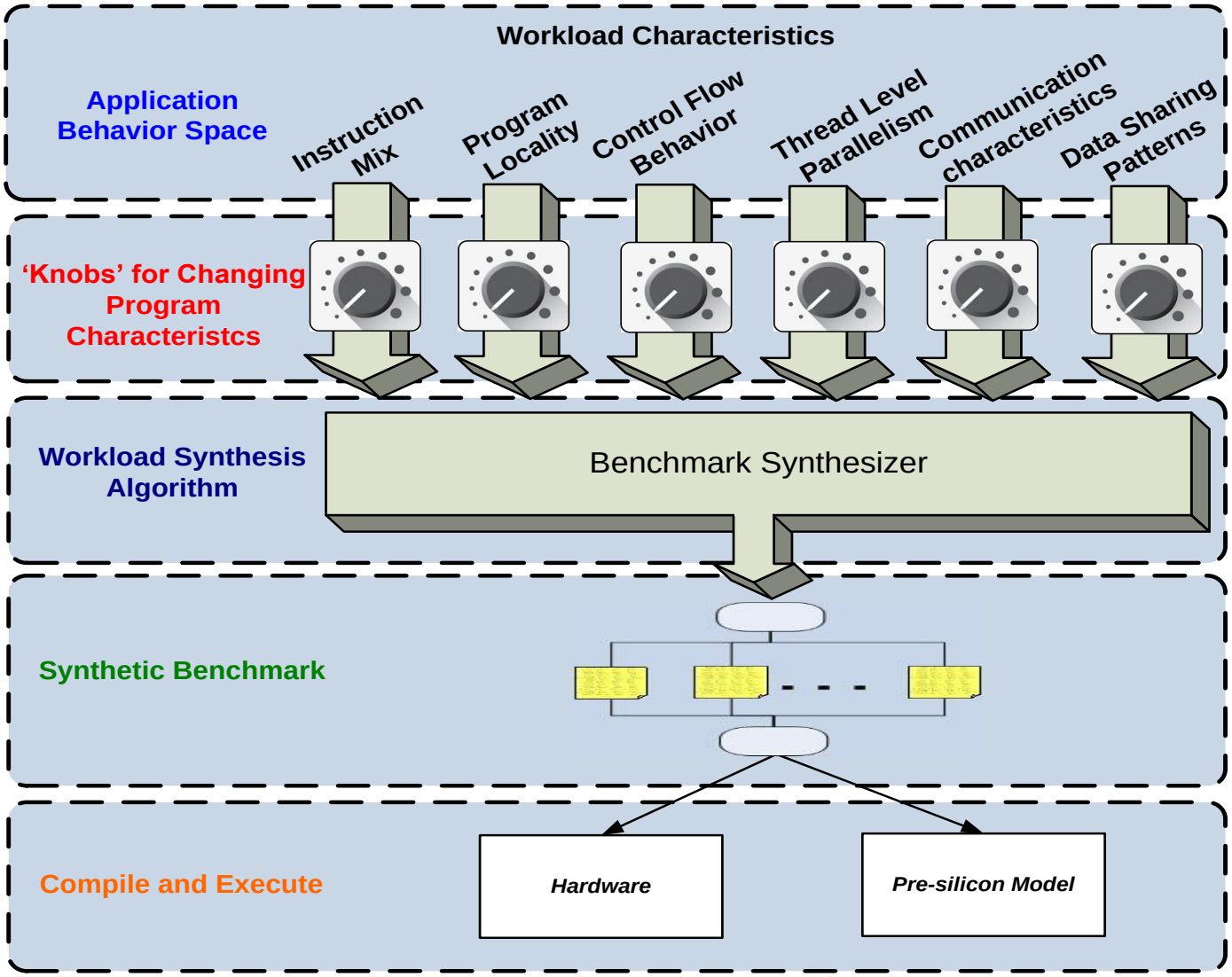


Code Generation

- **Step 5: Assign registers**
  - Destination registers – RoundRobin
  - Source registers – based on dependency
- **Step 6: Memory access model**
  - Ld/st access a set of 1-D arrays in a strided fashion
  - Ld/St - group into pools, assign array, 1 address calc instruction
  - Pointers - top of array at end of inner loop when required data foot print is touched
- **Step 7: MLP model**
  - Load-Load dependencies
  - For very infrequent highly bursty long latency loads, use 2 loops

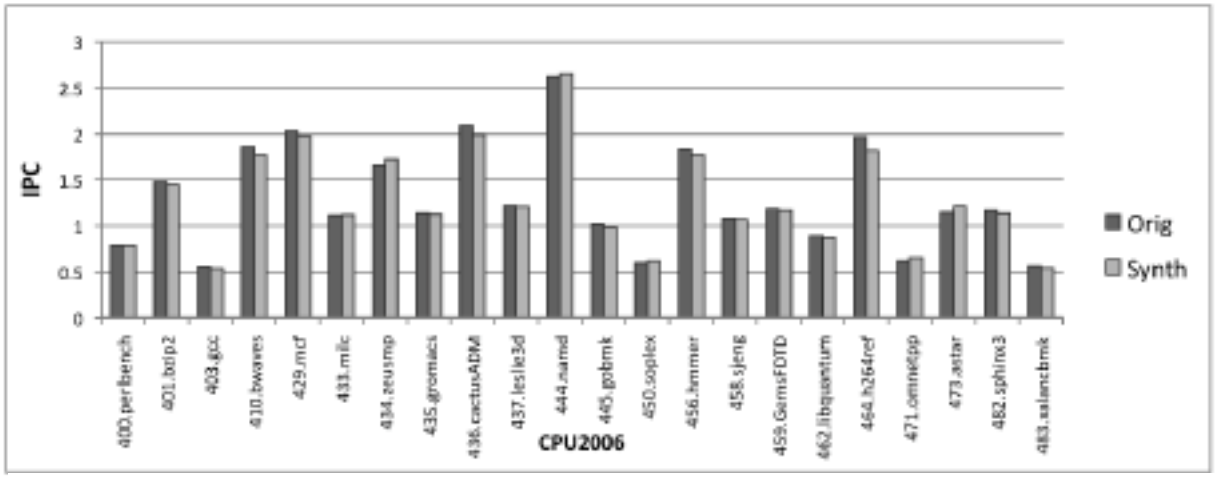


Adaptable Scalable Futuristic Benchmark Proxies

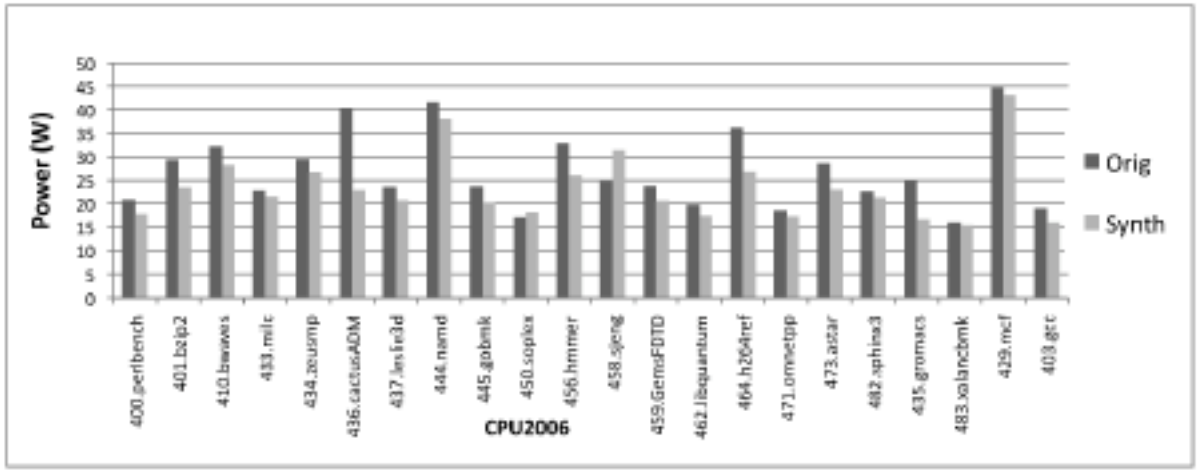


- Generate Clones by setting knobs to appropriate values
- Adaptable
- Scalable
- Futuristic

Validation of the Search Space



Average error = 2.8 %



Average error = 14 %

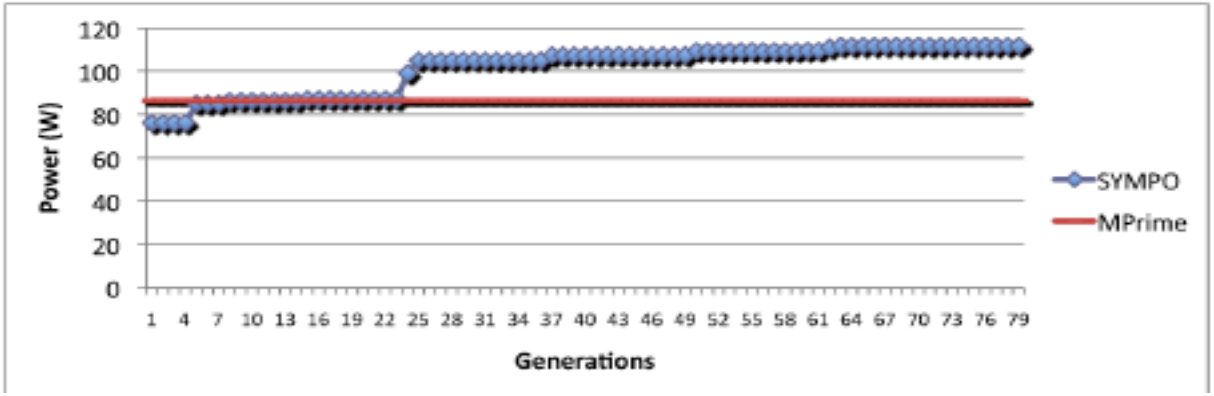
Validation of SYMPO on Alpha ISA

- **Comparison for three different machine configurations using Wattch for the most aggressive clock gating with**
  - Mprime popularly called the torture test
  - Comparison with SPEC CPU2006
  - Previous stressmark approach by Joshi et al [HPCA '08]

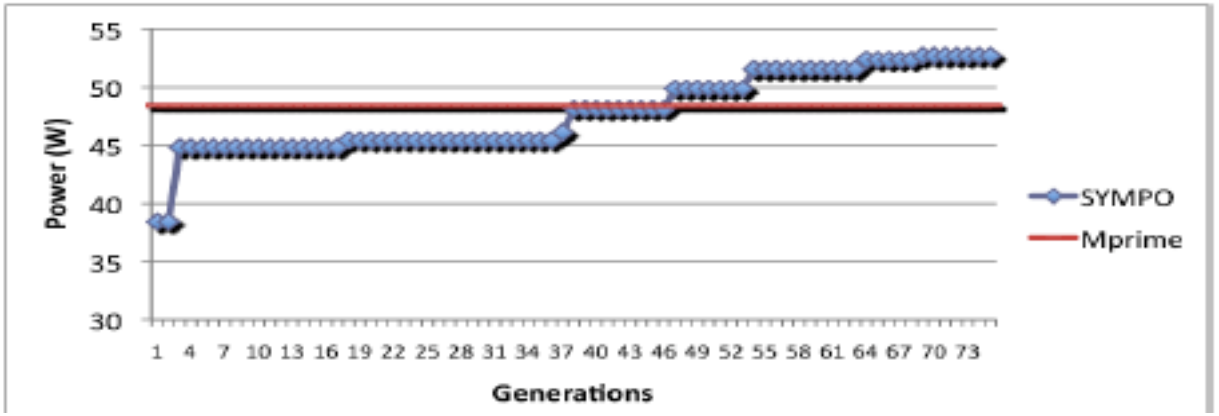
	Config 1	Config 2	Config 3
ALU	8 Int, 4 FP	4 Int, 2 FP	2 Int, 1 FP
L1 Cache	64 KB, 4-way	32 KB, 4-way	16 KB, 2-way
L2 Cache	4 MB, 8-way	4 MB, 8-way	256 KB, 4-way
ROB / LSQ	256/128	128/64	16/8
Machine Width	8	4	2
Branch predictor	Hybrid 4 KB	Hybrid 4 KB	2-level
Mem access time	150 cycles	150 cycles	40 cycles

SYMPO Vs Mprime on Alpha ISA

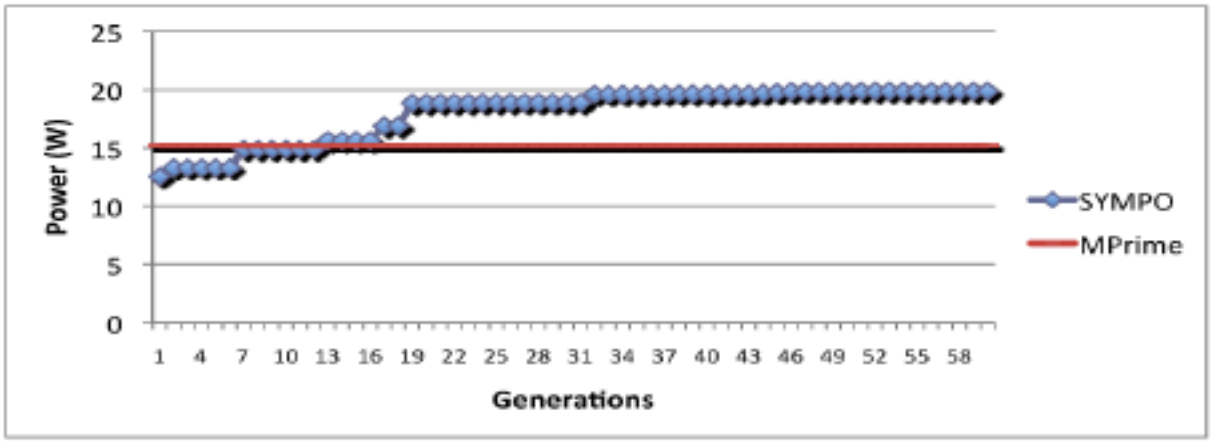
Config 1 - 30% more than MPrime, 15% more than Joshi et al.'s virus



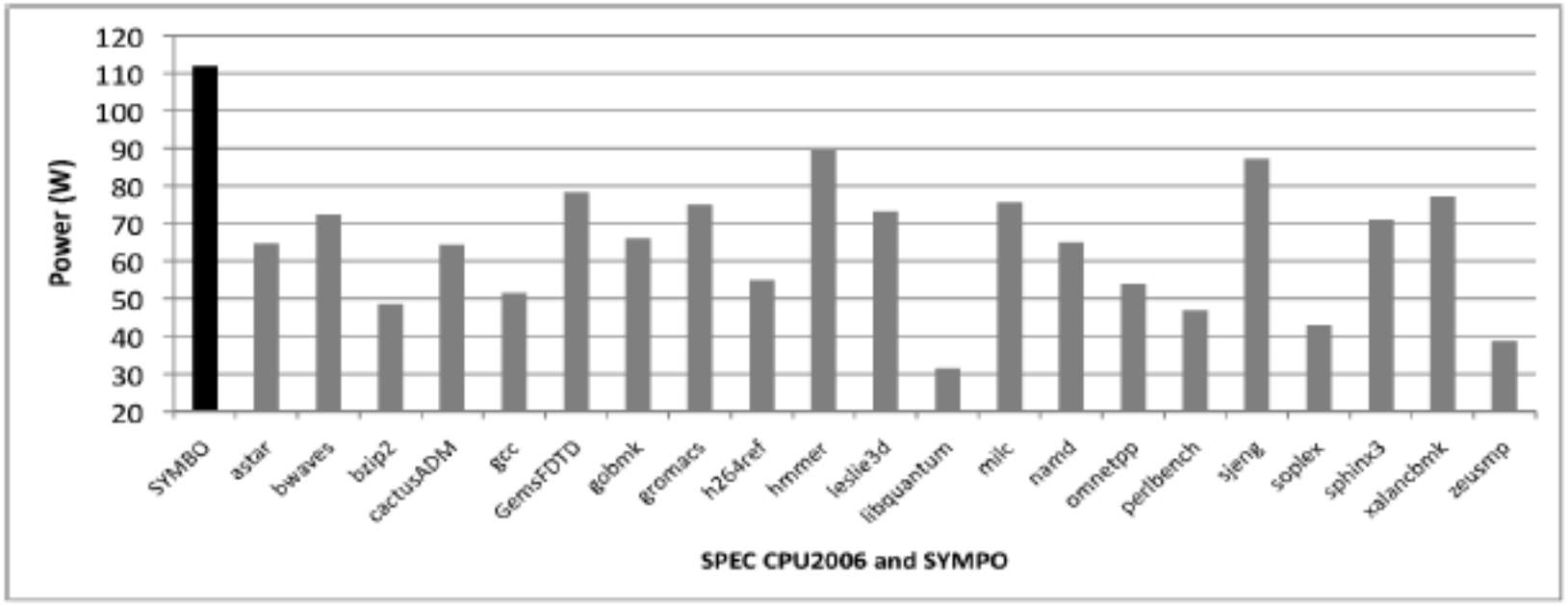
Config 2 - 8% more than MPrime, 9% more than Joshi et al.'s virus



Config 3 - 29% more than MPrime, 24% more than Joshi et al.'s virus



Comparison to SPEC CPU2006 on Alpha ISA



- **Comparison to SPEC CPU2006 on config3: 89.2 Watts compared to 111.79 Watts, where theoretical maximum is 220 Watts**

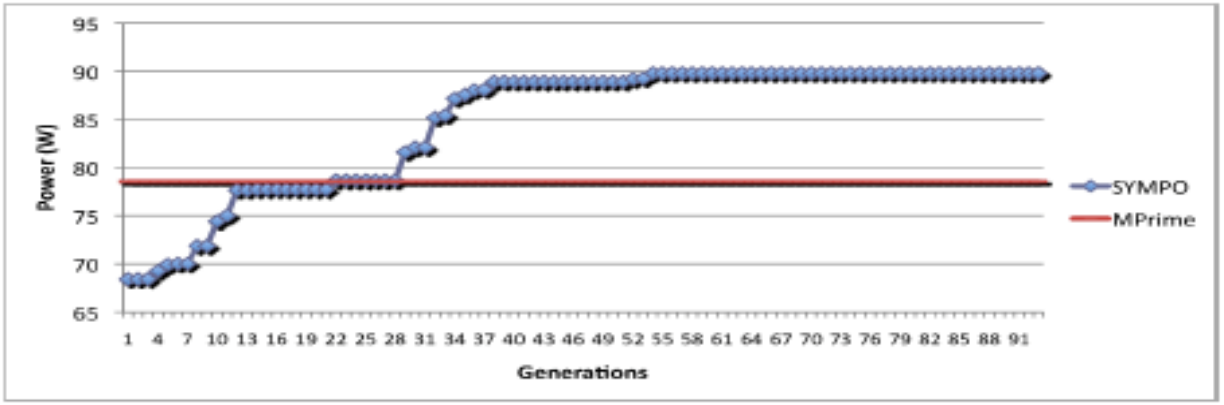
Validation of SYMPO on SPARC ISA

- **Comparison with**
  - Mprime popularly called the torture test
  - Comparison with SPEC CPU2006
  - For three different machine configurations
- **Virtutech Simics full system simulator with GEMS**
  - Detailed out-of-order processor model Opal with power models from Wattch for the most aggressive clock gating
  - Detailed memory simulator Ruby and DRAMsim for DRAM

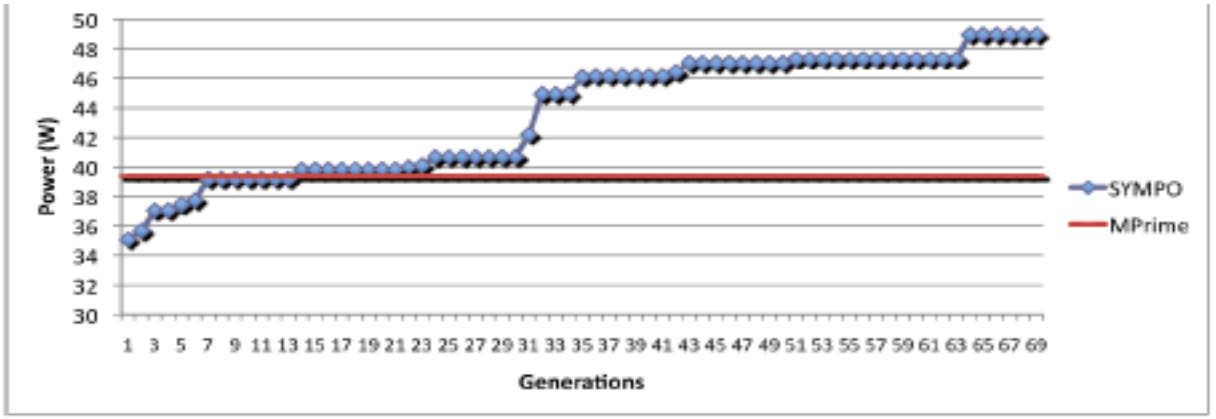
	Config 1	Config 2	Config 3
ALU	4 Int, 4 FP	3 Int, 2 FP	1 Int, 1 FP
L1 I- & D-Cache	64 KB, 256 MSHR	32 KB, 128 MSHR	16 KB, 64 MSHR
L1 Cache Latency	3 cycles	2 cycles	1 cycles
L2 Cache	4 MB, 128 MSHR	2 MB, 64 MSHR	512 KB, 32 MSHR
DRAM	8 GB	4 GB	2 GB
ROB	256 entries	128 entries	32 entries
Machine Width	8	4	2

SYMPO Vs Mprime on SPARC ISA

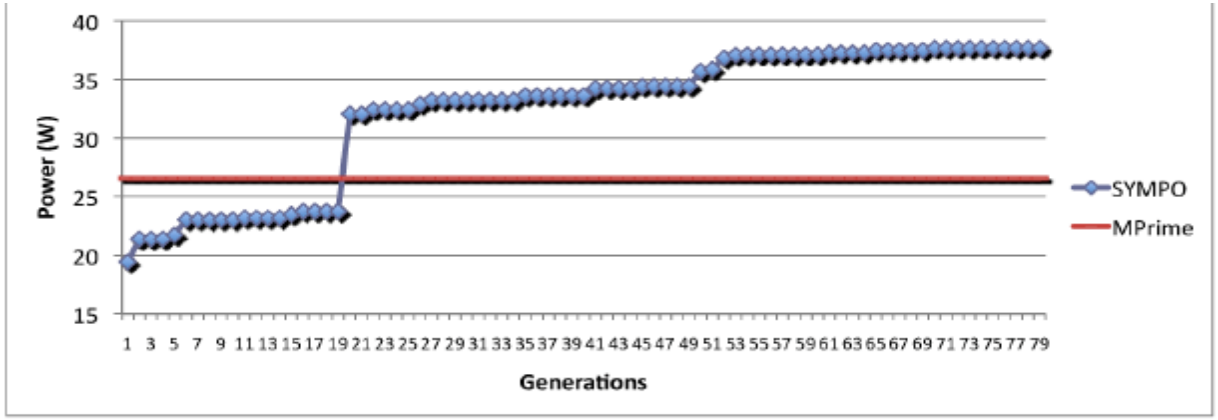
Config 1 - 14% more



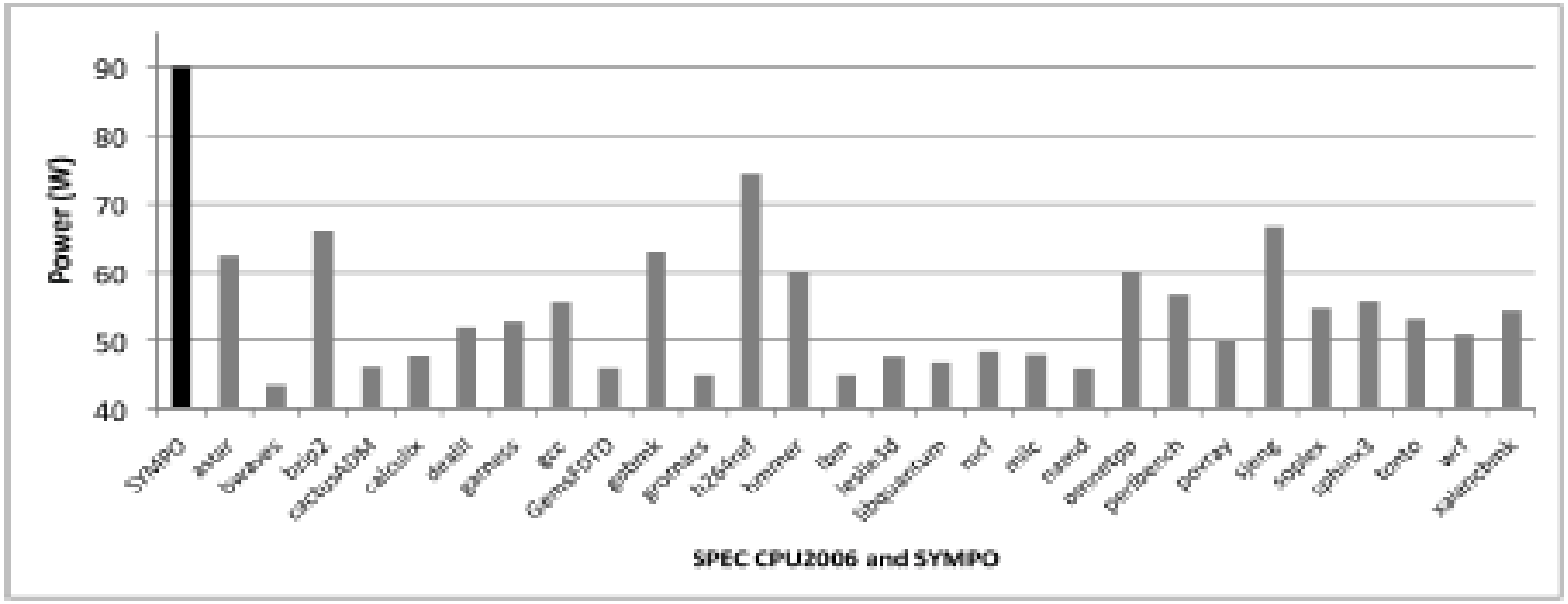
Config 2 - 24% more



Config 3 - 41% more

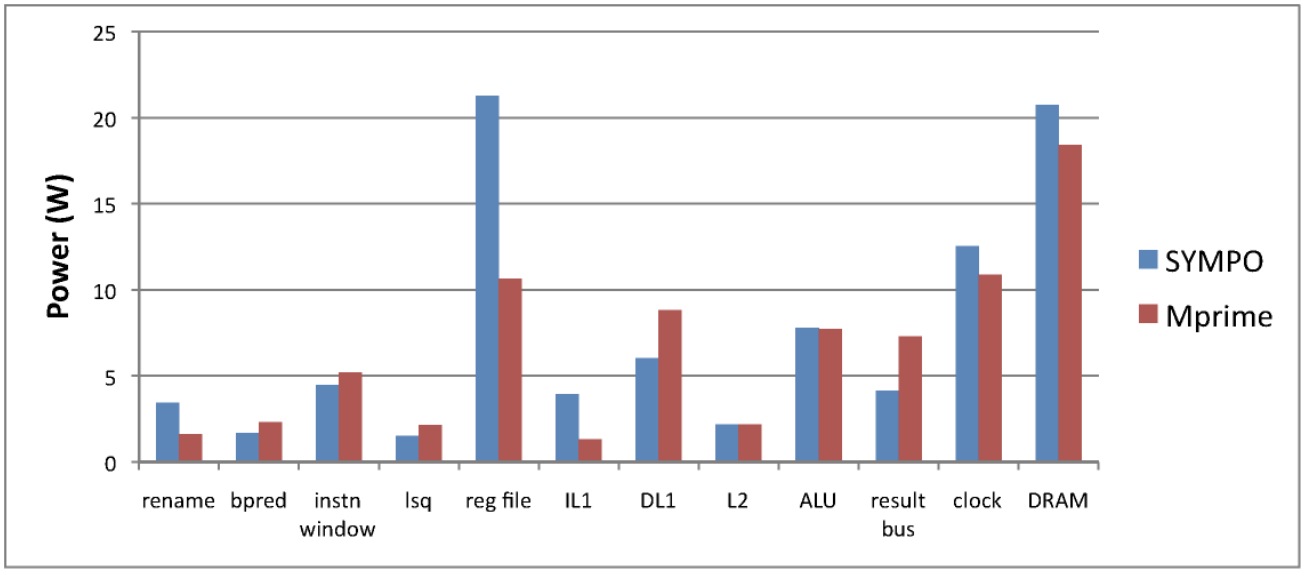


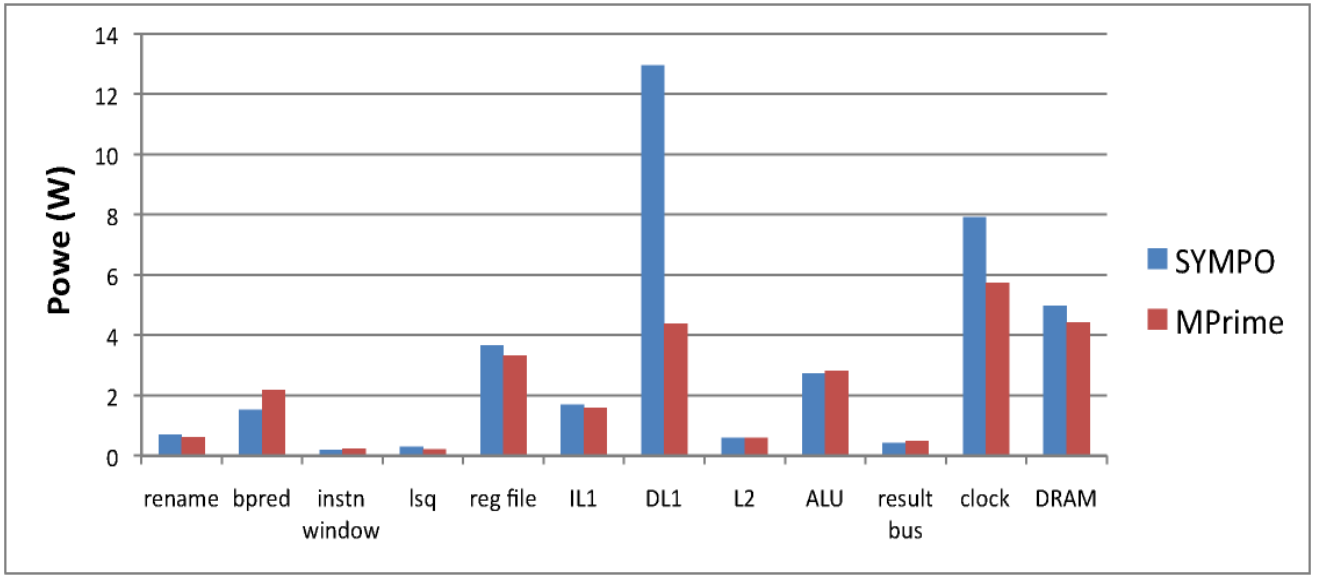
Comparison to SPEC CPU2006 on SPARC ISA



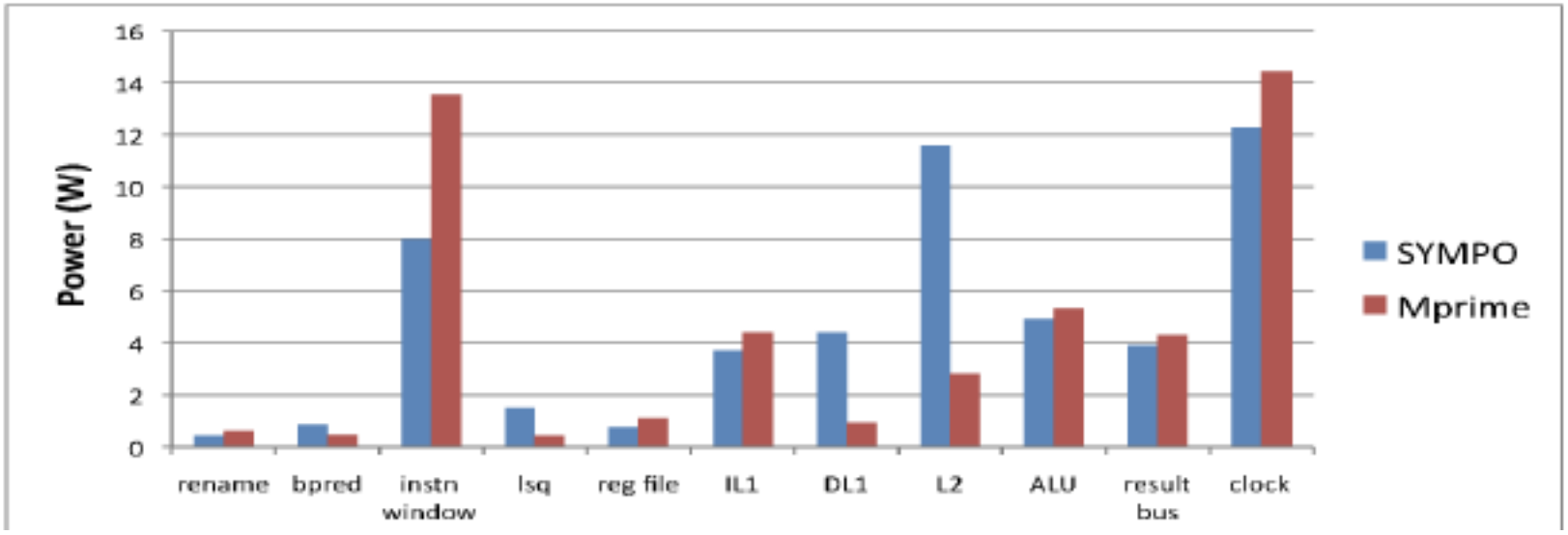
- **Comparison to SPEC CPU2006: 74.4Watts compared to 89.8Watts**

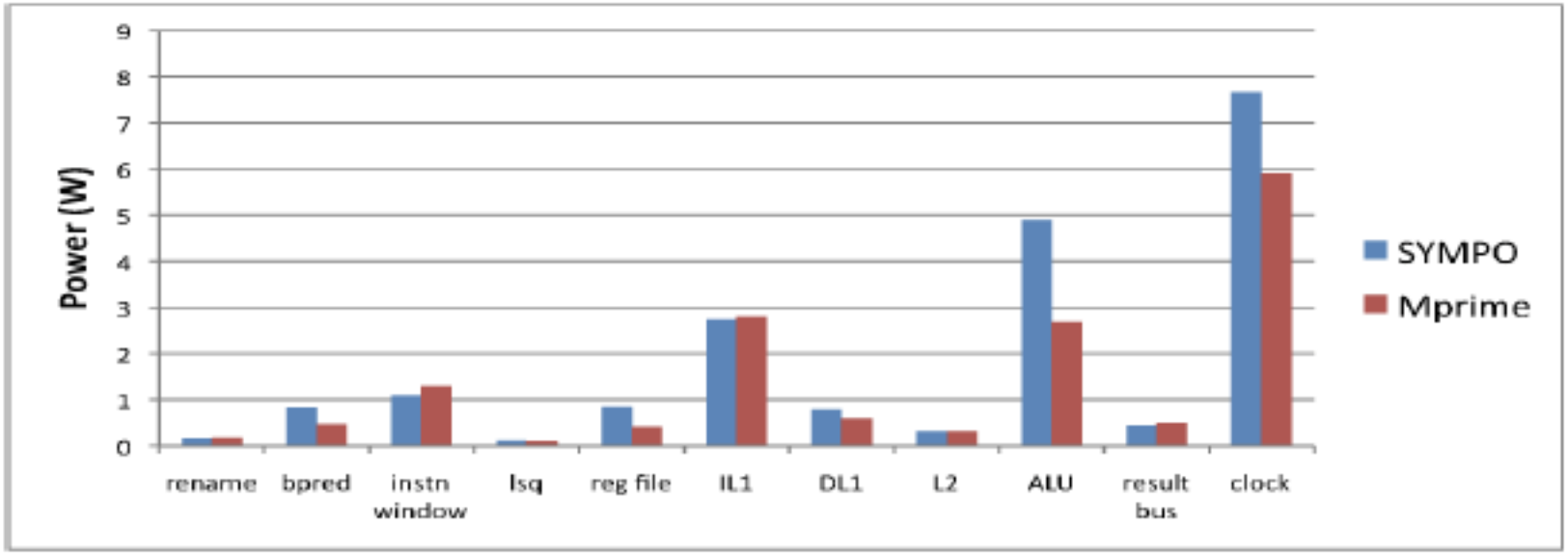
Uniqueness of Viruses – SPARC Config. 1 and 3



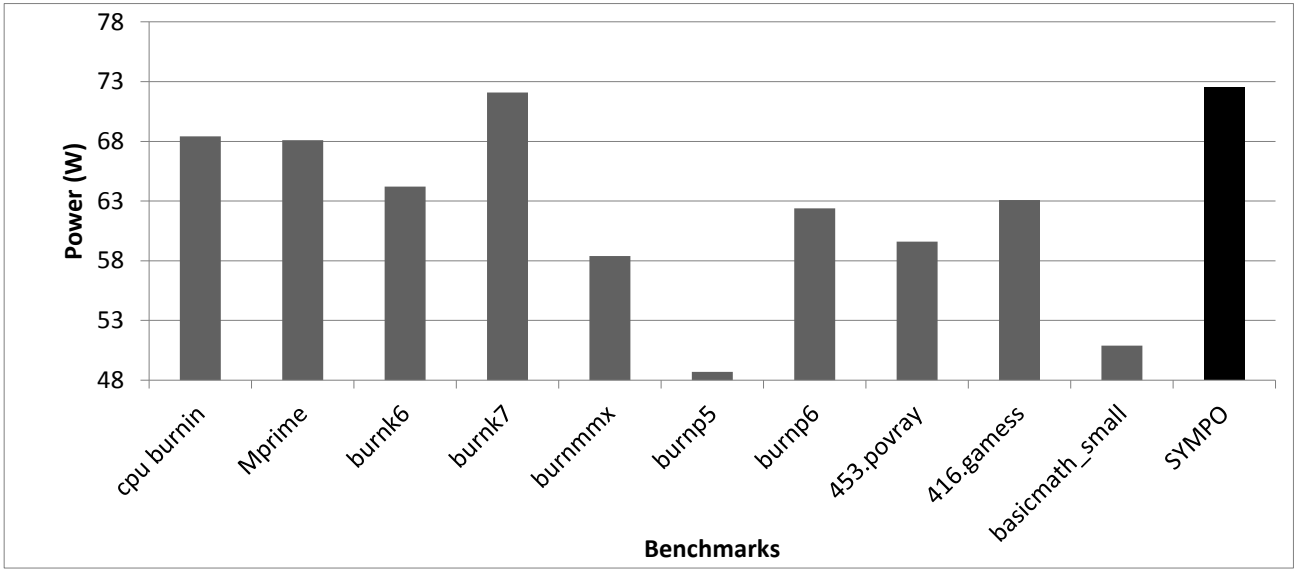


Uniqueness of Viruses – Alpha Config. 2 and 3





Validation on Real Hardware



- **Our code generator was not equipped to generate code using CISC ISAs**
  - Microarchitecturally equivalent system for the instrumented AMD Phenom II system on GEMS
  - Generated power viruses on SPARC ISA and translated to x86 using LLVM infrastructure

Summary

- **Proposed the usage of SYMPO, a framework to automatically generate system level max-power viruses for a given machine configuration**
- **Validated SYMPO on SPARC, Alpha and x86 ISAs by comparing with Mprime and CPU2006 on full system simulator and real hardware**

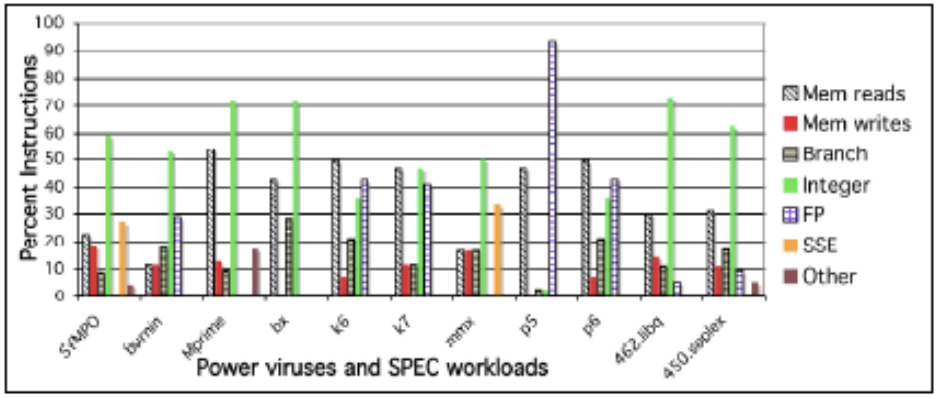
	SPARC Config 1	SPARC Config 2	SPARC Config 3	Alpha Config 1	Alpha Config 2	Alpha Config 3	X86 hardware
SYMPO	89.8 W	48.95 W	37.68 W	19.86 W	52.70 W	111.8 W	72.5 W
MPrime	78.56 W	39.36 W	26.58 W	15.20 W	48.41 W	86.58 W	68.1 W
% increase	14.3 %	24.36 %	41.78 %	30.6 %	8.80%	29.1 %	6.46 %

Thank You!!
Questions?

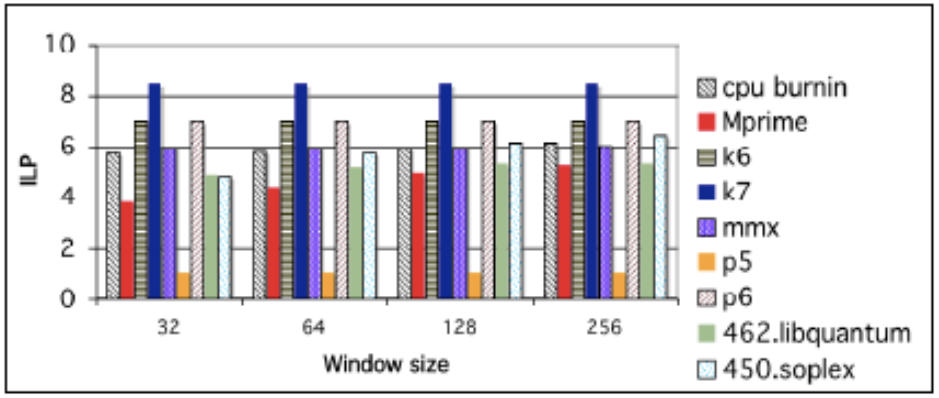
Laboratory for Computer Architecture
University of Texas at Austin

Back up Slides

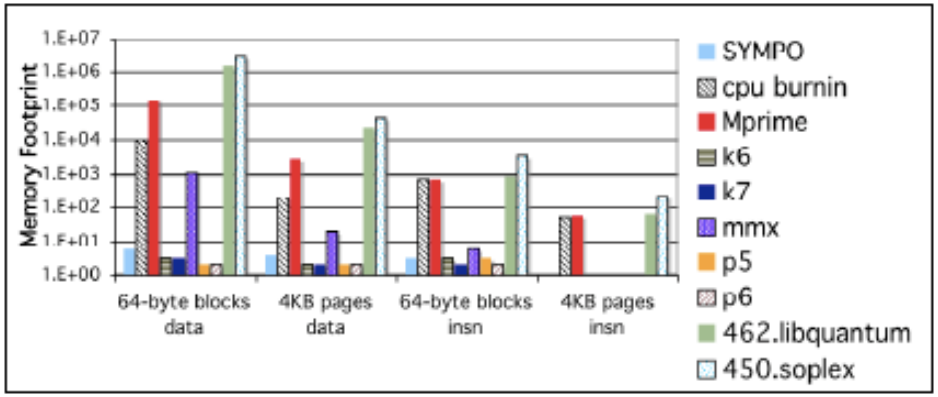
Characterization of Other Industry-grade Power Viruses



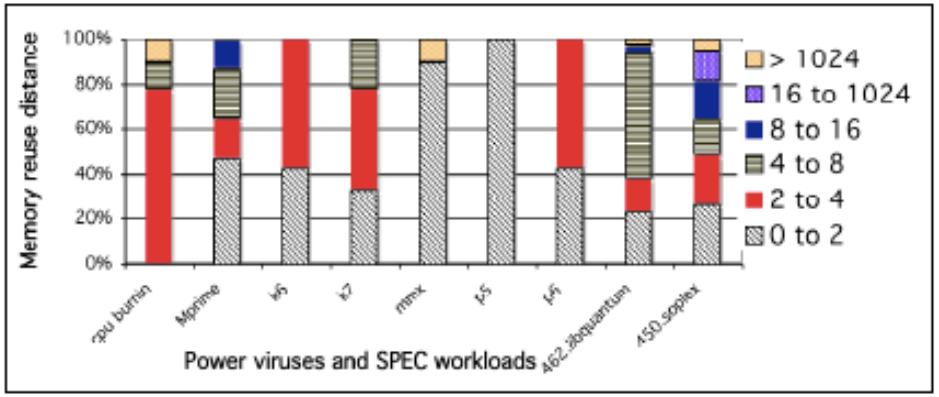
(a) Instruction mix



(b) Instruction level parallelism

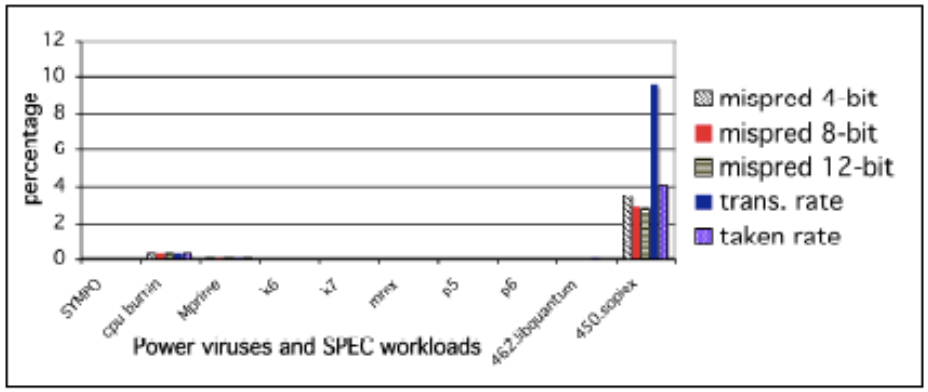


(c) Memory footprint

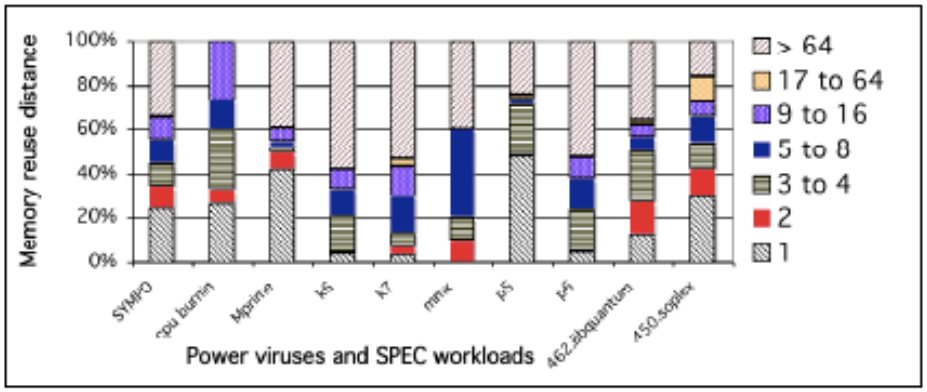


(d) Memory reuse distribution

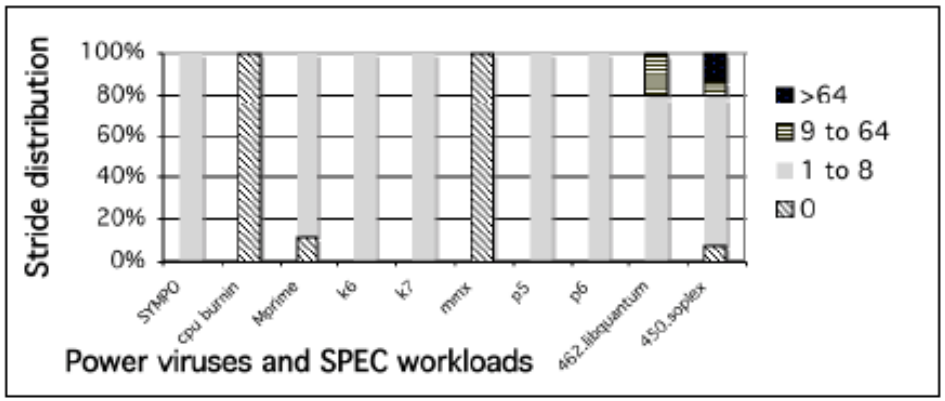
Characterization of Other Industry-grade Power Viruses



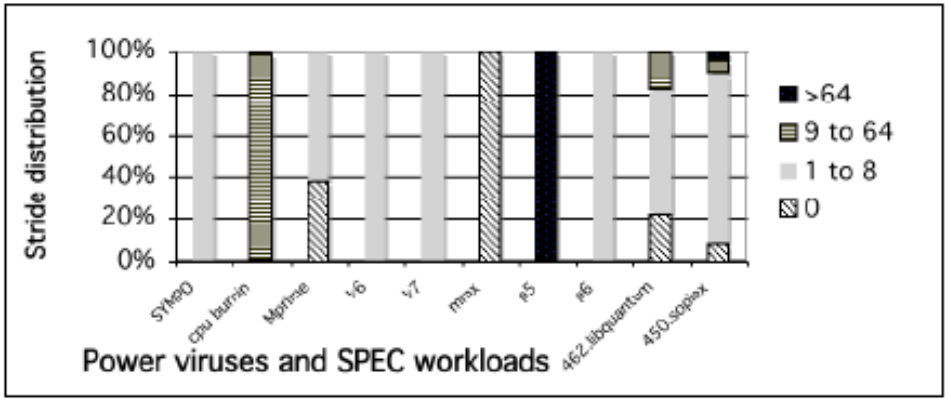
(a) Branch predictability



(b) Register reuse distance distribution



(c) Stride distribution of memory reads



(d) Stride distribution of memory writes