Braiding: a Scheme for Resolving Hazards in NORMA

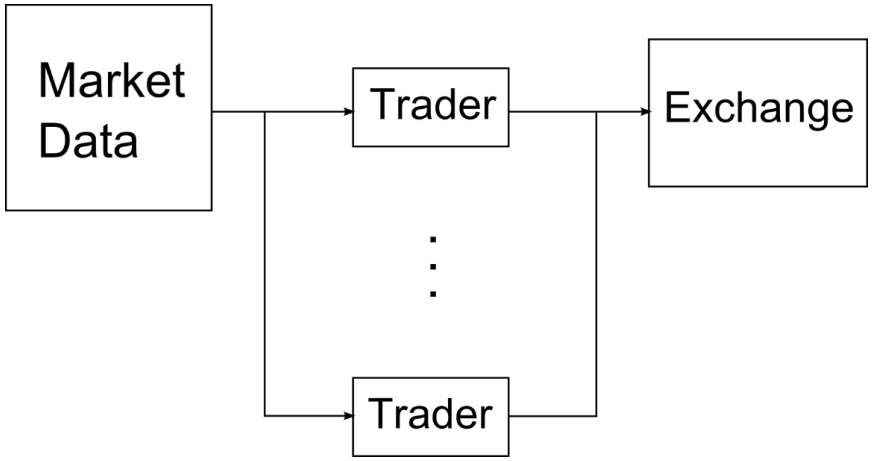
Stephen Tridgell, Duncan J.M. Moss, Nicholas J. Fraser and Philip H.W. Leong
School of Electrical and Information Engineering,
The University of Sydney

THE UNIVERSITY OF SYDNEY

Motivation (latency)

How to beat other people to the money (latency)

- Low latency trading looks to trade in transient situations where market equilibrium disturbed
  - 1ms reduction in latency can translate to $100M per year



- Latency also important to: prevent blackouts (cascading faults), turn off machine before it damages itself, etc

Information Week: Wall Street's Quest To Process Data At The Speed Of Light

Latency infrastructure already available

Exablaze Low-Latency Products

ExaLINK Fusion 48 SFP+ port layer 2 switch for replicating data typical 5 ns fanout, 95 ns aggregation, 110 ns layer 2 switch

Xilinx Ultrascale FPGA, QDR SRAM, ARM processor

ExaNIC X10 typical raw frame latency 60 bytes 780 ns

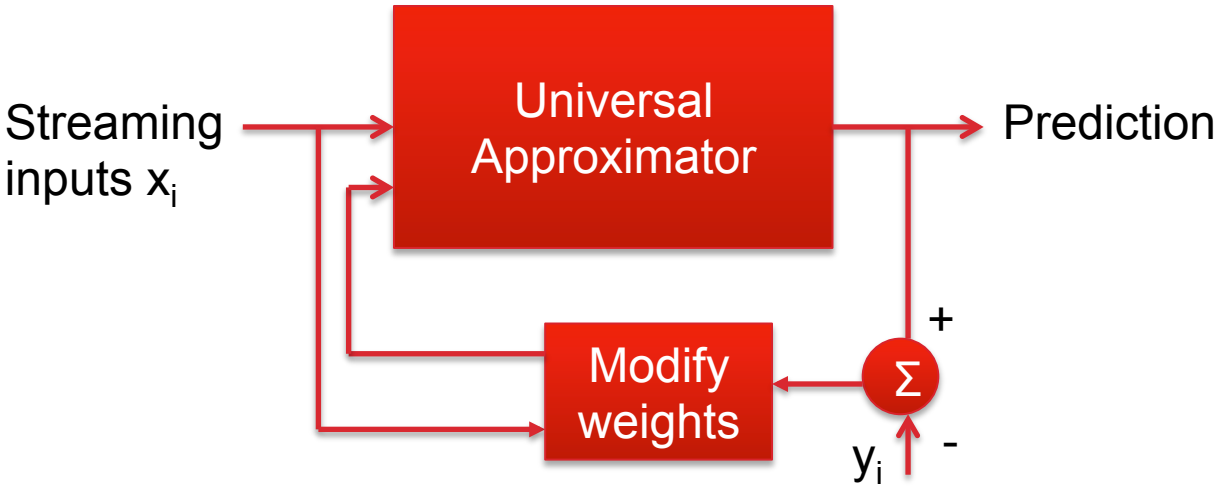
What we can't do: ML with this type of latency

Source: exablaze.com

Online Kernel Methods

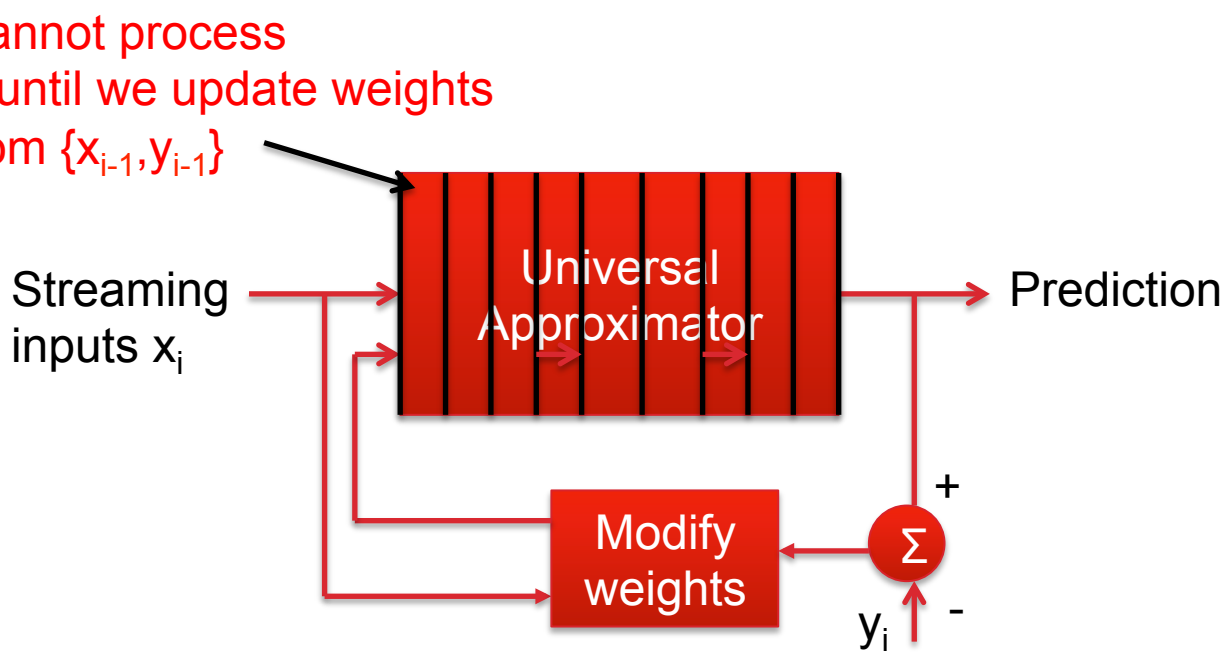
Examples are KLMS and KRLS

- Traditional ML algorithms batch based
  - Several passes through data
  - Requires storage of input data
  - Not suitable for massive datasets
- Our approach: online algorithms
  - Incremental, inexpensive state update based on new data
  - Single pass through the data
  - Can be high throughput, low latency



Obstacle to Pipelining

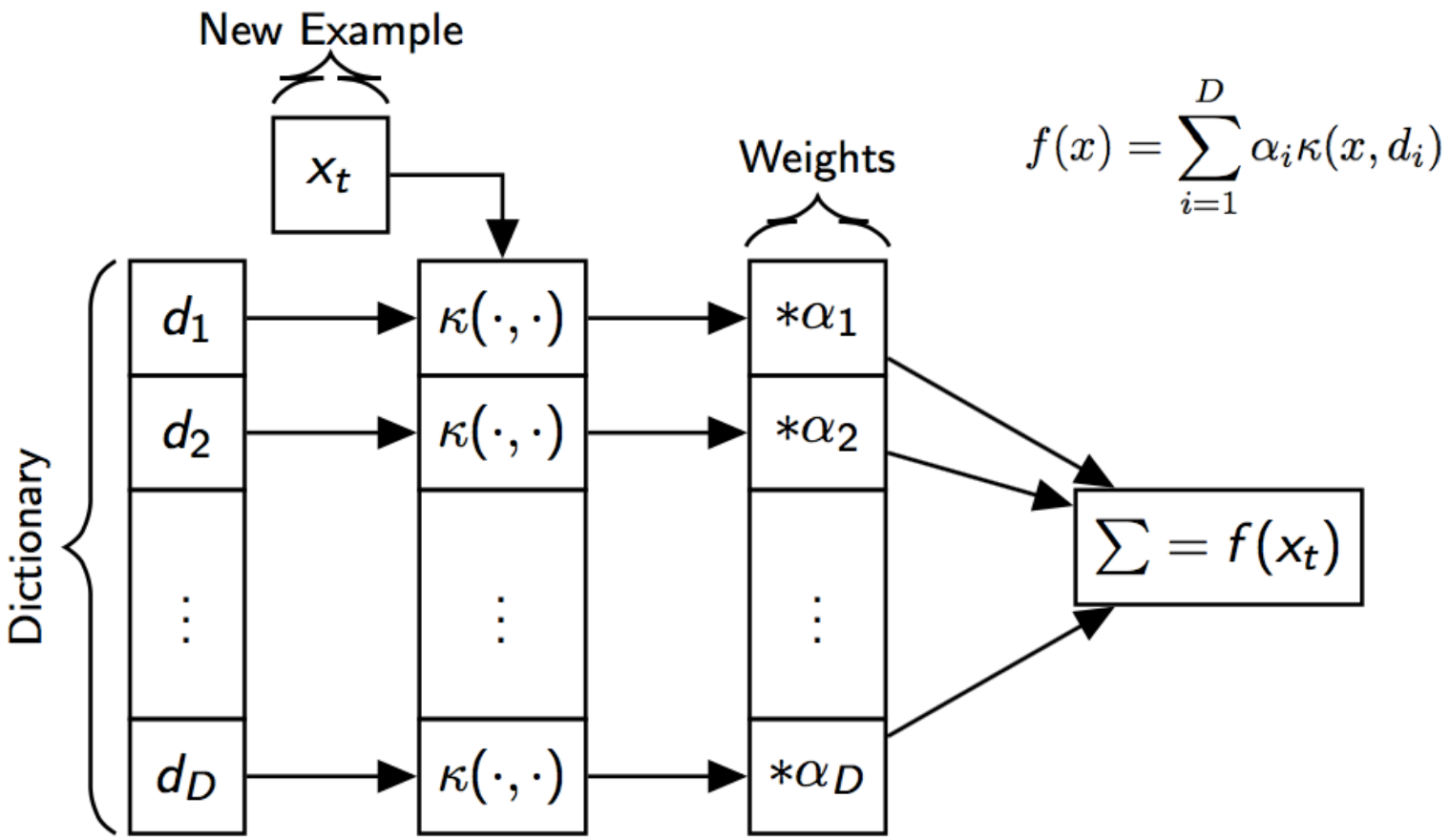
Dependency Problem



› This work

- Implementation of an online machine learning scheme: NORMA
- Resolve read-after-write dependencies through braiding

Datapath for NORMA



NORMA

Naive Online regularised Risk Minimization Algorithm

› Finds Dictionary d_i, and α_i (weights)

$$f(x) = \sum_{i=1}^{D} \alpha_i \kappa(x, d_i)$$

› Minimise predictive error ($R_{inst,\lambda}$) by taking a step in direction of gradient

$$f_{t+1} = f_t - \eta_t \partial_f R_{inst,\lambda}[f, x_{t+1}, y_{t+1}]\Big|_{f=f_t}$$

› Can be used for classification, regression, novelty detection

› Update for novelty detection

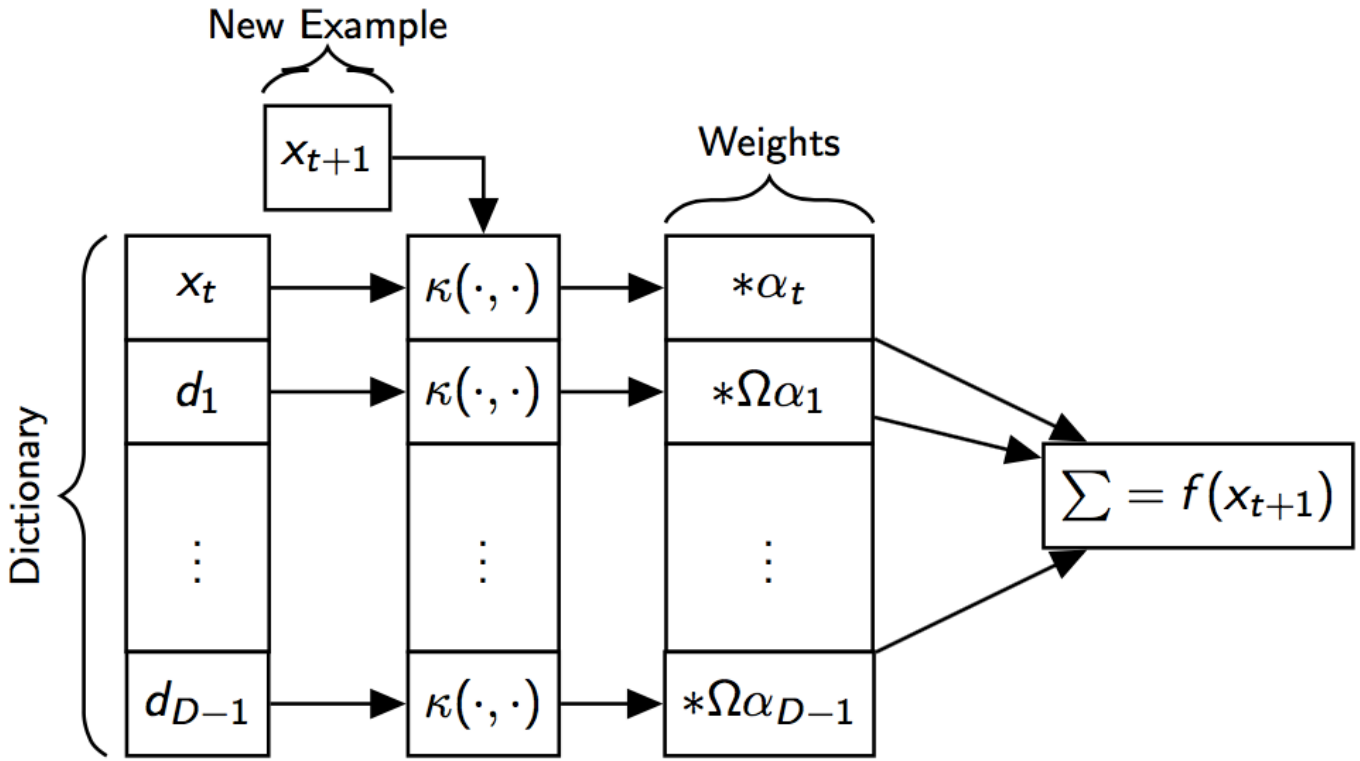
$$(\alpha_i, \alpha_t, \rho) = \begin{cases} (\Omega\alpha_i, 0, \rho + \eta\nu) \text{ if } f(x_t) \geq \rho \\ (\Omega\alpha_i, \eta, \rho - \eta(1-\nu)) \text{ otherwise} \end{cases}$$

Add x_{t+1} to dictionary

NORMA Update (Case 1)

$$(\alpha_i, \alpha_t, \rho) = \begin{cases} (\Omega\alpha_i, 0, \rho + \eta\nu) \text{ if } f(x_t) \geq \rho \\ (\Omega\alpha_i, \eta, \rho - \eta(1-\nu)) \text{ otherwise} \end{cases}$$

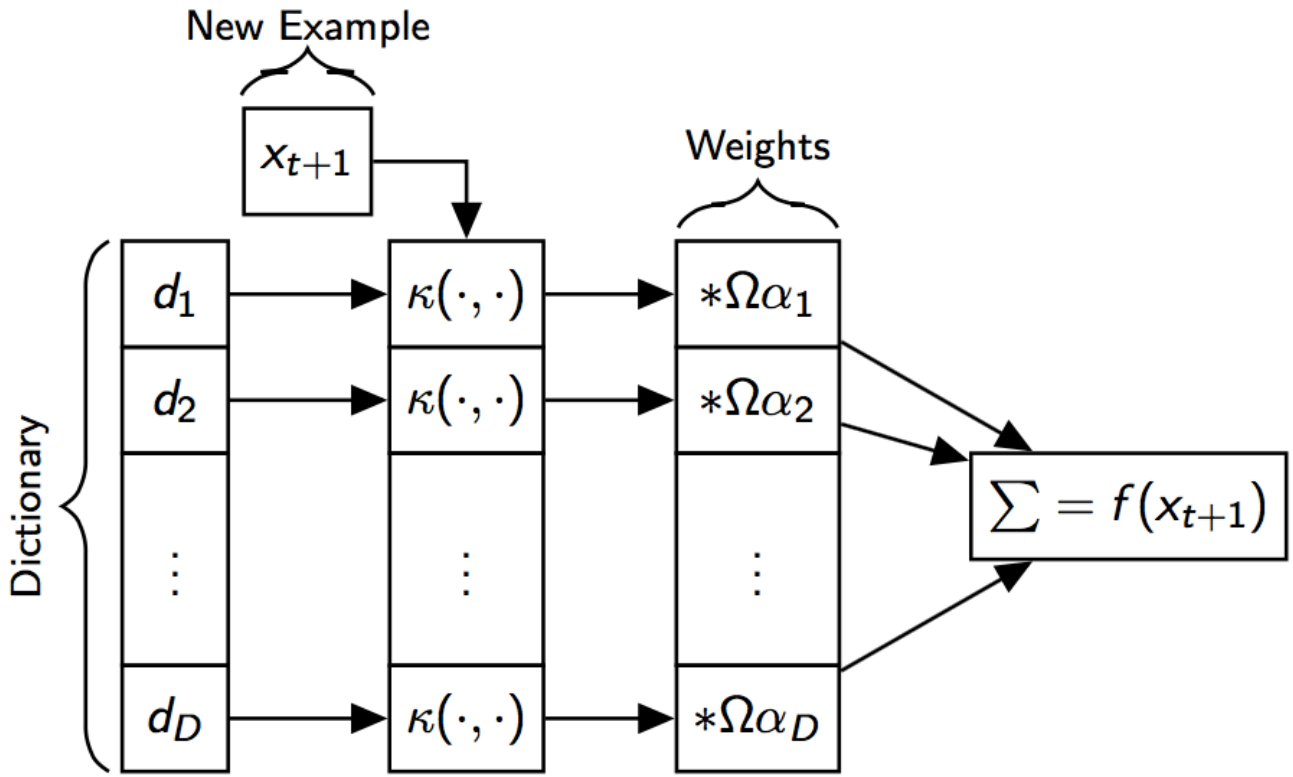
(Add x_t to dictionary)



NORMA Update (Case 2)

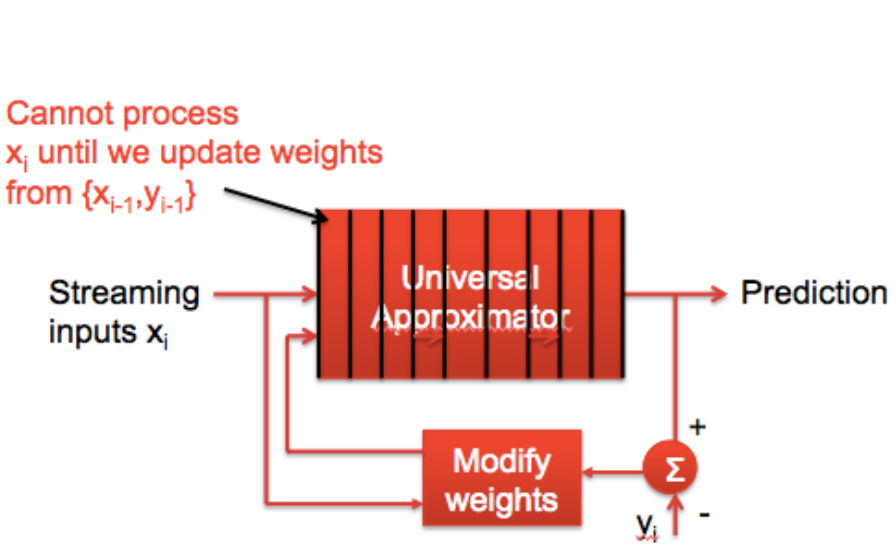
$$(\alpha_i, \alpha_t, \rho) = \begin{cases} (\Omega\alpha_i, 0, \rho + \eta\nu) \text{ if } f(x_t) \geq \rho \\ (\Omega\alpha_i, \eta, \rho - \eta(1-\nu)) \text{ otherwise} \end{cases}$$

(Add x_t to dictionary)



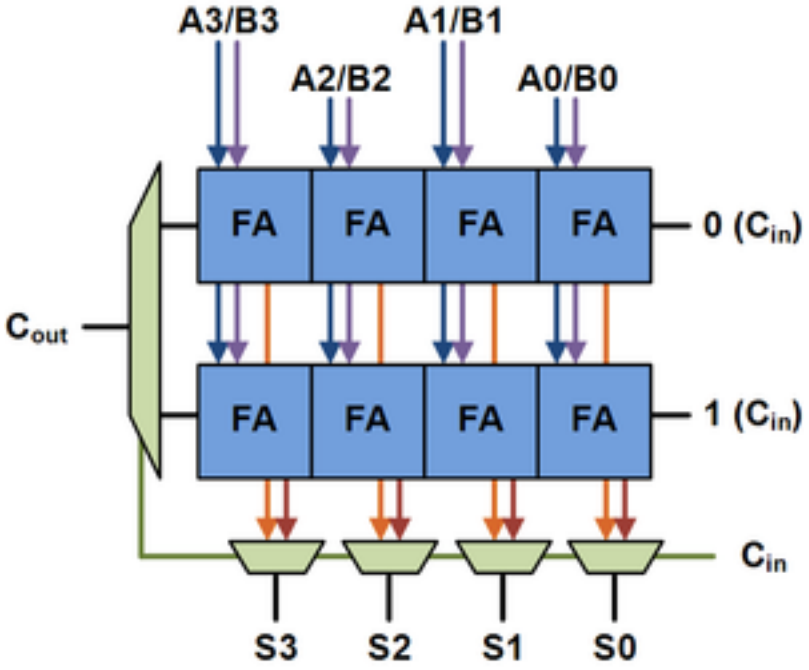
Properties of NORMA

- NORMA is a sliding window algorithm
  - If new dictionary entry added $[d_1,\cdots d_D] \rightarrow [x_t,d_1,\cdots d_{D-1}]$
  - Weight update is just a decay $\alpha_i \rightarrow \Omega\alpha_i$
  - Update cost is small compared to computing $f(x_t)$
- **Is this really true?**



Idea

- Recall carry select adder
  - implement both cases in parallel and select output



Source: Wikipedia

Braiding

$$f(x_{t+1}) = \sum_{i=1}^{D} \alpha_i \kappa(x_{t+1}, d_i)$$

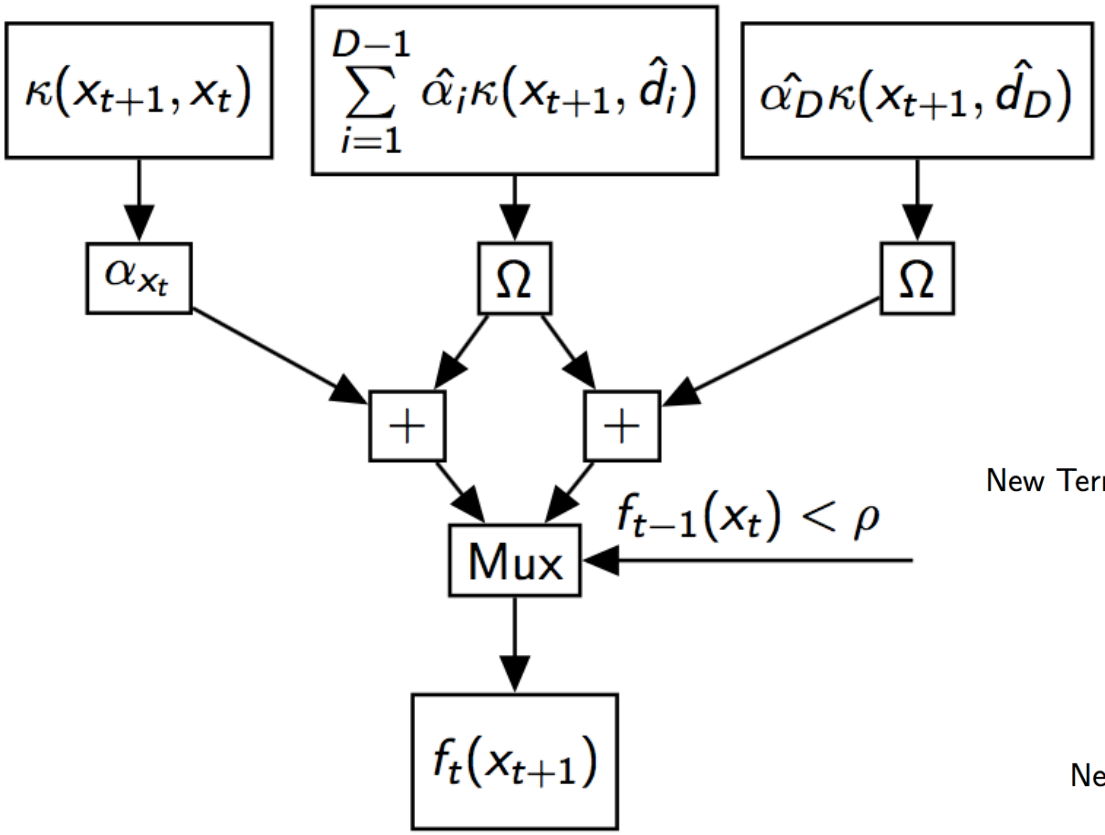
Use the previous dictionary for x_t denoted $\hat{d}_i$

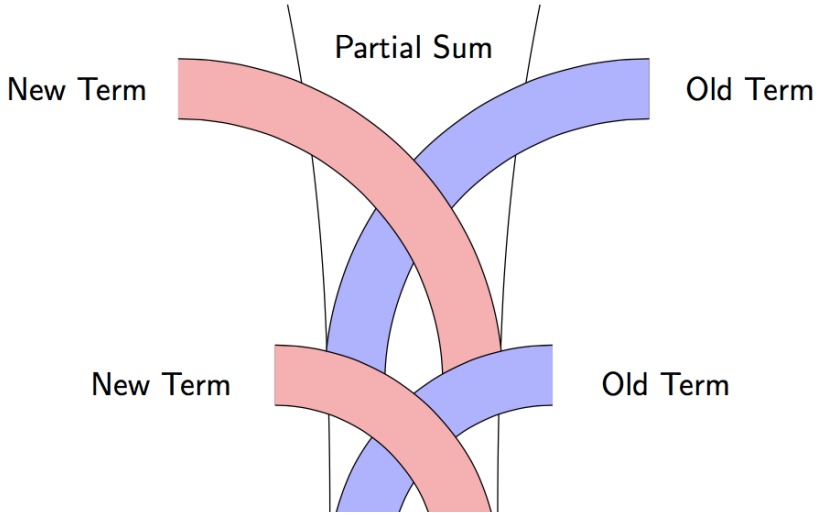
$$f(x_{t+1}) = \sum_{i=1}^{D-1} \Omega\hat{\alpha}_i \kappa(x_{t+1}, \hat{d}_i) + \text{something}$$

if x_t is added then this term $= \alpha_{x_t} \kappa(x_{t+1}, x_t)$

if x_t is not added then this term $= \Omega\hat{\alpha}_D \kappa(x_{t+1}, \hat{d}_D)$

Braiding Datapath





Generalised to p cycles

$$f_t(x_{t+1}) = \sum_{i=1}^{D-p} \Omega^p \hat{\alpha}_i \kappa(x_{t+1}, \hat{d}_i)$$

$$q \begin{cases} + \begin{cases} 0 \text{ if } x_{t+1-p} \text{ is not added} \\ \Omega^{p-1} \alpha_{x_{t+1-p}} \kappa(x_{t+1}, x_{t+1-p}) \text{ otherwise} \end{cases} \\ + \begin{cases} 0 \text{ if } x_{t+2-p} \text{ is not added} \\ \Omega^{p-2} \alpha_{x_{t+2-p}} \kappa(x_{t+1}, x_{t+2-p}) \text{ otherwise} \end{cases} \\ \vdots \\ + \begin{cases} 0 \text{ if } x_t \text{ is not added} \\ \alpha_{x_t} \kappa(x_{t+1}, x_t) \text{ otherwise} \end{cases} \end{cases}$$

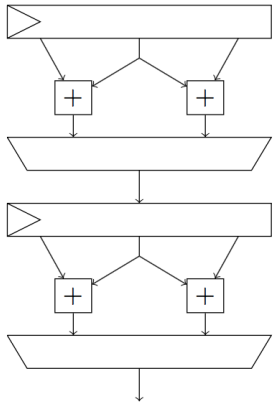
$$+ \sum_{i=D-p+1}^{D-q} \Omega^p \hat{\alpha}_i \kappa(x_{t+1}, \hat{d}_i)$$

Pipeline (p cycles)

$m_i = \kappa(d_i, x_j)$
(k cycles)

$f_t(x_{t+1}) = \sum_{i=1}^{D} \alpha_i m_i$ (s cycles)

$\alpha(f_t(x_{t+1}))$
(1 cycle)



Implementation

- Implemented in Chisel
- On XC7VX485T- 2FFG1761C achieves ~133 MHz
- Area O(FDB^2) (F=dimensionality of input vector), time complexity O(FD)
- Speedup 500x compared with single core CPU i7-4510U (8.10 fixed)

F=8, D=	16	32	64	128	200
Frequency (MHz)	133	138	137	131	127
DSPs (/2,800)	309	514	911	1,679	2,556
Slices (/759,000)	4,615	8,194	14,663	29,113	46,443
Latency (cycles)	10	11	12	12	13
Speedup (×)	47	91	178	344	509
Latency reduction (×)	4.69	8.30	14.9	28.7	39.2

THE UNIVERSITY OF SYDNEY

Comparison of Architectures

› Core with input vector F=8 and dictionary size D=16

Design	Precision	Freq MHz	Latency Cycles	T.put Cycles	Latency nS	T.put nS
Vector KNLMS	Single	282	479	479	1,699	1,699
Pipelined KNLMS	Single	314	207	1	659	3.2
Braided NORMA	8.10	113	10	1	89	8.8

THE UNIVERSITY OF SYDNEY

Summary

- Braiding: rearrangement of a sliding window algorithm for hardware implementations
  - NORMA used but other ML algorithms possible
- Compared with pipelined KNLMS,
  - 20x lower latency at 1/3 of the throughput
- Open source (GPLv2): github.com/da-steve101/chisel-pipelined-olk