Reconfigurable Acceleration of Fitness Evaluation in Trading Strategies

INGRID FUNIE, PAUL GRIGORAS, PAVEL BUROVSKIY, WAYNE LUK, MARK SALMON

Department of Computing
Imperial College London

Published in Proceedings of ASAP Conference, 2015

Achievements Overview

A. **Novel pipelined architecture for trading strategies:**
   - evaluate fitness function of complete expression trees

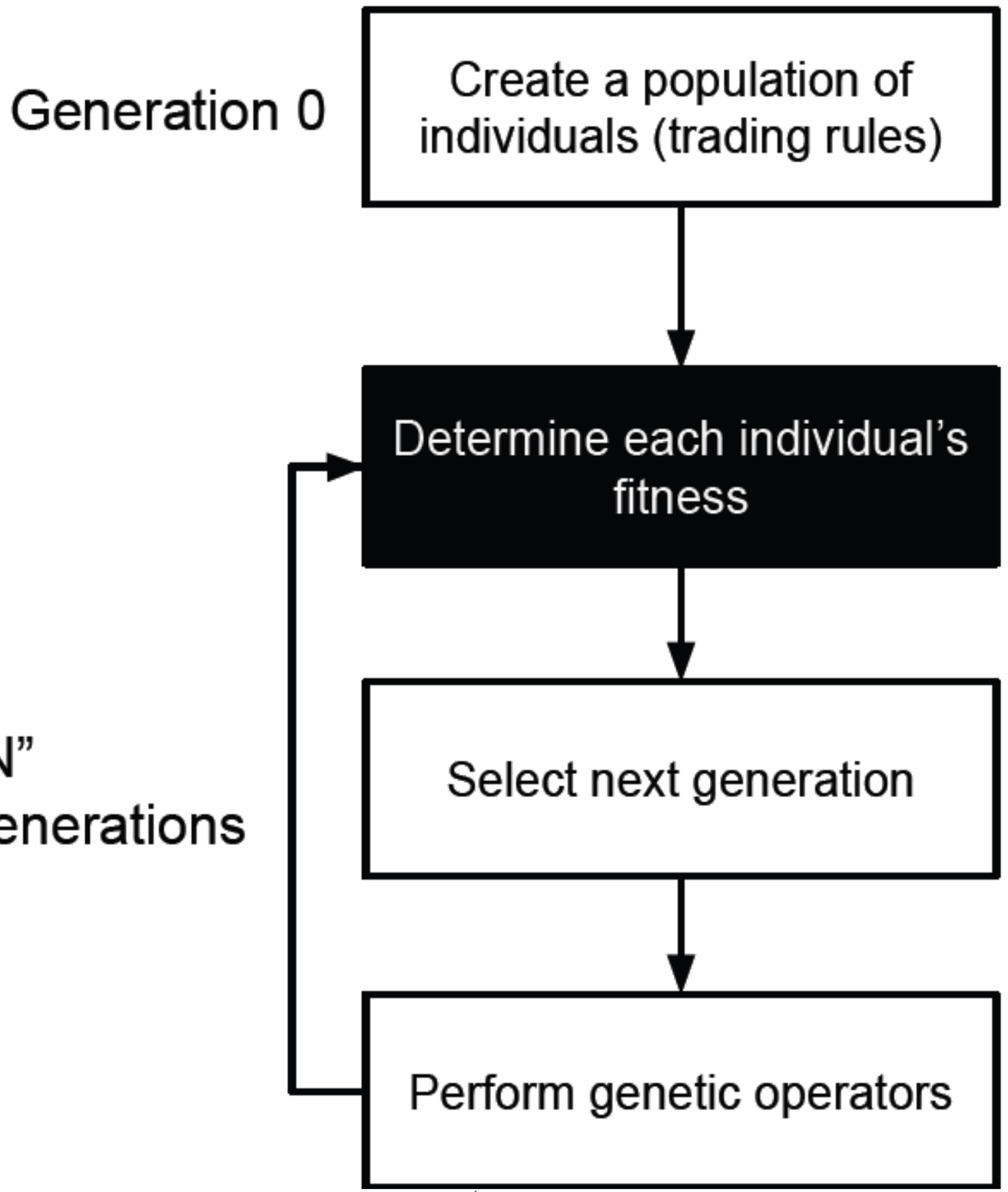
B. **Mixed precision optimised round-off error+performance**
   - compute: fixed-point, accumulation: floating point

C. Stratix V 5SGS FPGA: **22 speedup = 3.5x higher return**
   - than optimised 12-core CPU C++ implementation

Challenges

- Huge quantity of data
- Speed of processing
- Accurate market status feedback

Optimise trading: Genetic Algorithm



for acceleration

Trading Indicators

Moving Averages (MA)

$$SMA = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n}$$

$$SMA_{\text{today}} = SMA_{\text{yesterday}} + \frac{p_{M-n}}{n} - \frac{p_M}{n}$$

William%R

$$\%R = \frac{high_{N\,days} - close_{today}}{high_{N\,days} - low_{N\,days}} \times -100$$

Trading Strategy Example

IF

[(0.1 + SlopeBid) / (levelsBid * 0.2)] + [(deltaBid / depthBid) + (slopeAsk - 0.1)]

≤

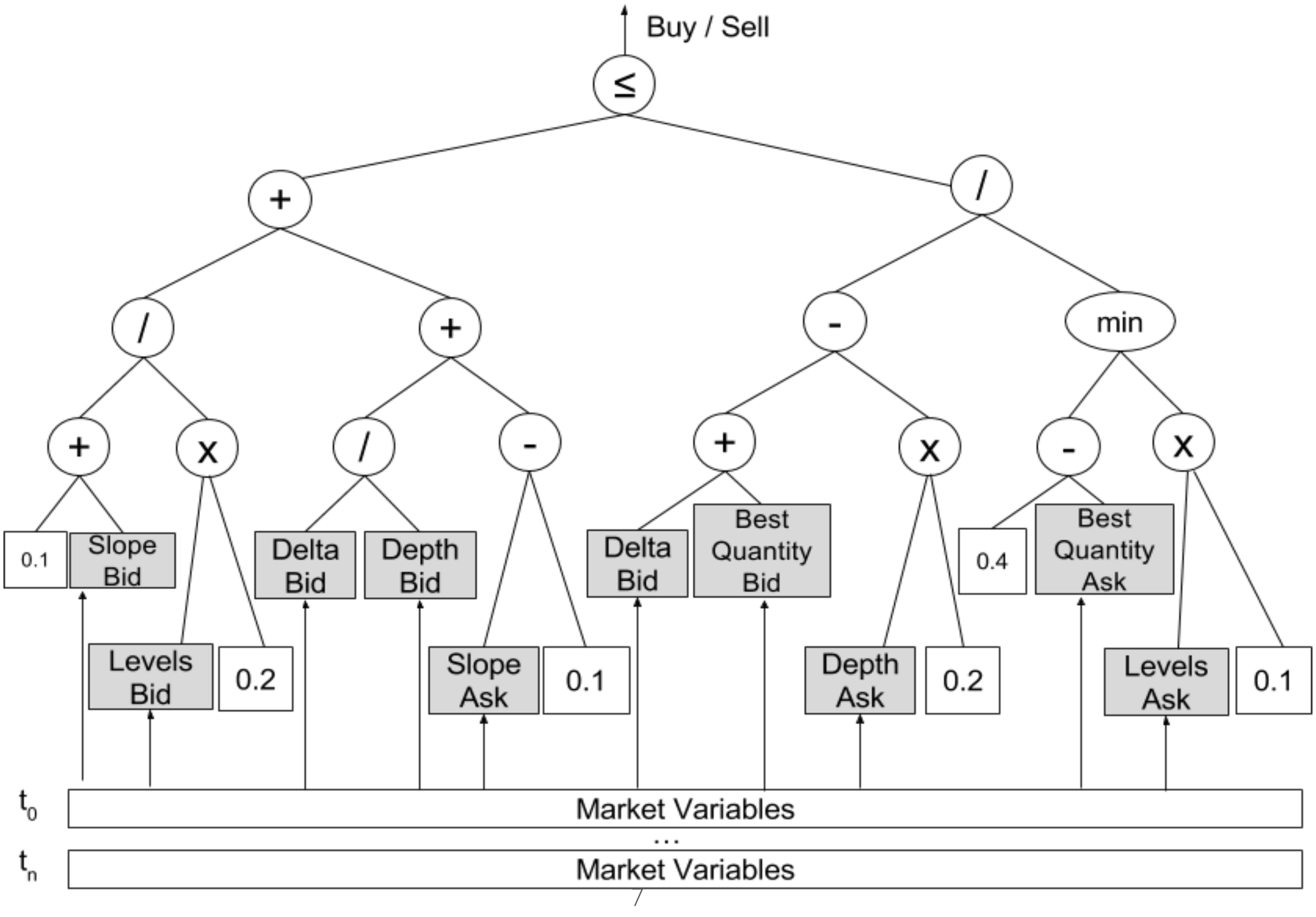
[(DeltaBid + BestQuantityBid) - (DepthAsk * 0.2)] / [***min***(0.4 - BestQuantityAsk), (LevelsAsk * 0.1))]

THEN **BUY**

ELSE **SELL**

The financial instrument

Trading Strategy representation



Fitness Evaluation Details

We examine the fitness profitability of filter trading rules. Thus the trader will only reevaluate its position when:

$$|p_t - p_{t1}| >= k$$

where *pt* is price at time t, pt1 price at time t1 with t1 the time of last trader's transaction and k is the filter threshold.

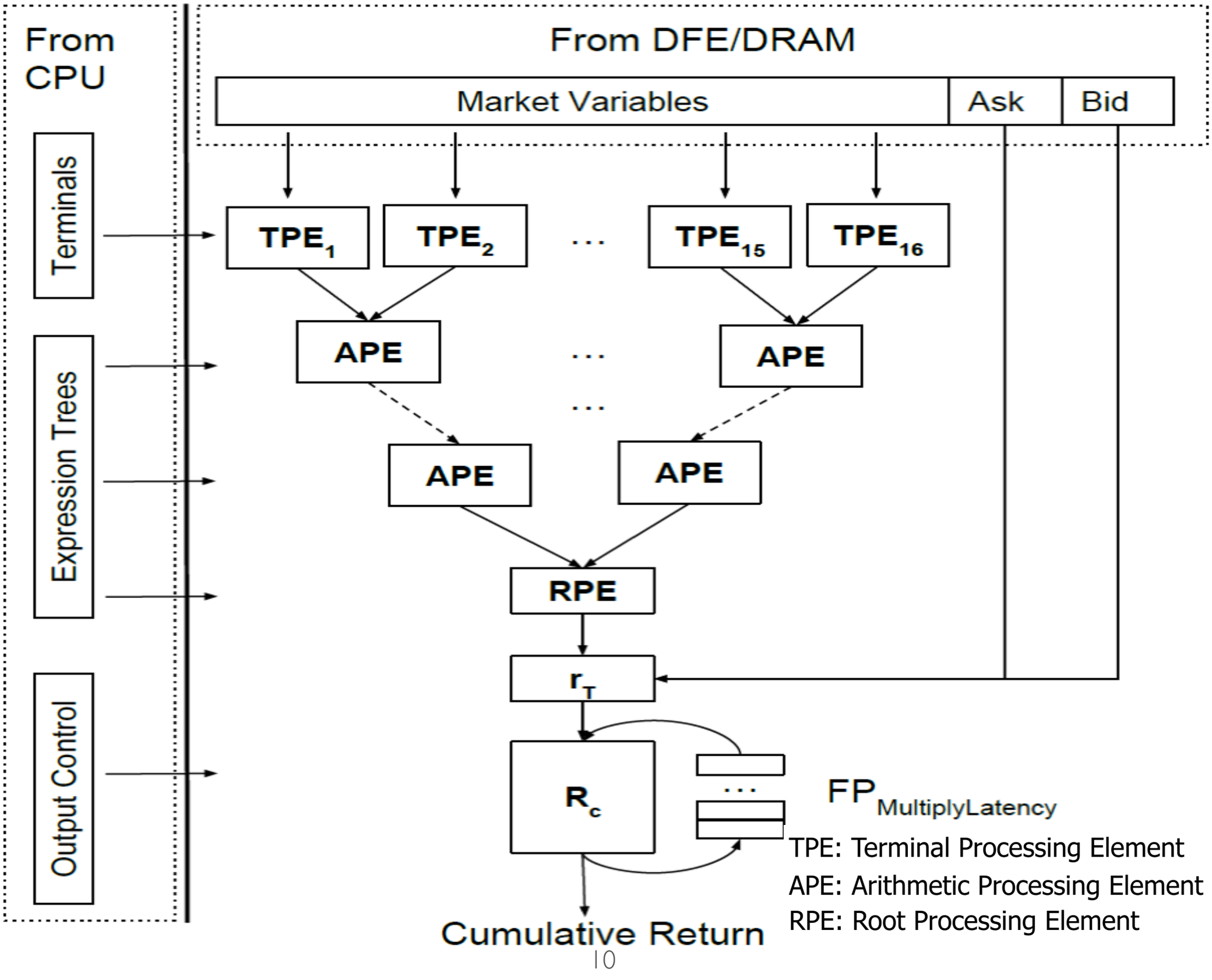
Fitness Evaluation Details

The performance measure used to evaluate our trading rules is represented by the *cumulative returns:*

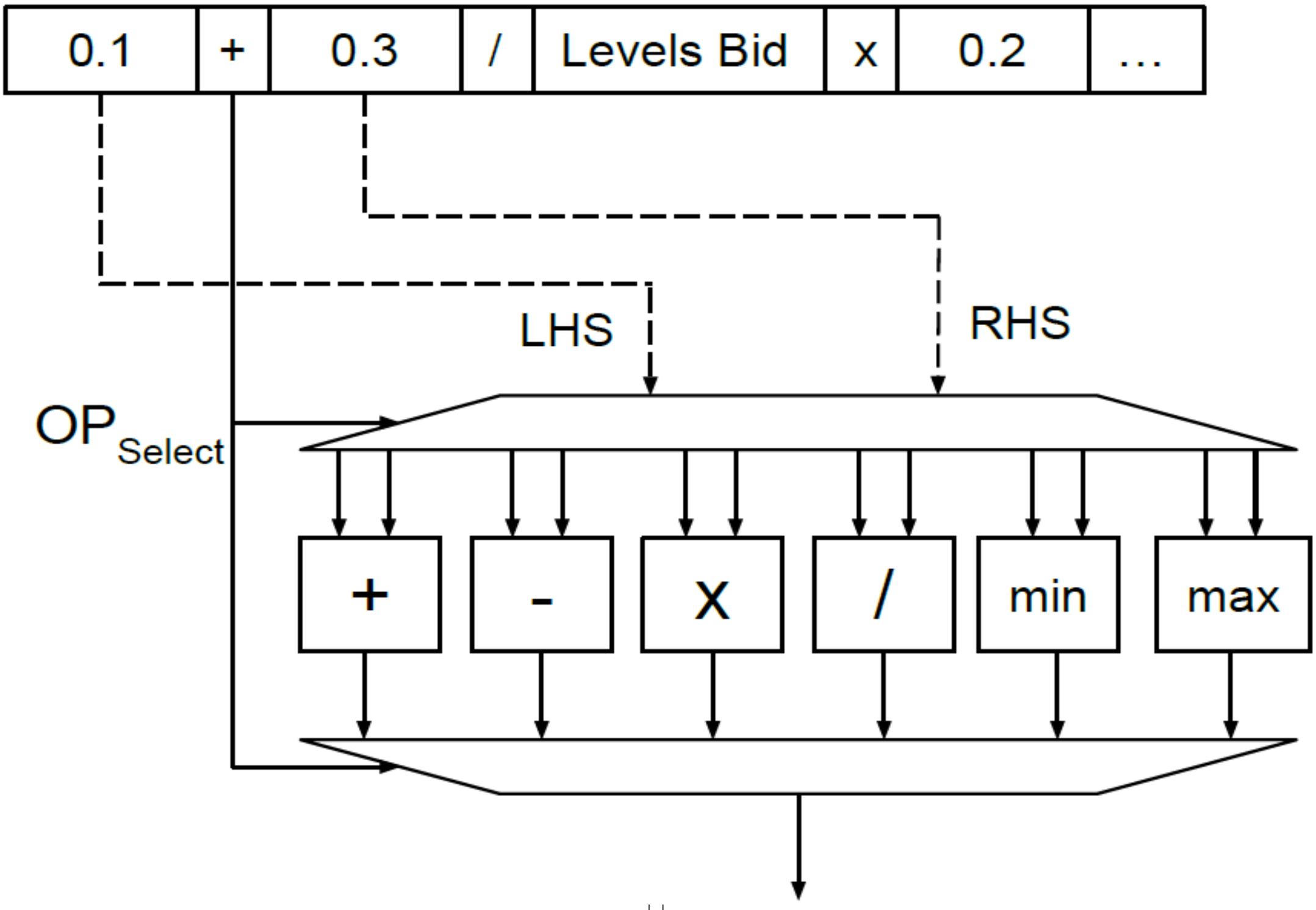
$$R_c = \Pi_t(1 + z_t * r_t) - 1$$

Where rt is the one-period return of the exchange rate, pt corresponds for the best bid or best ask price and zt is the sterling position.

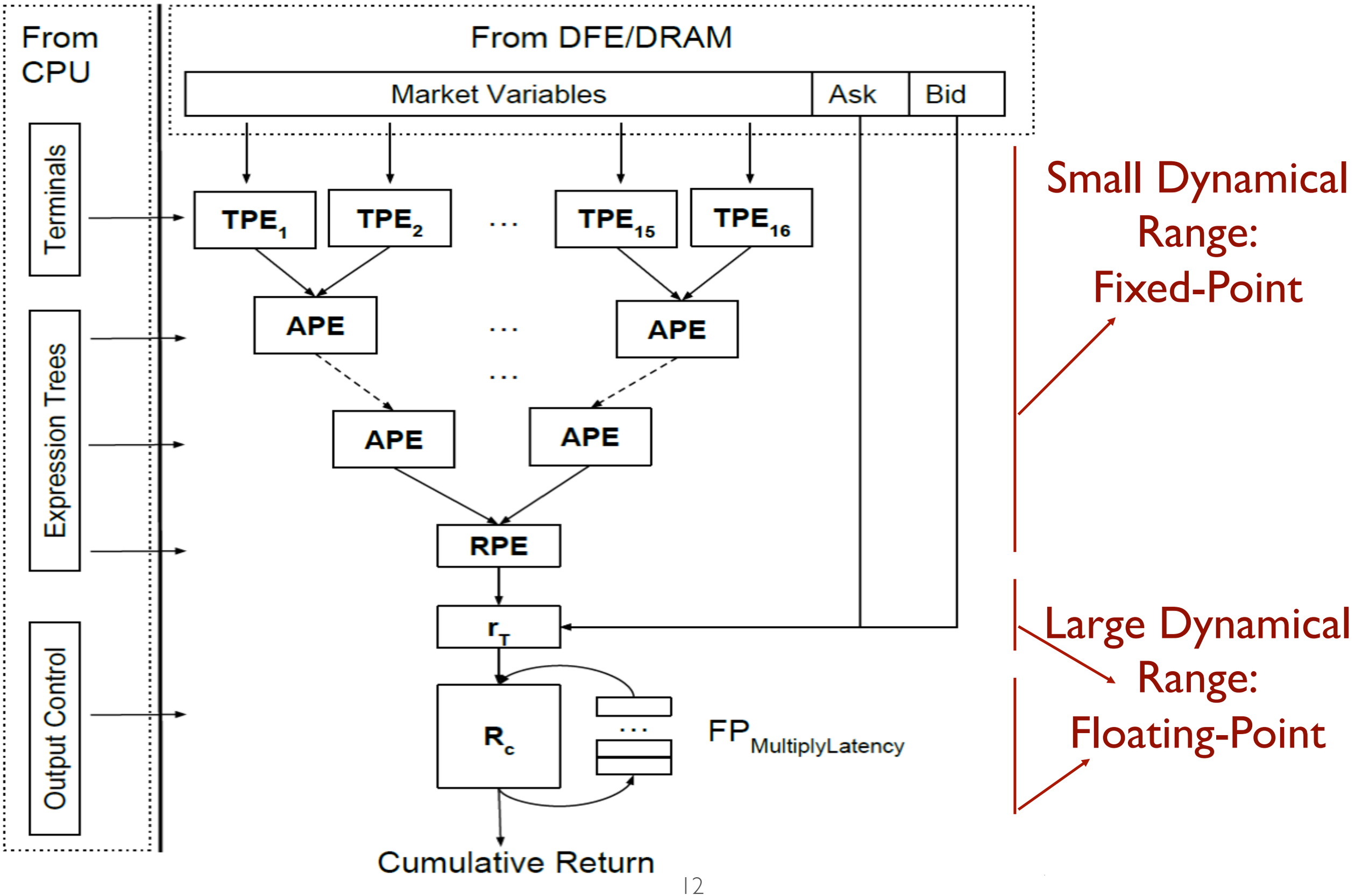
ARCHITECTURE



Arithmetic Processing Element



Mixed precision



Mixed precision

32-bit fixed-point representation:

- 16 bit integer
- 16 bit fraction

Single floating-point representation:

- 8 bit integer
- 24 bit fraction

Double floating-point representation:

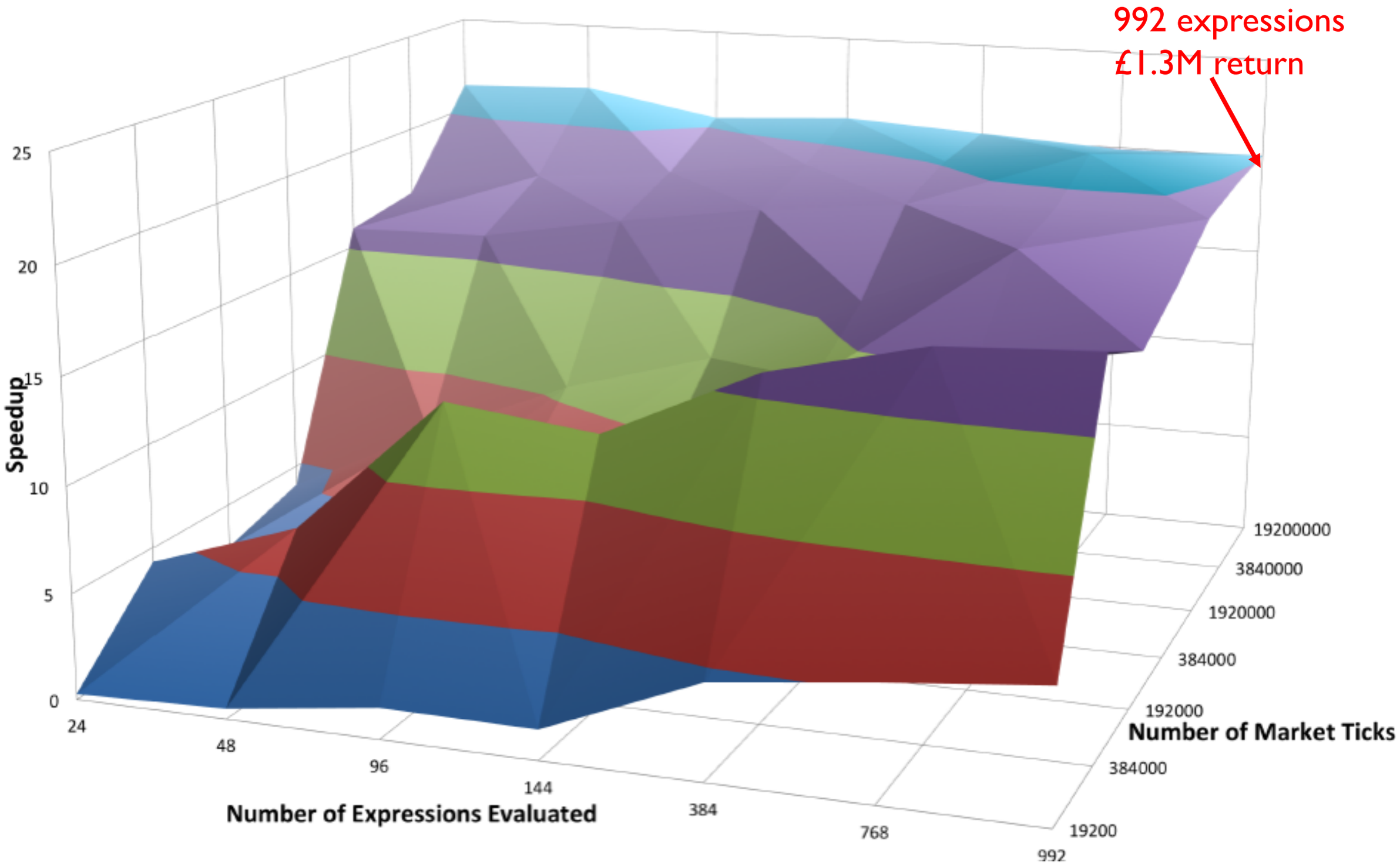
- 11 bit integer
- 53 bit fraction

System Properties

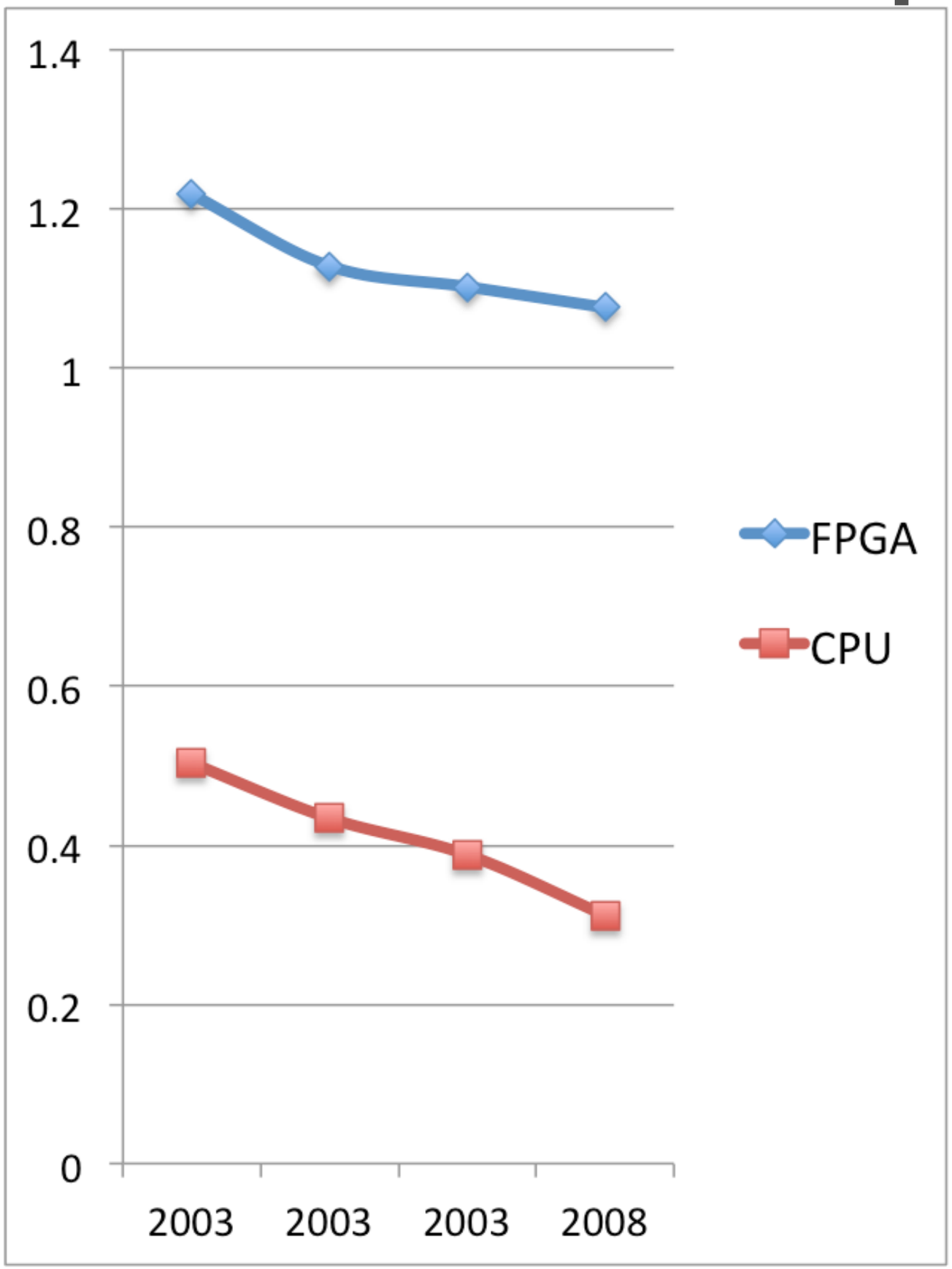
CPU	Dual-Intel Xenon E5-2640, 6 cores per CPU
CPU Cache	15 MB
CPU DRAM	64 GB DDR3-1333
CPU DRAM Bandwidth	42.6 GB/s (Peak)

FPGA	Stratix V 5SGSMD5N1F45C2
FPGA DRAM	48 GB
FPGA DRAM Bandwidth	38 GB/s (Achieved)
CPU to FPGA Bandwidth	2 GB/s

Evaluation


992 expressions
£1.3M return
Speedup
25
20
15
10
5
0
24
48
96
144
384
768
992
Number of Expressions Evaluated
19200000
3840000
1920000
384000
192000
384000
19200
Number of Market Ticks

Comparison



~20x speedup

means:

Financial institution:

~3.5x higher return!

Regulators can *analyse*:

~20x more trading rules

Measured Speedup

# Pipes	1	2	4	8
Average CPU Time	251.652	251.652	251.652	251.652
FPGA Time (s)	98.7812	50.0130	25.0830	12.5353
Measured Speedup	2.5475	5.0317	10.0327	20.0435

Maximum number of computing units on FPGA: 8

Results obtained after evaluating on 19.2M market ticks, 992 expressions, clock frequency at 190MHz

Statistically Significant Predictability

- Anatolyev-Gerko test
- T-Statistics test

Anatolyev-Gerko Test

BUY/ SELL follows the **majority** of the trading strategies

is the number of trading strategies

#	Jan(20-24)' 03	Feb(17-21)' 03	March(10-14)' 03	March 31, '08
992	1.82	1.52	1.73	-0.37
768	1.56	1.63	1.22	0.46
384	1.20	1.17	0.93	-0.66
144	0.84	0.92	0.44	-1.02

Work Under Review

Highly-efficient run-time reconfiguration design:

- *at compile time* we prepare a number of likely configurations (e.g. one with the "/" completely removed, and one which includes it)

- *at runtime* we group the expressions according to operator usage, thus loading the appropriate configuration for each group; we then execute it and send the results to the CPU

Submitted to Journal of Signal Processing Systems, 2016

Future Work

- Determine the best tradeoff between energy/power and performance
- Extend the GP alphabet to identify more complex trading strategies
- Explore other kinds of trading strategies optimizations

Summary

A. **Novel pipelined architecture for trading strategies:**
   - evaluate fitness function of complete expression trees

B. **Mixed precision optimised round-off error+performance**
   - compute: fixed-point, accumulation: floating point

C. Stratix V 5SGS FPGA: **22 speedup = 3.5x higher return**
   - than optimised 12-core CPU C++ implementation

APPENDIX

T-Statistics Test

#	Jan(20-24)' 03	Feb(17-21)' 03	March(10-14)' 03	March 31, '08
992	1.625	0.801	0.543	0.463
768	1.608	0.768	0.477	0.494
384	1.486	0.698	0.312	0.395
144	1.583	0.606	0.301	0.434

is the Trading rules number

Individual Returns

#	Jan(20-24)' 03	Feb(17-21)' 03	March(10-14)' 03	March 31, '08
992	1.278	1.188	1.103	1.076
768	1.047	1.024	0.998	0.937
384	0.904	0.856	0.889	0.793
144	0.789	0.683	0.654	0.578

is the Trading rules number

Testing for Robustness

In **aggregate-testing** we evaluate how many times (out of 150 tests performed) the TTest trading strategy from January appears in the February and March top (15% TTest) strategies, and so on.

Table: 2003 sample aggregate testing

Iterations	Feb/March	Jan/March	Jan/Feb
1000	137/118	142/108	136/120
800	127/102	130/103	122/119
600	122/98	117/92	117/103
400	117/92	121/99	115/92
200	103/86	97/93	95/84

Testing for Robustness

In **rejection-acceptance** testing we evaluate whether or not the least t strategies are selected again in the best population half after earlier having rejecting them, according to the following formula:

$$|TotalAccepted_N = \sum_{i=0}^{N-1} i * itAccepted_i$$

Where *itAccepted* = # of individuals being accepted after one iteration only, with i representing the iteration number.

Table: 2003 sample rejection-acceptance testing

Iterations	Jan%	Feb%	March%	Jan-March%
1000	38,40%	30,51%	40,07%	36,33%
600	45,11%	44,19%	39,44%	42,91%
200	28,70%	21,08%	25,18%	21,08%

Terminal PRocessing element

The TPE processes expression terminals which can be either constant or indices into the market variables.

Root Processing element

The RPE evaluates comparison operators such as: $\leq$, $\geq$.

The result produced decides whether to buy or sell the particular financial instrument.

Design Operation

1. Load market data to accelerator DRAM;
2. Queue expression trees from CPU to FPGA BRAM;
3. Evaluate expression on market data;
4. Fetch next expression to FPGA BRAM;
5. Output partial results to CPU;
6. Repeat the above steps until done.