A Many Threaded CUDA Interpreter for genetic programming

W. B. Langdon

CREST lab,

Department of Computer Science

KING'S College LONDON

University of London

Slides presented at EuroGP 2010, LNCS 6021, p146-158

Introduction

- Running tree GP on graphics hardware
- How
- 8692 times faster than PC without GPU
- Solved 20 input Boolean multiplexor problem
- Solved 37 input Boolean multiplexor problem (all 137 10^9 tests)

Threat: No More Moore's Law

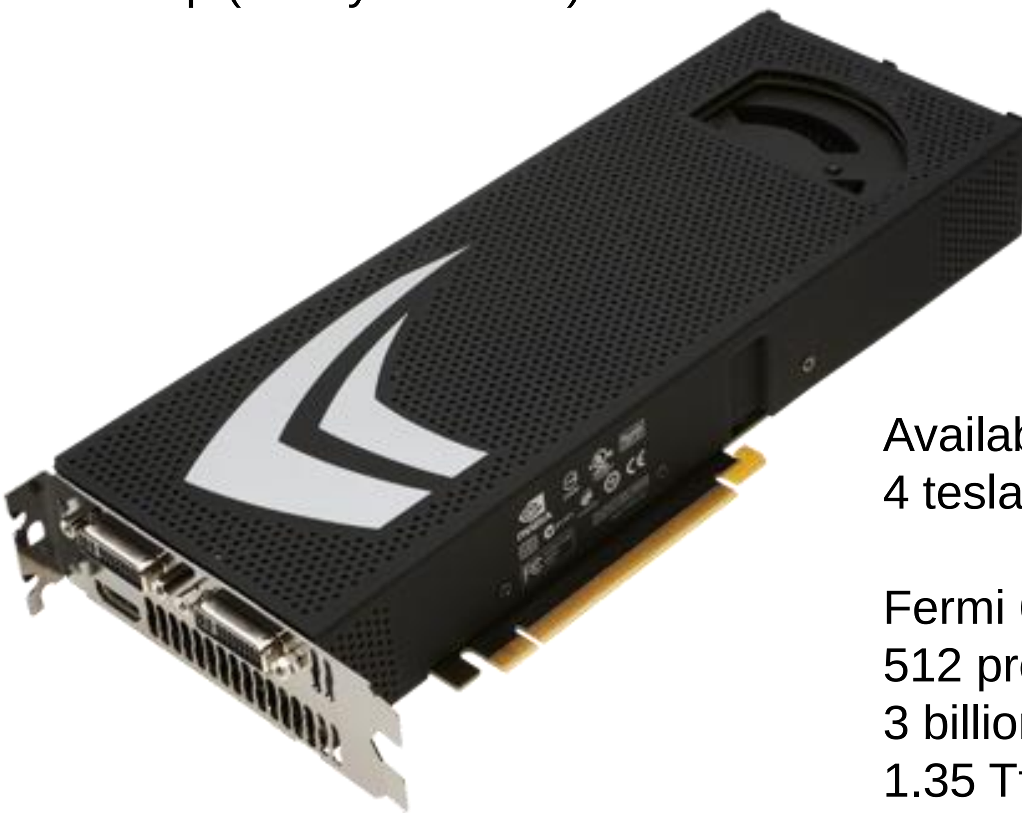
- CPUs no longer double in speed
- BUT number of transistors is still doubling
  - More complicated CPU
  - Parallel
- Today a single graphics card can contain hundreds of fully functioning CPUs running in parallel

Benefit: Moore's Law applies to number of transistors

2 240 Stream Processors

Clock 1.24 GHz ¾ Tflop (nbody estimate)

1992 MByte

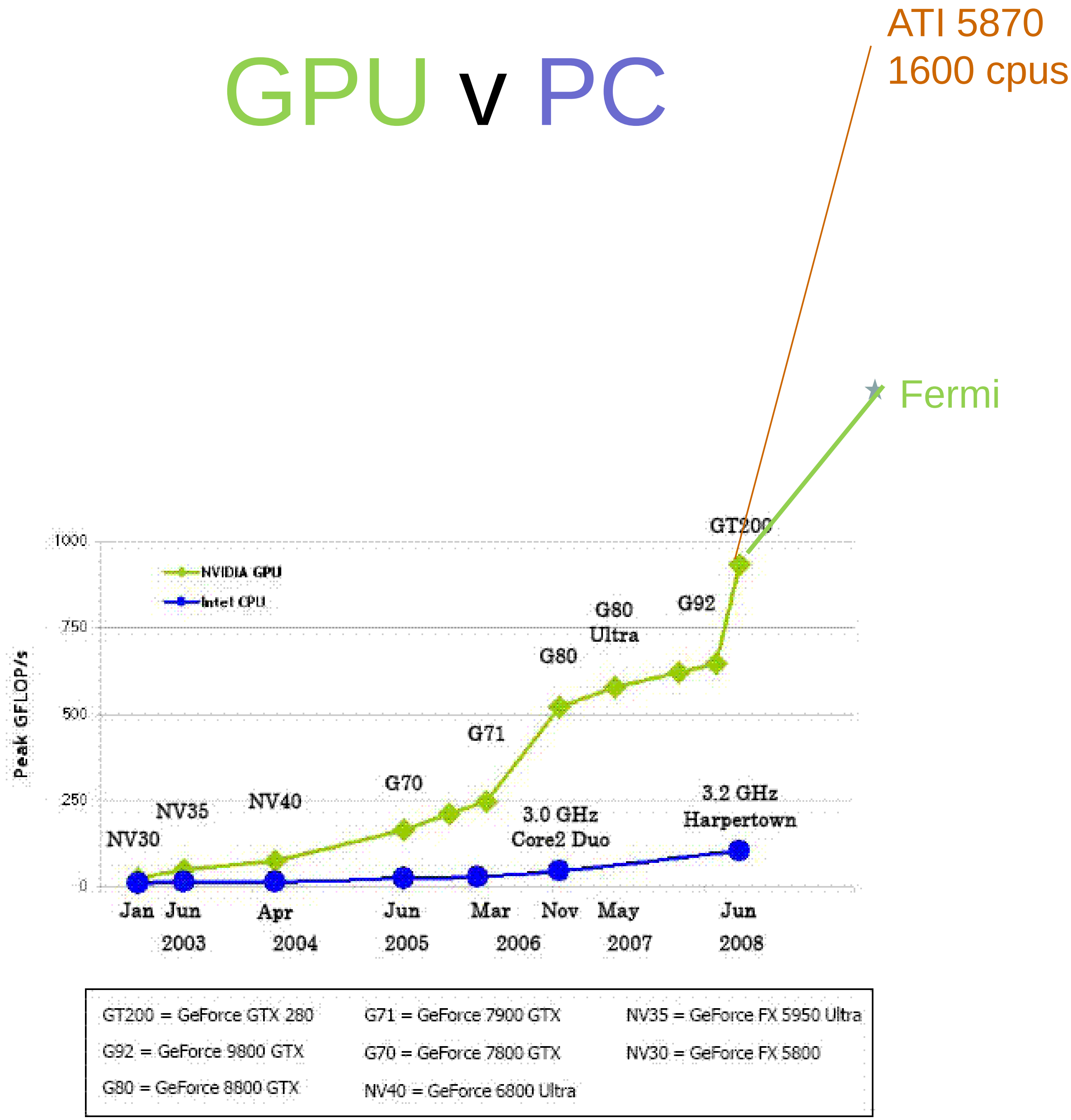


Available 1.5GHz
4 tesla up to 16GBytes

Fermi 64 bit (March 26)
512 processors
3 billion transistors
1.35 Tflops (manufacture)

nVidia GeForce 295 GTX 10½   4⅜ inches

GPU v PC


ATI 5870
1600 cpus
Fermi
GT200
G92
G80
Ultra
G80
G71
G70
NV40
NV35
NV30
3.0 GHz
Core2 Duo
3.2 GHz
Harpertown
NVIDIA GPU
Intel CPU
Peak GFLOP/s
1000
750
500
250
0
Jan
Jun
2003
Apr
2004
Jun
2005
Mar
2006
Nov
May
2007
Jun
2008
GT200 = GeForce GTX 280
G92 = GeForce 9800 GTX
G80 = GeForce 8800 GTX
G71 = GeForce 7900 GTX
G70 = GeForce 7800 GTX
NV40 = GeForce 6800 Ultra
NV35 = GeForce FX 5950 Ultra
NV30 = GeForce FX 5800

Speed up

- Speed comes from combining and improving four GP techniques:
  - Graphics hardware
  - Sub machine code GP (use all 32 bits)
  - Random sampling of fitness cases
  - Reverse Polish Notation CUDA interpreter

Graphics hardware	480	
Sub machine code GP	32	
Sampling fitness cases	512	(20 mux)
	16,777,216	(37 mux)
RPN CUDA interpreter	1	

Sub Machine Code GP

- Graphics cards supports many data types
  - RapidMind 2 only used float
- Pack 32 Boolean bits into one integer
  - AND int does 32 Boolean logic in one go
- Each thread does 32 fitness cases
  - All tests for D_0 D_1 D_2 D_3 D_4 in one go
- Correct bit mask = ~(answer XOR target)
  - Fitness = count correct bits
  - Seibert’s fast bit count (3 lines v loop 32)

Sampling Fitness Cases 1

- Too many training cases to use all.
  - So train on randomly selected sample
- When a GP individual passes all 8192 tests in the random sample, then check all 137 10^9 tests.
- Use whole GPU to test one program
  - Can stop first time any test fails
  - If fail abort other tests running in parallel

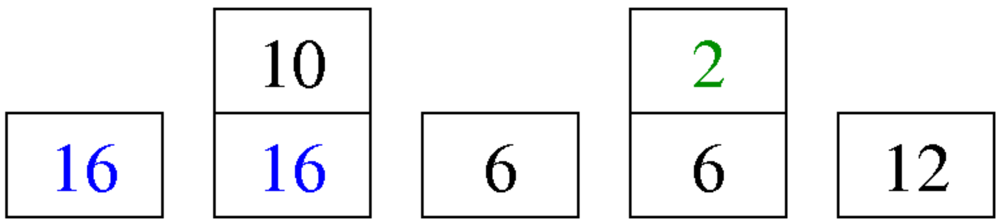
Sampling Fitness Cases 2

- Using submachine code GP so can test all 32 lower 5 bits patterns. Sample top 32bits
- For each random pattern invert top 32bits to also test its complement.
- Sample needs 8192/32/2=128 pseudo random numbers
- Reduce noise by using same random sample for all 4 members of tournament
- Each generation and each tournament has different sample

Reverse Polish Tree Interpreter

$(A-10)*B$ $A=16$ $B=2$

(MUL (SUB A 10) B) A 10 − B *



(Mul (Sub A 10) B) ≡ A 10 - B

Variable (terminal): push onto stack

Function pop arguments, do operation, push result

1 stack per program. All stacks in shared memory.

PC moves linearly from start→end expression

Representing the Population

- Same structure on host as GPU.
  - Avoid explicit format conversion when population is loaded onto GPU.
- Genetic operations act on Reverse Polish:
  - random tree generation (eg ramped-half-and-half)
  - subtree crossover
  - 2 types of mutation
- Requires only one byte per leaf or function.
  - So large populations (millions of individuals) are possible.
- Like GPquick (but GPquick uses linearised *prefix*)
- nVidia CUDA kernel replaces RapidMind

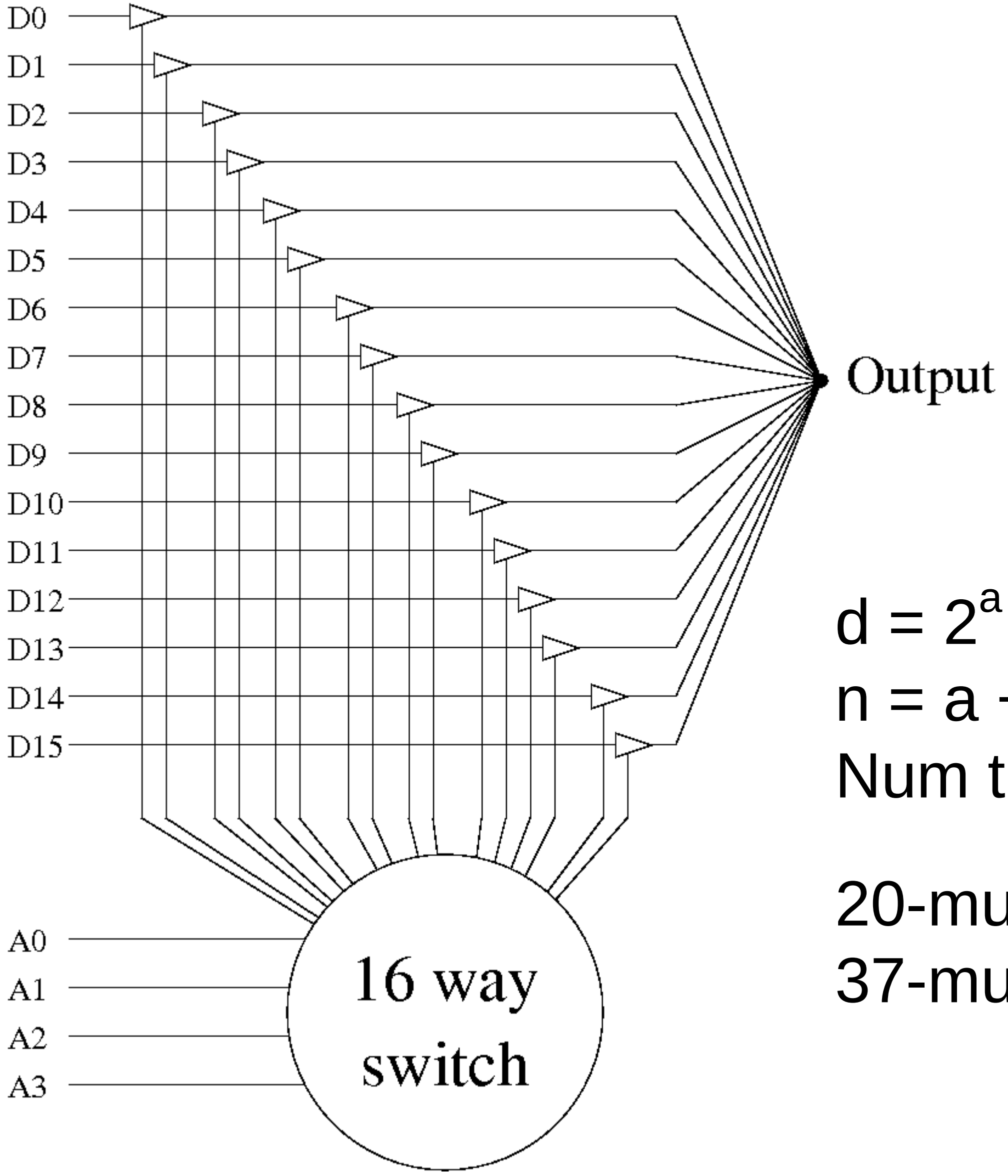
CUDA Interpreter: Summary

- Put stack in fast shared memory
- Randomised testing
- Choice between sequential and parallel
- Use 1↔256 threads for one test
  - – reduce by parallel sum into one fitness value.
  - – Siebert's bit count (replaces 32 loops)
- 1 Program in fast read-only global memory
- Interprets 261 10^9 GP primitives per sec.
- (670 billion per second sustained peak)

Experiments

- 20 multiplexor solved
  - Full test 2^{20} = 1,048,576
  - sample size = 2048
- 37 multiplexor solved
  - Full test 2^{37} =137 billion test cases
  - sample size = 8192

Boolean Multiplexor



$d = 2^a$

$n = a + d$

Num test cases = 2^n

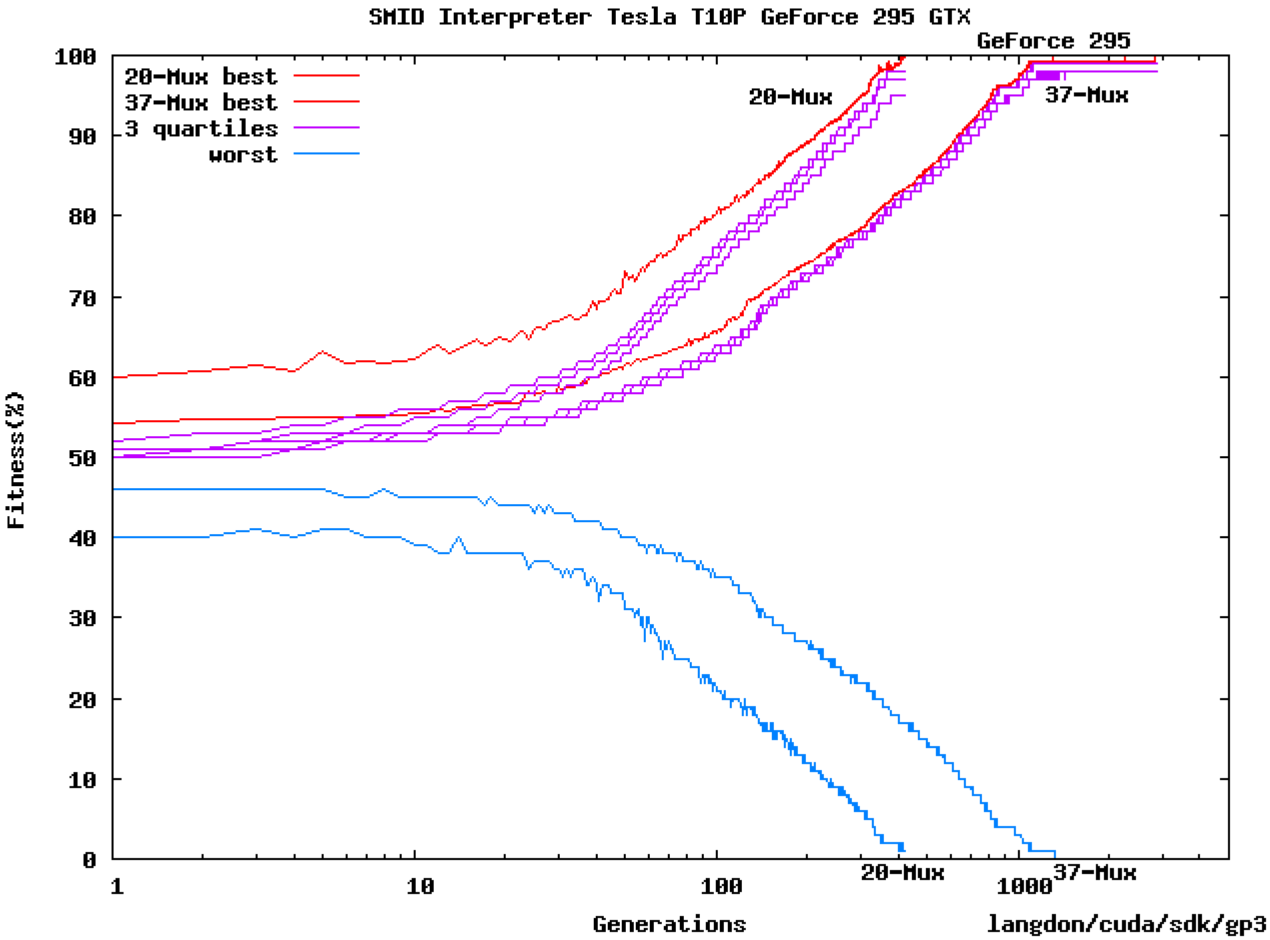
20-mux 1 million test cases

37-mux 137 10^9 tests

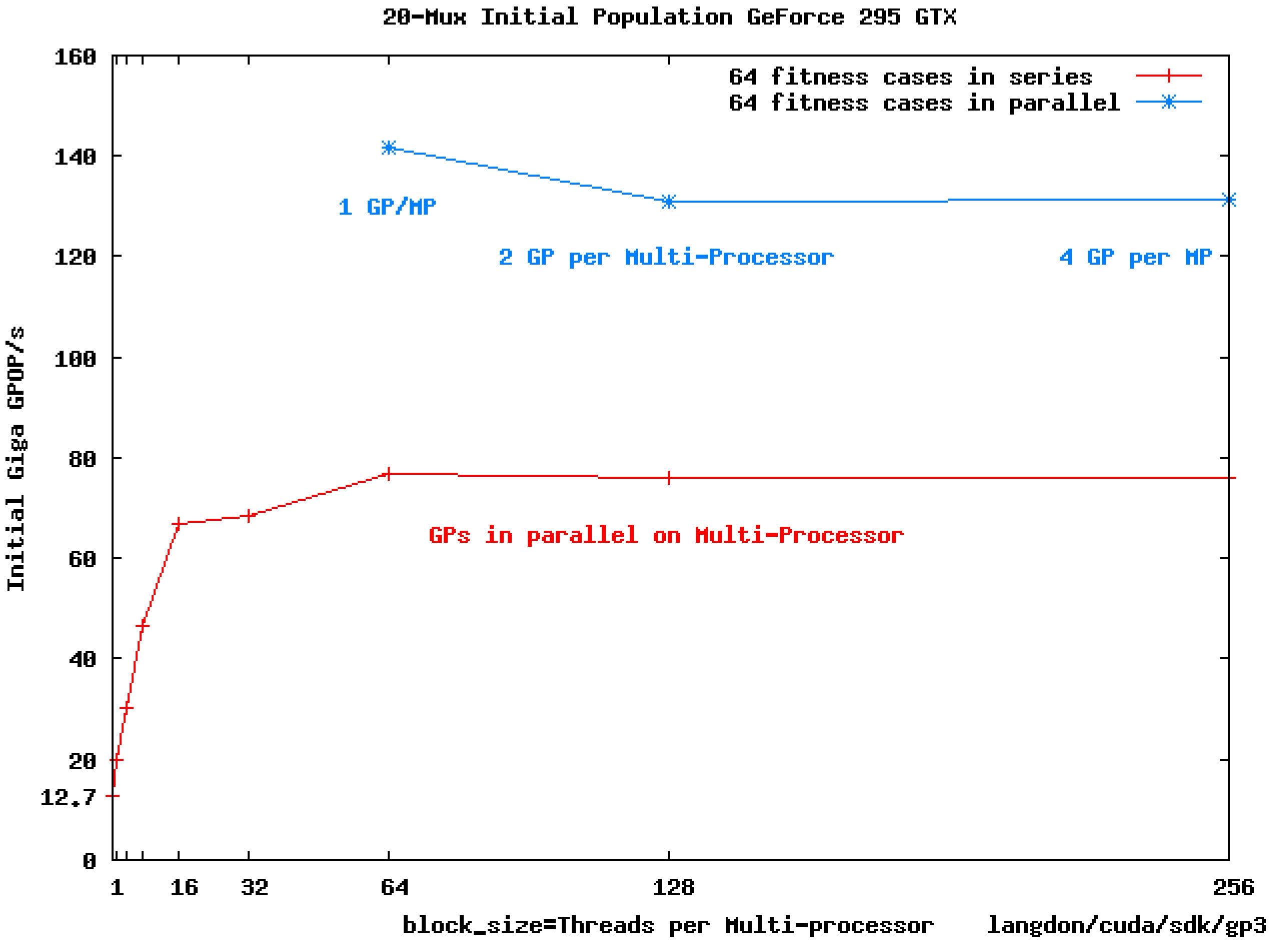
20-Mux 37-Mux

- Function set: AND OR NAND NOR
- Terminal set: $D_0..D_{37}$ (D_0-D_5 packed into int)
- Fitness: tests past
- Population: ¼ million binary trees
- Parameters:
  - Ramped ½-½, tournament size=4,
  - 50% crossover, 50% mix of mutation,
  - max depth 15, max size 1023.
- Up to 5000 generations

Evolution of 20-Mux and 37-Mux

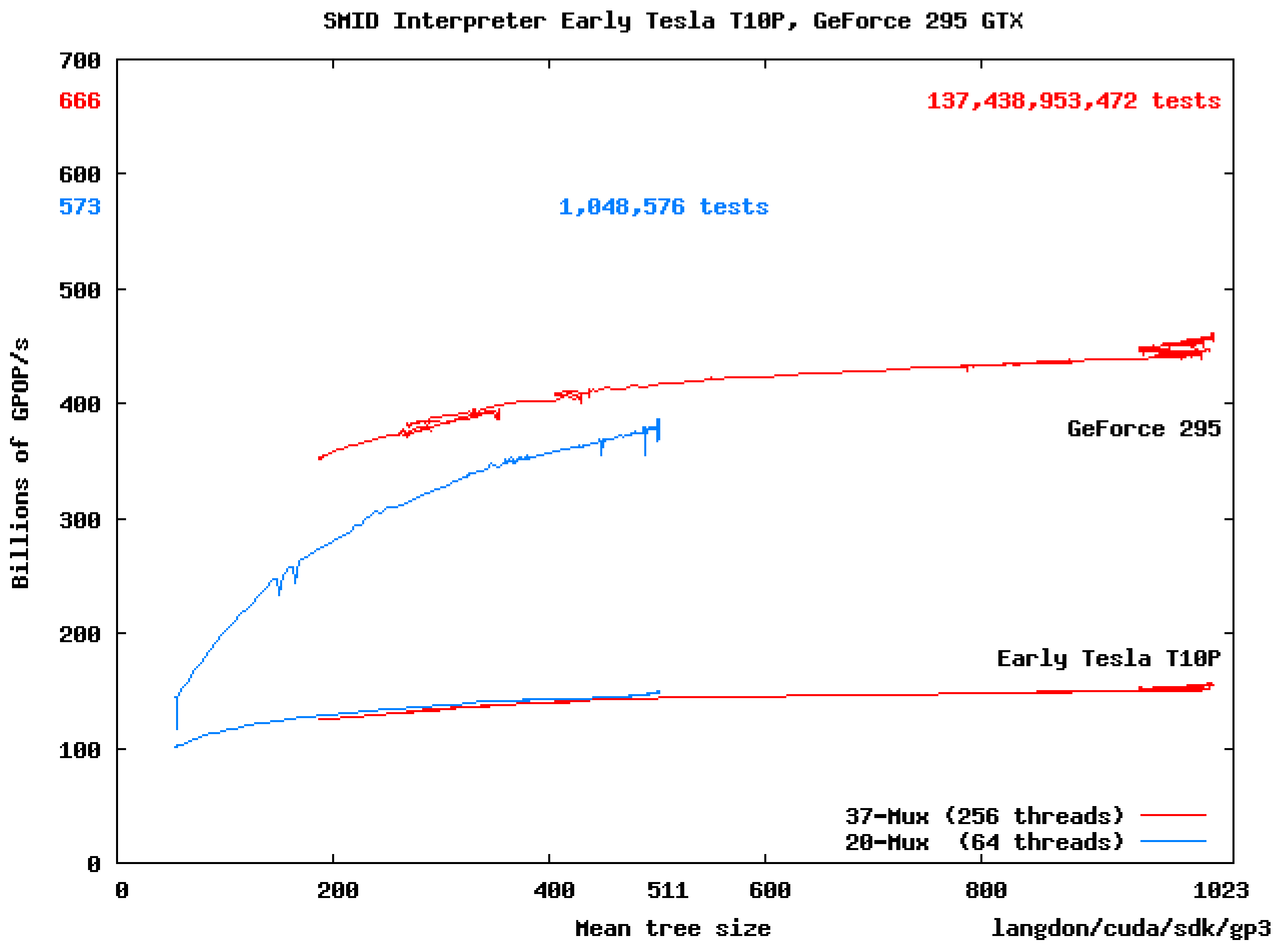


Performance v Test v Threads



CREST

Performance v Program size



GP Performance 295 GTX

- GPU 261 10^9 GP operations/second averaged across whole run.
  - GPU so fast fitness testing not dominating. PC host now also important (not optimised)
- Sustained peak 670 10^9 GP ops/sec
  - When validation single best program
  - One program fits in “constant” memory
  - 37-Mux speed up 476 $10^9 \rightarrow$ 670 10^9

Code via FTP
cs.ucl.ac.uk/genetic/gp-code/gp32cuda.tar.gz

Conclusions

- GP CUDA interpreter allows choices of
  - which aspects of fitness are done in parallel
  - explicit location of key data structures to get best from GPU hardware.
- Submachine code GP on graphics cards
- Randomise test case selection
  - Evolve on tiny (less than 10^{-6}th) fraction of whole. Then validates on all.
- Cheap - your own "cluster" performance
- FAST - 20-mux and 37-mux solved.