“Video Tracking” using Profile Guided Dataflow Transformation

Andrew Wallace, Institute for Sensors, Signals and Systems
Heriot-Watt University

… being a discussion of some of the activity on

EP/K009931/1: Programmable embedded platforms for remote and compute intensive image processing applications, 2013-2017 (‘RATHLIN’)

Greg Michaelson, Rob Stewart, Deepayan Bhowmik, Nathanel Lemessa Baisa, HWU, Roger Woods, Fahad Siddiqui, Colm Kelly and Burak Bardak, QUB (with INSA, Clermont-Ferrand, EPFLausanne, Thales, Xilinx)

ECIT | The Institute of Electronics, Communications and Information Technology

EPSRC
Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

Objectives of the EPSRC Programme

- Use a “model of computation” dataflow process network (DPN) representation which will allow the processing and data organisation needs of image processing/analysis (IP/A) to be readily captured.
- Develop a domain specific Image Processing Processor (IPPro) processor architecture
- Develop code translation and transformation techniques that will allow efficient implementation on a variety of platforms (e.g. Multicore CPU, FPGA, GPU, IpPro)
- Develop a Domain Specific Language, RIPL, ‘above’ the DPN,for ease of use by practitioners
- Evaluate using a set of prototypical and novel IP/A algorithms expressed as application specific DPNs

ECIT The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

The Target: a Distributed, Heterogeneous Architecture

Research Challenge

- Data intensive image processing applications e.g. Video Analytics, Surveillance, Smart Cameras and other sensors
- Option of distributed front-end image processing to reduce communication (and other costs) of backend processing

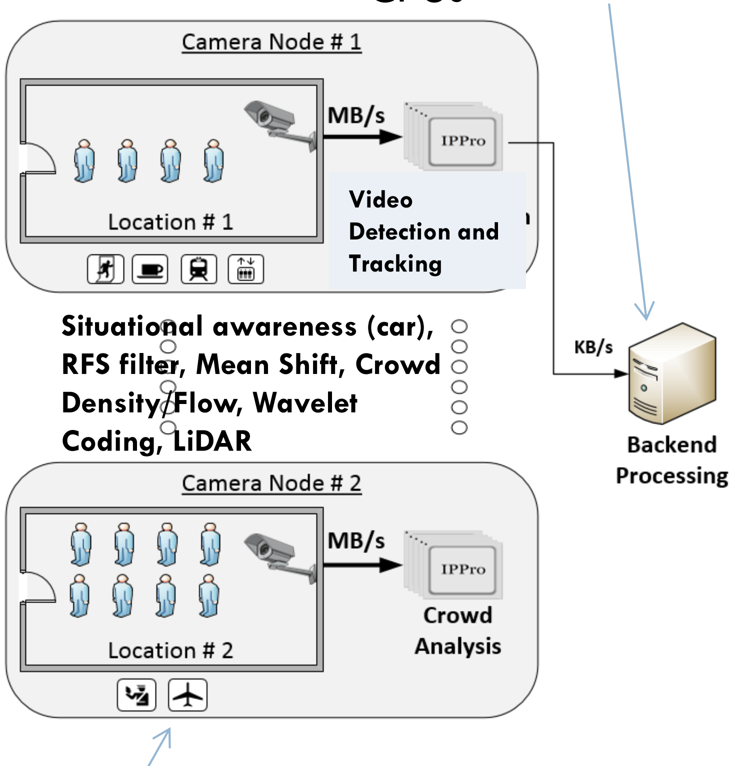
Research Methodology

- Distributed computing
- Data or control level parallelism (DLP)
- Programmability and Performance
- Single/Multiple Instruction Multiple Data (S/MIMD)

Processor based Architecture

- Programmability
- Scalability
- Flexibility
- Efficient Resource Utilization

e.g. multicore CPUs


Camera Node # 1
MB/s
IPPro
Location # 1
Video Detection and Tracking
Situational awareness (car), RFS filter, Mean Shift, Crowd Density/Flow, Wavelet Coding, LiDAR
KB/s
Backend Processing
Camera Node # 2
MB/s
IPPro
Location # 2
Crowd Analysis

e.g. Xilinx-7: FPGA hardware plus ARM

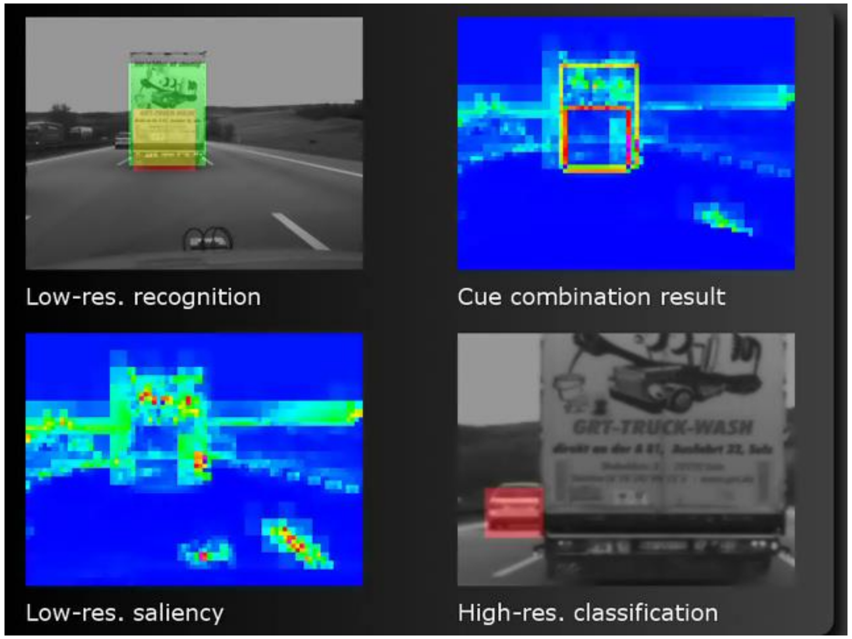
ECIT The Institute of Electronics, Communications and Information Technology

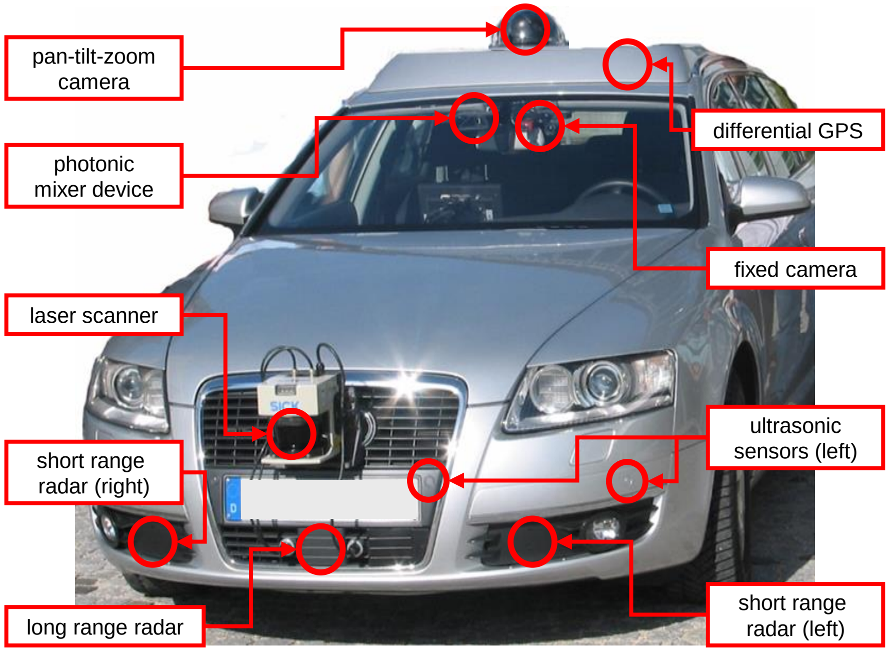
EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

What kind of Dynamic Imaging?

Multi-target tracking, either from a CCTV network, or from a mobile vehicle or vehicles (Sensor – Region – Algorithm Utility)





S Matzka, AM Wallace and YR Petillot, "Efficient Resource Allocation for Automotive Attentive Vision Systems", IEEE Transactions on Intelligent Transportation Systems, 859-872, 2012.

ECIT | The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

What kind of Dynamic Imaging?

Multi-target tracking, either from a CCTV network, or from a mobile vehicle or vehicles

W Limprasert, AM Wallace and G Michaelson. “Real-time People Tracking in a Camera Network”, IEEE Journal on Circuits and Systems, 263-271 June 2013

ECIT | The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

What's the problem?

Code development for parallel or heterogeneous architectures …. e.g.
it took several months to hand-craft GPU code to detect, track and associate 5-10 subjects with live video with two cameras at 10fps (40fps on recorded video)

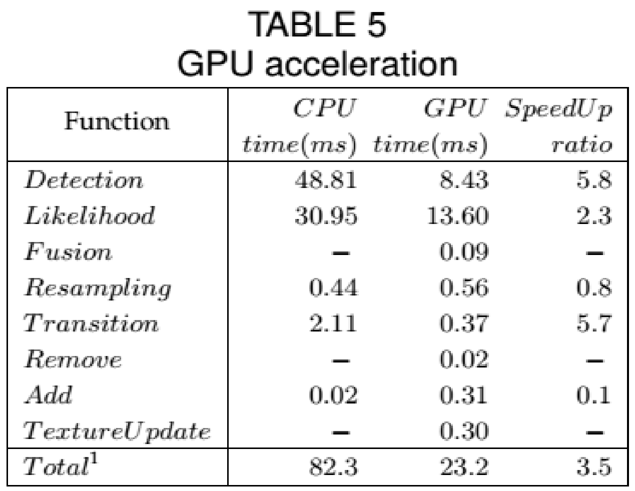
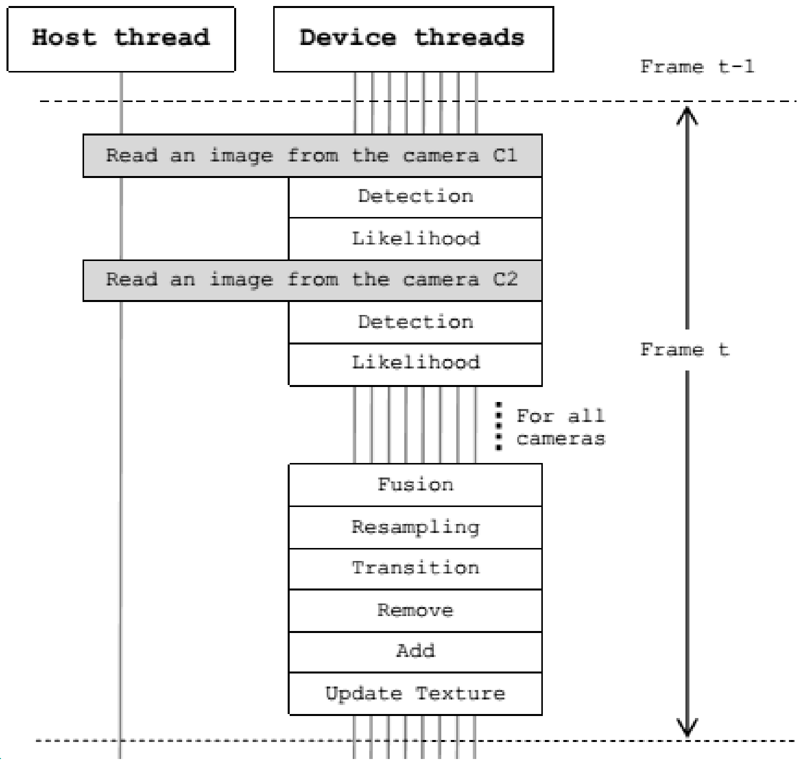


TABLE 5
GPU acceleration

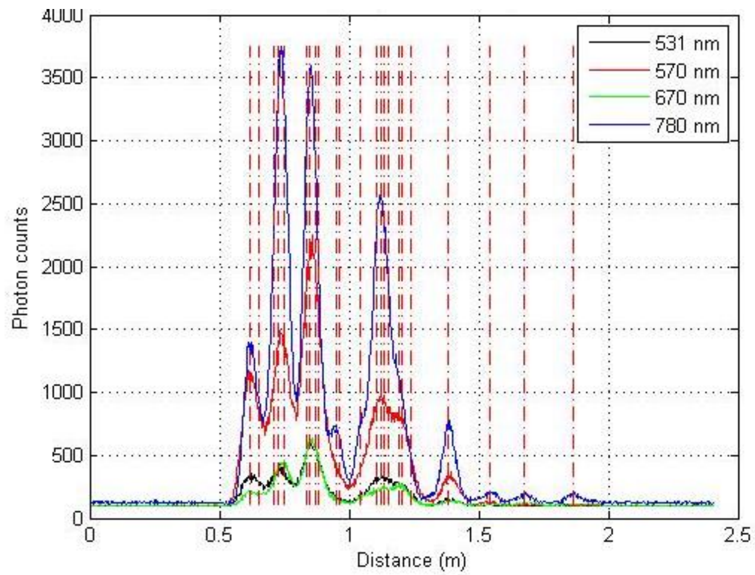
Function	*CPU time(ms)*	*GPU time(ms)*	*SpeedUp ratio*
Detection	48.81	8.43	5.8
Likelihood	30.95	13.60	2.3
Fusion	–	0.09	–
Resampling	0.44	0.56	0.8
Transition	2.11	0.37	5.7
Remove	–	0.02	–
Add	0.02	0.31	0.1
TextureUpdate	–	0.30	–
Total[1]	82.3	23.2	3.5

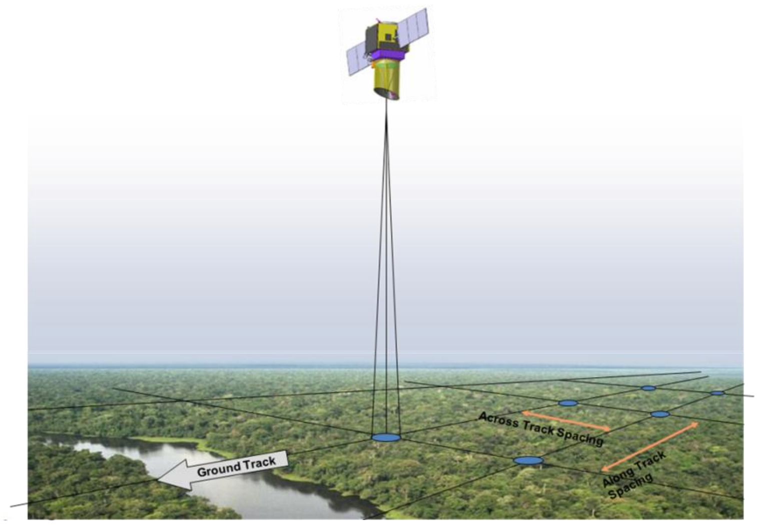


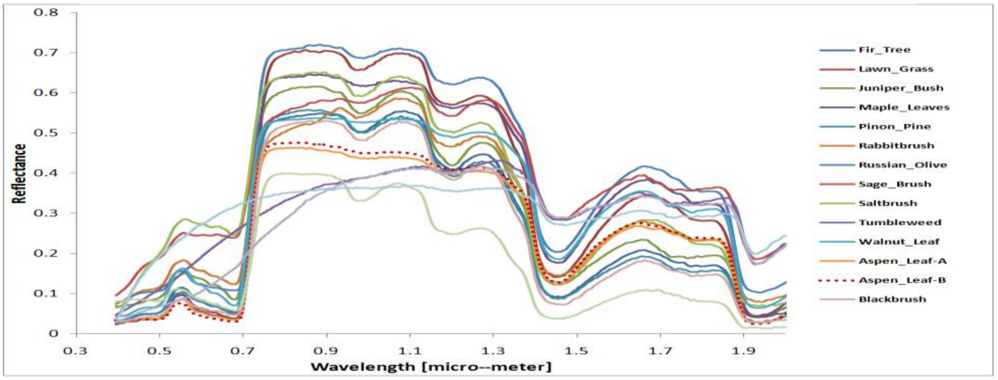
W Limprasert, AM Wallace and G Michaelson. "Real-time People Tracking in a Camera Network", IEEE Journal on Circuits and Systems, 263-271 June 2013

What kind of Dynamic Imaging? Sensing Forests: Multispectral LiDAR

Using Multi- or Hyper-spectral Lidar, it is possible to sense a single footprint, or build a 3D image of the scene below. This presents challenges to spectrally unmix pixels and images, such that structure, materials and material variation can be inferred.







AM Wallace, A McCarthy, C Nichol, X Ren, S. Morak[2], D Martinez-Ramirez, I. H. Woodhouse and GS Buller, “Design and of Evaluation of Multi-spectral LiDAR for the Recovery of Arboreal Parameters” IEEE Transactions on Geoscience and Remote Sensing, 52(8), 4942-4954, 2014

What’s the problem?

Code development for parallel or heterogeneous architectures …. e.g. it took several months (and inevitable algorithmic changes) to hand-craft Beowulf code to analyse (RJMCMC) full waveform multispectral LiDAR for tree canopy data, to recover structure and physiology.

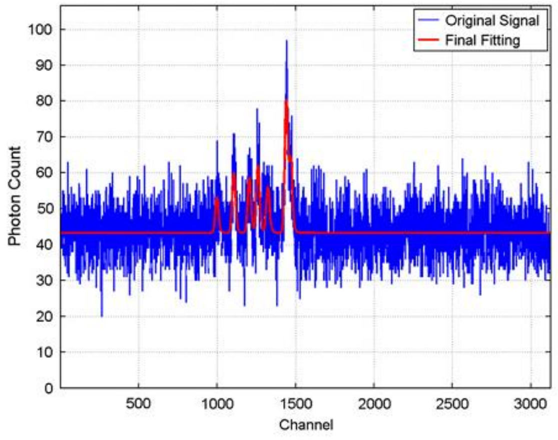
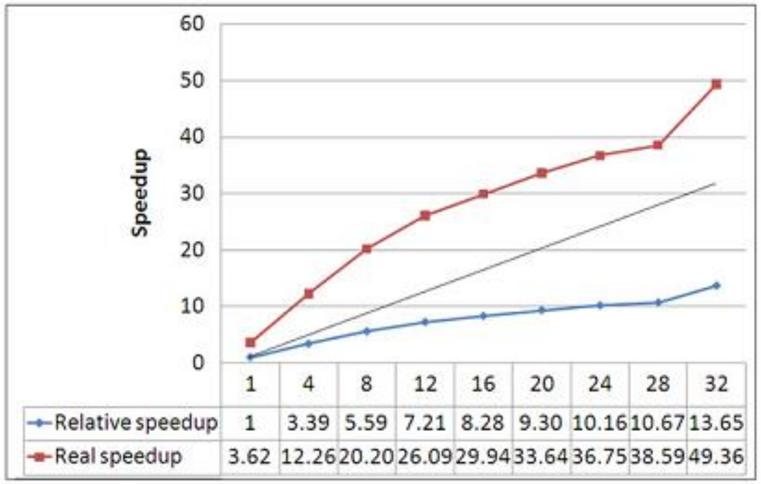


Fig. 17. Final fitting result of the real data (shown in Fig. 4) using DP SSD-RJMCMC.

Single footprint data



	1	4	8	12	16	20	24	28	32
Relative speedup	1	3.39	5.59	7.21	8.28	9.30	10.16	10.67	13.65
Real speedup	3.62	12.26	20.20	26.09	29.94	33.64	36.75	38.59	49.36

J Ye, AM Wallace A Al Zain and J Thompson, Parallel Bayesian Inference of Range and Reflectance from LaDAR Profiles, Journal of Parallel and Distributed Computing, 73(4), 383–399, 2013.

ECIT The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

What’s the problem? Simple HOG on an IpPro

Direct transformation to a custom FPGA, or (better) direct FPGA Coding of HOG using the IpPro (QUB) is laborious, and not necessarily optimal.

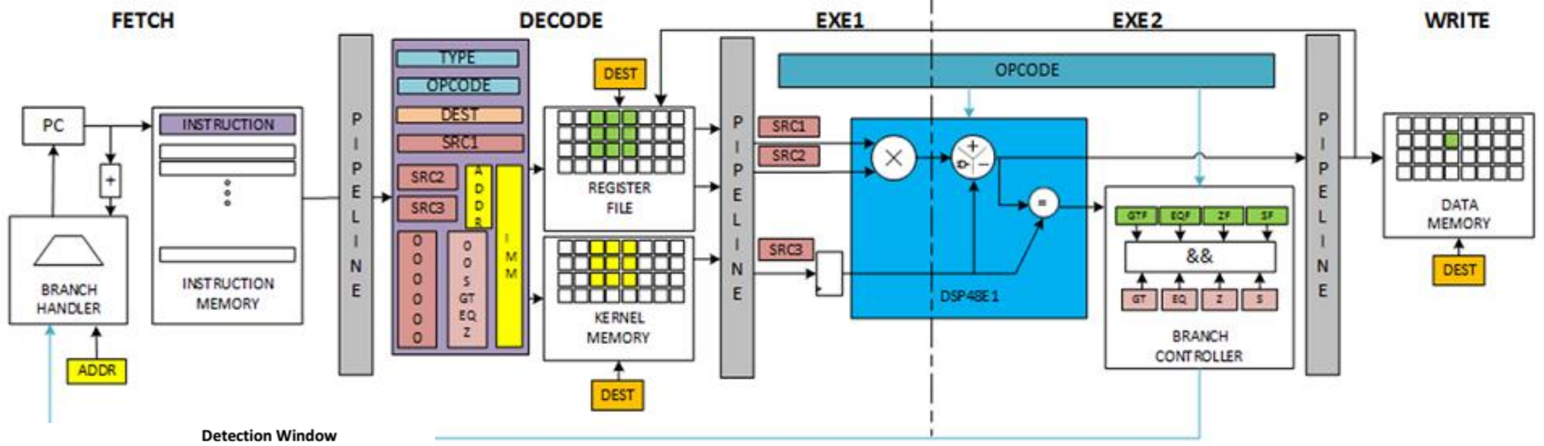


Fig. 2. IPPro Processor Datapath

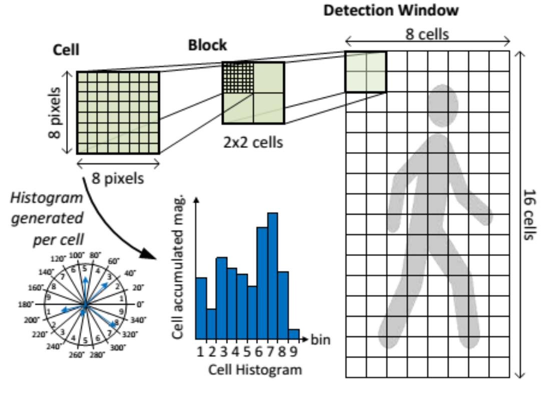


Fig. 2: Detection window divided into blocks and cells producing an array of histograms per window

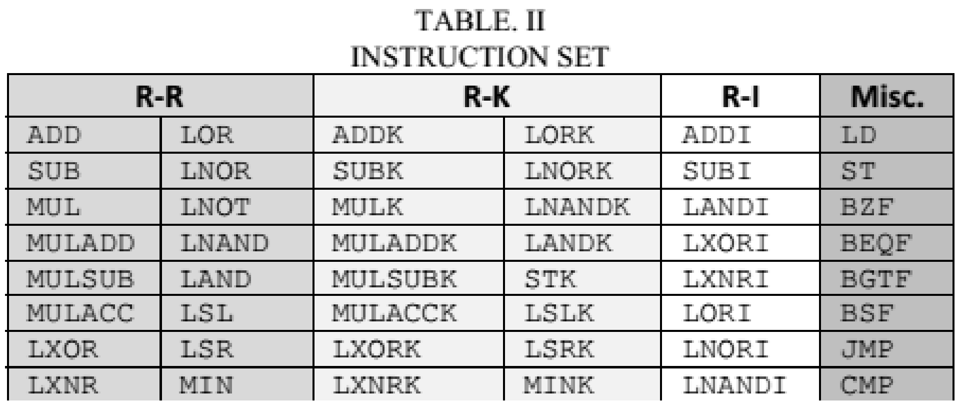


TABLE. II
INSTRUCTION SET

R-R		R-K		R-I	Misc.
ADD	LOR	ADDK	LORK	ADDI	LD
SUB	LNOR	SUBK	LNORK	SUBI	ST
MUL	LNOT	MULK	LNANDK	LANDI	BZF
MULADD	LNAND	MULADDK	LANDK	LXORI	BEQF
MULSUB	LAND	MULSUBK	STK	LXNRI	BGTF
MULACC	LSL	MULACCK	LSLK	LORI	BSF
LXOR	LSR	LXORK	LSRK	LNORI	JMP
LXNR	MIN	LXNRK	MINK	LNANDI	CMP

Fahad Manzoor Siddiqui, Matthew Russell, Burak Bardak, Roger Woods, Karen Rafferty IPPro: FPGA based Image Processing Processor, Proc GlobalSIP Conference 2014

Wouldn't it be nice if ……..

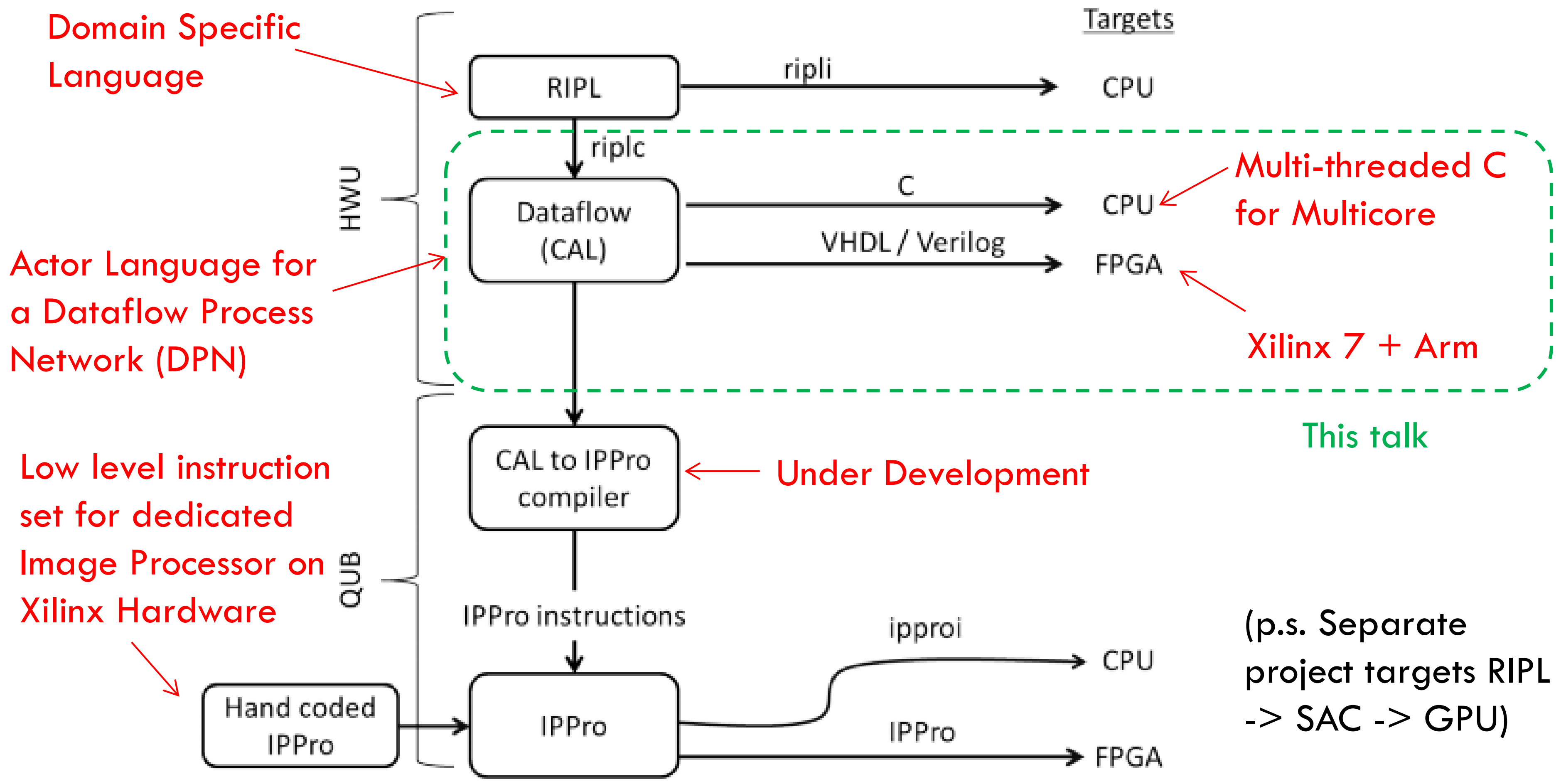
- We could express any given algorithm as high level abstractions, <u>drastically reducing code development time</u>, yet
- easily translate that code into executable code for a variety of parallel architectures, and
- transform that code (using either analysis or profiling) to optimise "performance", e.g.
  - ✓ … for <u>speed, memory use, power consumption, cost</u>, and
- either use a single platform, or mix and match processors (e.g. CPU, FPGA, GPU, IpPro) to meet the desired objectives, yet
- match or even better "hand-crafted" code

ECIT The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

Algorithm Development: the Rathlin Model



(p.s. Separate project targets RIPL -> SAC -> GPU)

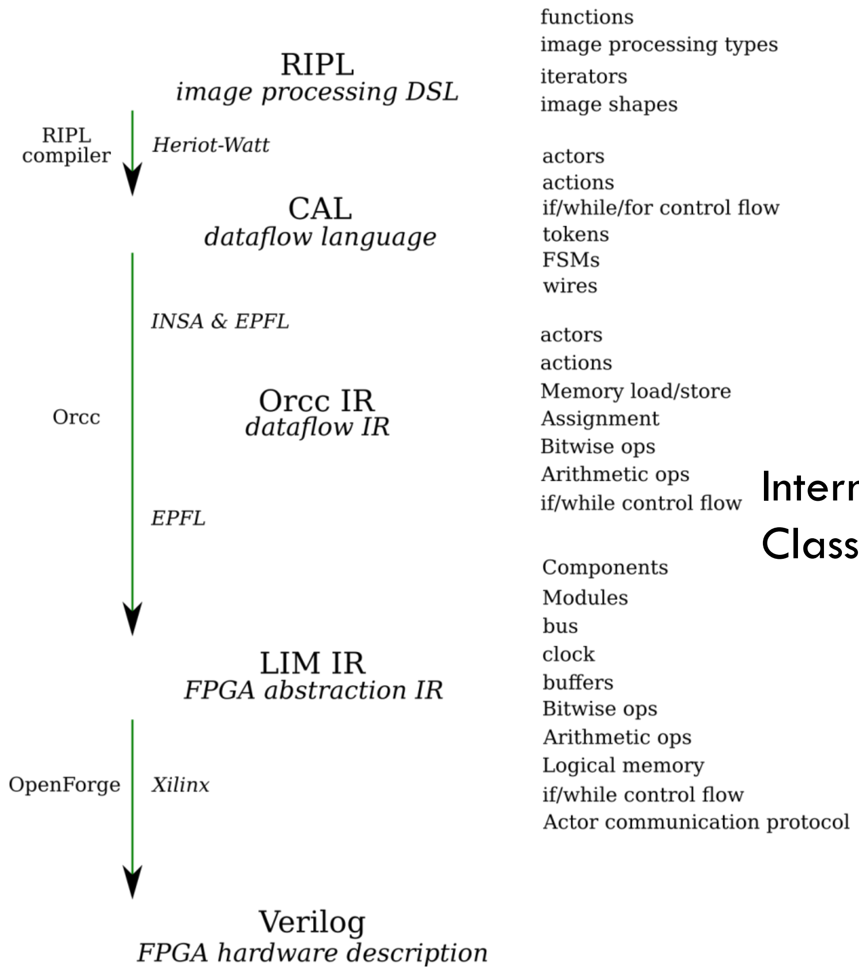
ECIT | The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

Compiler Flow for FPGA route: RIPL to CAL

- Inline all function calls into the main function.
- Replace all RIPL type declarations to CAL array declarations.
- Generate stream-based actors for each use of a RIPL iterator.
- Derive dataflow wires from implicit data dependencies between RIPL variables.
- Generate CAL files for each actor, where there is one actor per RIPL iterator.
- Generate an XML/XDF file for the wire connections.



Internal Java Classes

ECIT | The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

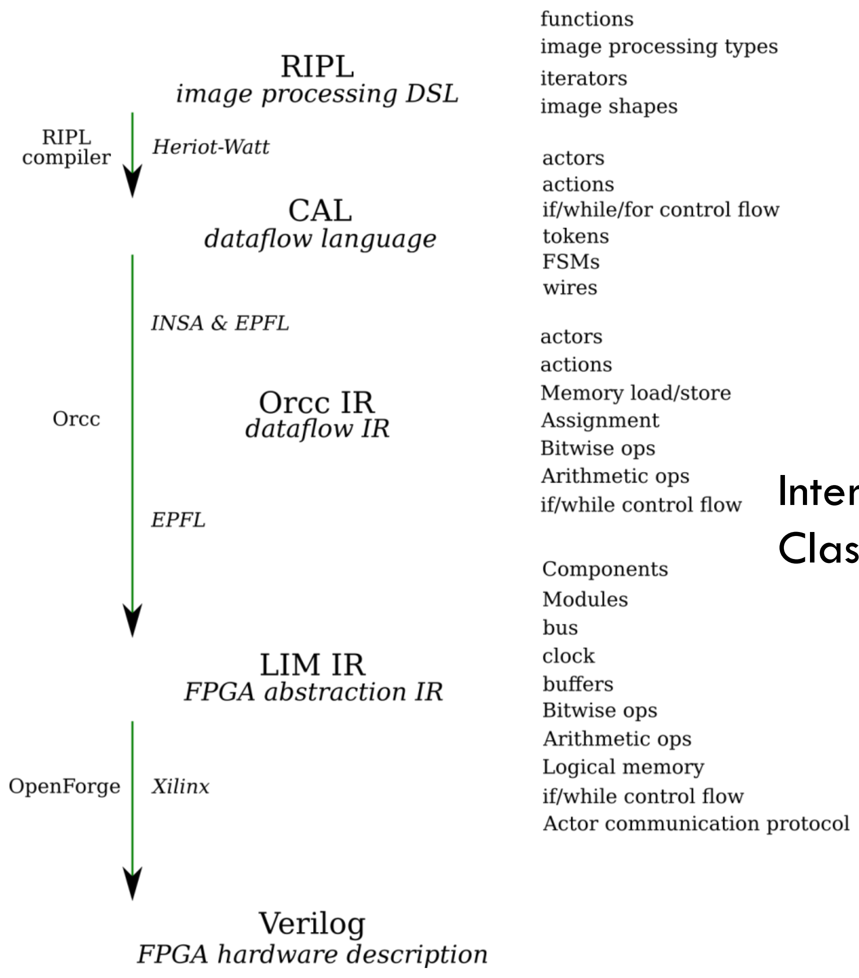
Compiler Flow: CAL to Verilog

The Orcc frontend parses CAL syntax into an abstract syntax tree (AST) which is then mapped to a dataflow IR.

The Xronos backend of the Orcc compiler generates Verilog for each actor, and a VHDL file that describes the network of wires between actors.

It does this by compiling the dataflow IR to a language independent model (LIM) IR which abstracts FPGA hardware.

OpenForge, open sourced by Xilinx, is used to compile LIM IR to Verilog.

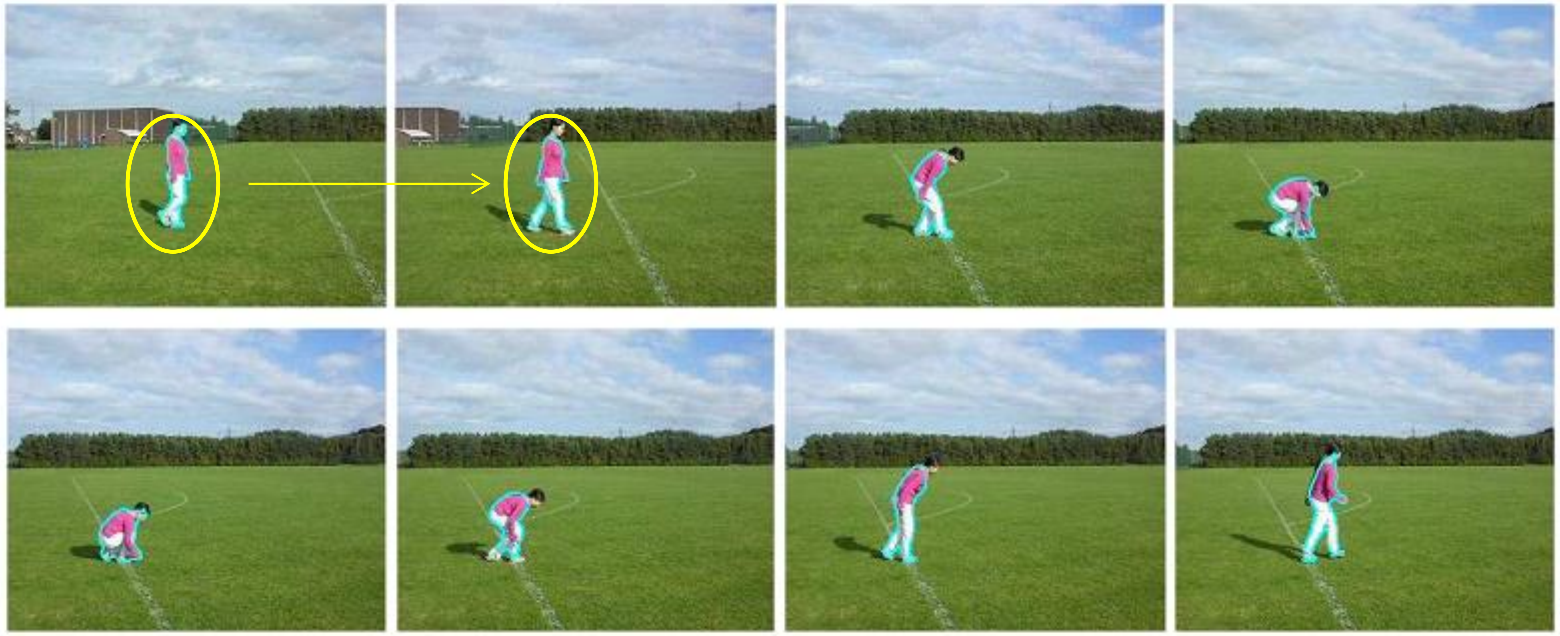


Internal Java Classes

Why DPNs?

- Image processing and analysis algorithms can be classified and developed in broad categories based on their algorithmic description:
  - point, local, global, temporal, adaptive or random.
- These classifications can be used to understand the hardware requirements and memory estimations.
- Early exposure to these requirements in dataflow representations can be used in optimisation, resource allocation and code synthesis.
- There is an established and active community working around the open source ORCC tools

Mean Shift Algorithm – Exemplar



Originally using a fixed template and a colour model with an Epanechnikov Kernel, this algorithm has been used many times (>8000 citations) and has been adapted in many ways for image segmentation and Object Tracking. For our purposes, we have several previous language (Matlab, C, Hume, Renesas) codes, there are a number of published hand-crafted FPGA implementations we can compare against, and it is challenging, because of the optimisation loop and necessary precision, but not impossible for the IpPro.

Comaniciu and Meer, IEEE Trans PAMI 24(5), Mean Shift: a robust approach towards feature space analysis, pp603-618, 2002

ECIT | The Institute of Electronics, Communications and Information Technology

EPSRC
Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

DPN Exemplar: Mean Shift Algorithm for Tracking

The basic approach is to create a probability density function (PDF) in frame (n+1), based on the colour histogram in frame (n) or a fixed model, and use an iterative procedure to find the maximum in this PDF that defines the new position of the object being tracked.

Usually, the PDF is based on the similarity between centre-weighted colour histograms, using an Epanechnikov kernel; the similarity function is usually defined from the Bhattacharya distance.

No. of bins in histogram

Bhattacharya distance

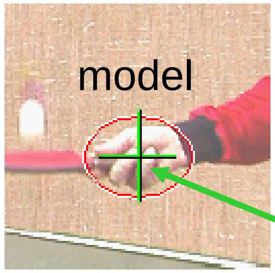
$$\rho[\hat{p}(y), q] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(y) q_u}$$

Model colour histogram

Target colour histogram
at position y

DPN Exemplar: Mean Shift Algorithm for Tracking

- The kernel is recursively moved from the starting position in the previous frame $\hat{y}_0$ to a new position $\hat{y}_1$ until convergence



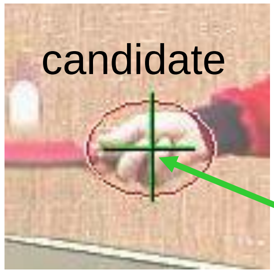
weights

Samples in PDF

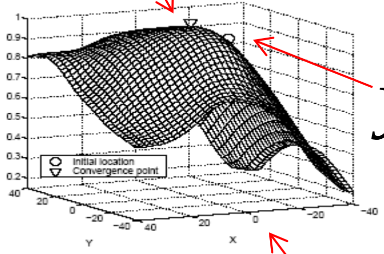
$$\hat{y}_1 = \frac{\sum_{i=1}^{n_h} x_i w_i g\left(\left\|\frac{\hat{y}_0 - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{\hat{y}_0 - x_i}{h}\right\|^2\right)}$$

$\hat{y}_0$

Window size



$\hat{y}_1$



$\hat{y}_0$

Similarity function

where $g(x) = -k'(x).$

E-kernel

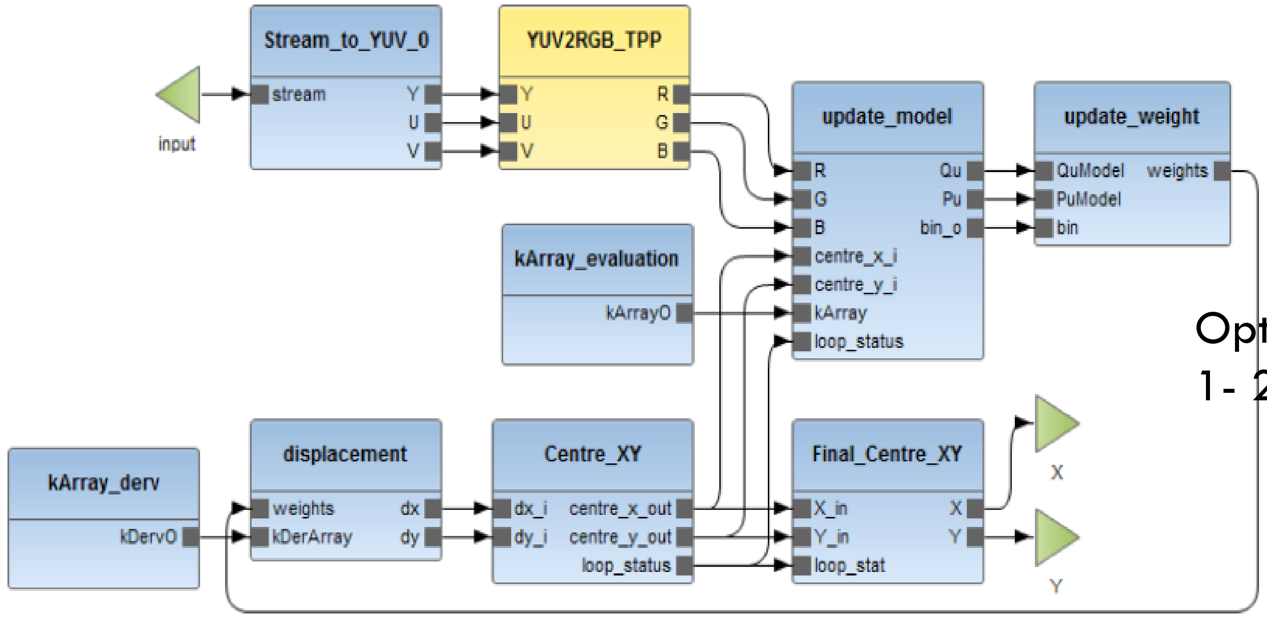
and $w_i = \sum_{i=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{y}_0)}} \delta[b(x_i) - u]$

Mean shift tracking algorithm

```
**Given** object position $y_0$ in frame n
Compute Epanechnikov kernel
Compute object colour model, $q_u(y_0)$
Repeat
    Read next frame (n+1)
    Compute object candidate model $p_u(y_0)$
    Compute similarity function, $p(y)$
    Repeat
        Derive weights $w_i$ for each pixel in candidate window
        Compute new candidate position, $y_1$
        Evaluate similarity function, $p(y)$
    Until $|y_1 - y_0| <$ € (near zero) or oscillatory or limit
Until end of sequence
```

What does the CAL Program look like (1)?

First, there is a Dataflow Process Network consisting of several actors – normally encoded with a XDF file that defines the connectivity and parameter passing between the several actors.



Optimisation Loop:
1- 20 iterations

See ORCC: Dataflow Programming made easy - http://orcc.sourceforge.net/

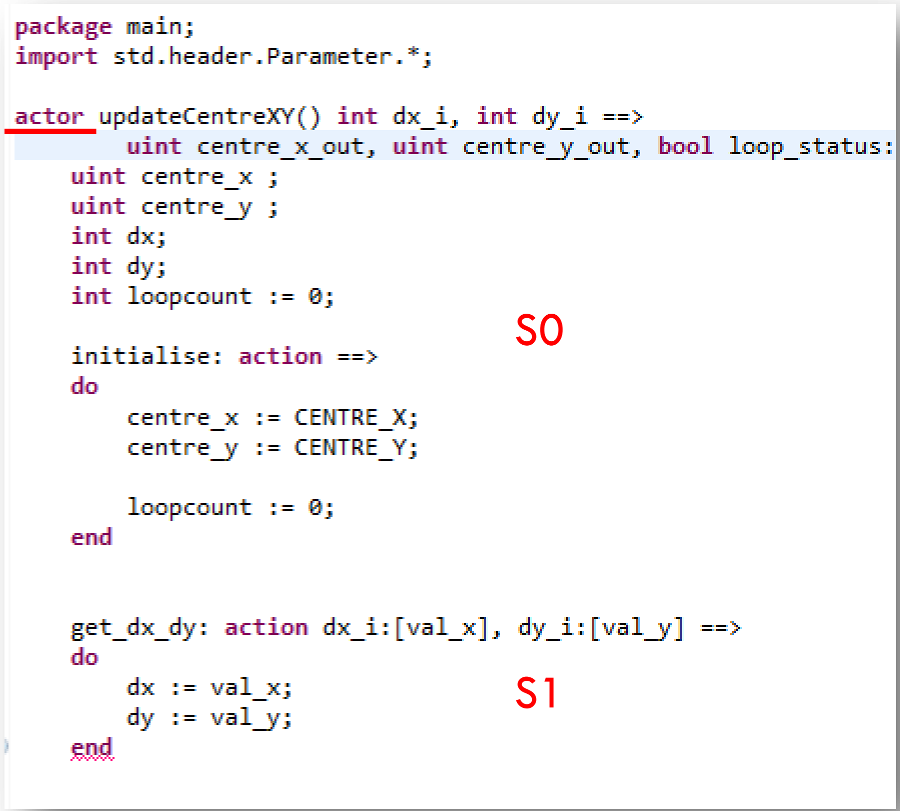
ECIT | The Institute of Electronics, Communications and Information Technology

EPSRC
Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

What does the CAL Program look like (2) ?

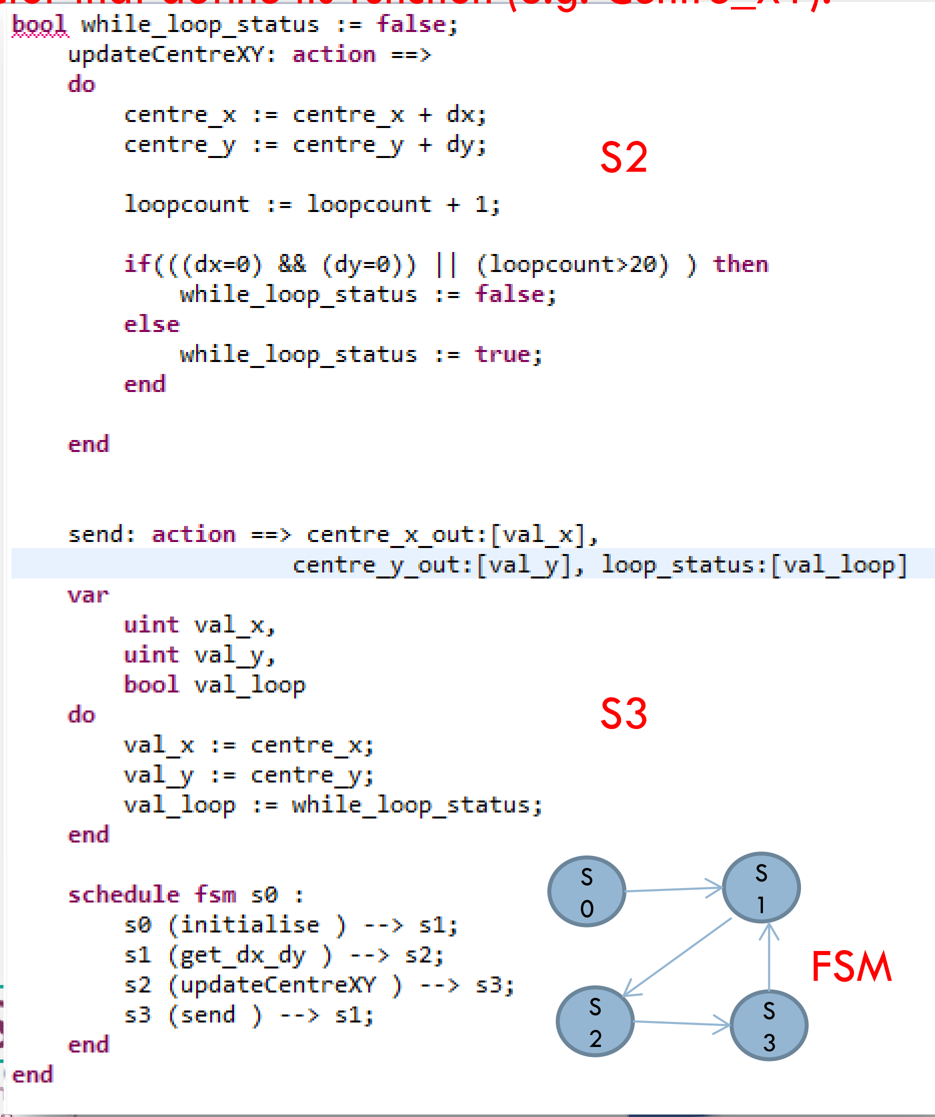
Second, there are CAL statements within each actor that define its function (e.g. Centre_XY).



```
package main;
import std.header.Parameter.*;

actor updateCentreXY() int dx_i, int dy_i ==>
        uint centre_x_out, uint centre_y_out, bool loop_status:
    uint centre_x ;
    uint centre_y ;
    int dx;
    int dy;
    int loopcount := 0;

    initialise: action ==>
    do
        centre_x := CENTRE_X;
        centre_y := CENTRE_Y;

        loopcount := 0;
    end


    get_dx_dy: action dx_i:[val_x], dy_i:[val_y] ==>
    do
        dx := val_x;
        dy := val_y;
    end
```

S0

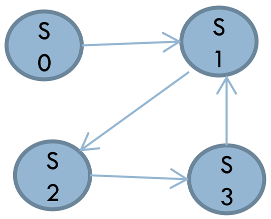
S1



```
bool while_loop_status := false;
    updateCentreXY: action ==>
    do
        centre_x := centre_x + dx;
        centre_y := centre_y + dy;

        loopcount := loopcount + 1;

        if(((dx=0) && (dy=0)) || (loopcount>20) ) then
            while_loop_status := false;
        else
            while_loop_status := true;
        end

    end


    send: action ==> centre_x_out:[val_x],
                        centre_y_out:[val_y], loop_status:[val_loop]
    var
        uint val_x,
        uint val_y,
        bool val_loop
    do
        val_x := centre_x;
        val_y := centre_y;
        val_loop := while_loop_status;
    end

    schedule fsm s0 :
        s0 (initialise ) --> s1;
        s1 (get_dx_dy ) --> s2;
        s2 (updateCentreXY ) --> s3;
        s3 (send ) --> s1;
    end
end
```

S2

S3



$$\begin{tikzcd} S0 \arrow[r] & S1 \arrow[dl] \\ S2 \arrow[r] & S3 \arrow[u] \end{tikzcd}$$

FSM

This Actor has four actions scheduled by a FSM

ECIT The Institute of Electronics, Communications and Information Technology

EPS Engineerin Research Council

Original Version: FPGA Synthesis of each Actor

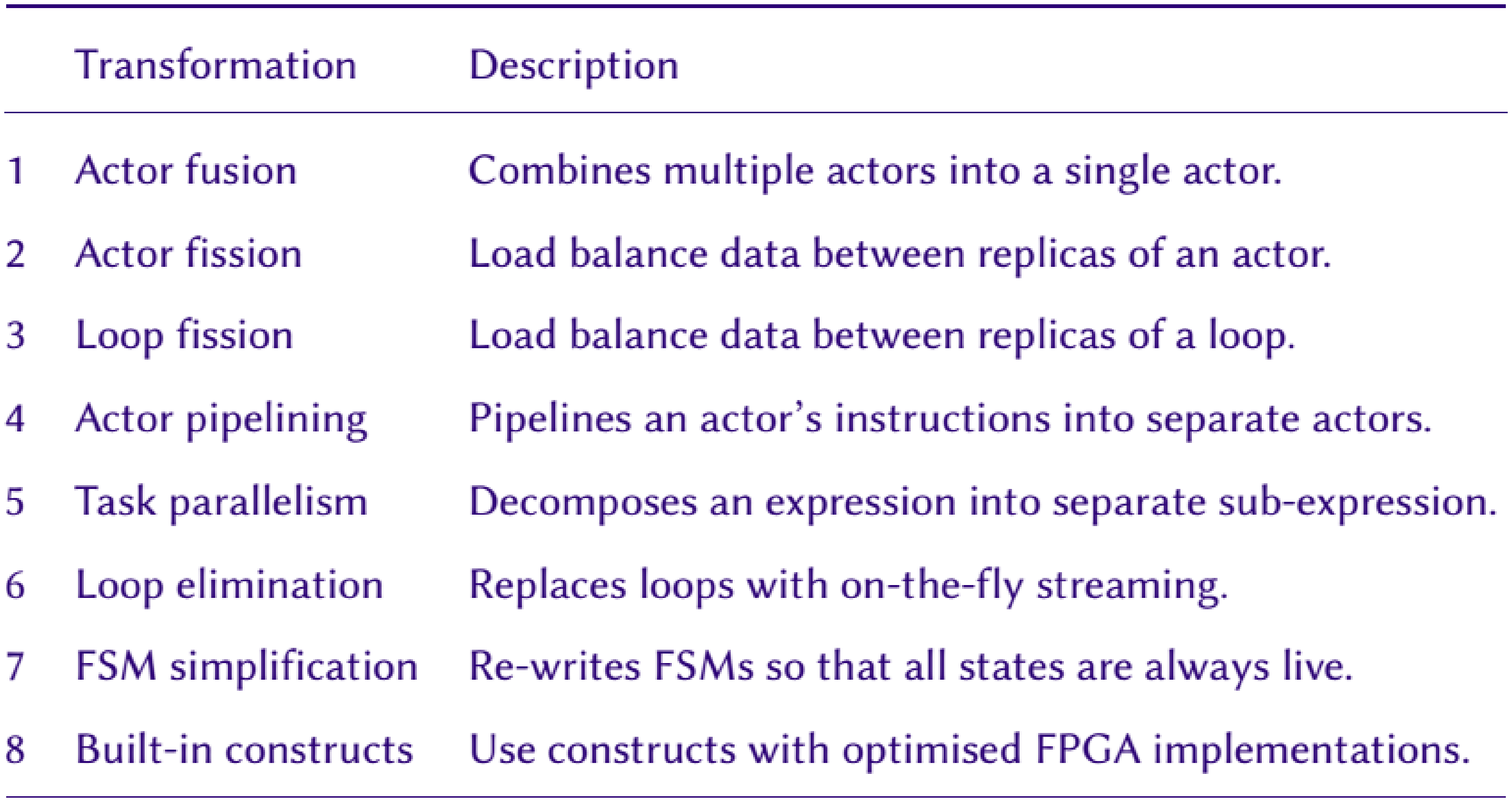
	Slice LUT	Slice registers	Block RAM /FIFO	DSP48E	FMax (MHz)
Naive	**3664**	**8777**	**88**	**49**	**55.41**
Final_XY	76	80	0	0	721.48
Centre_XY	182	199	0	0	530.81
Stream_to_YUV	90	287	24	0	420.07
update_model	1042	2399	30	0	148.74
YUV2RGB	300	957	7	0	126.71
displacement	545	1326	2	9	73.40
update_weight	556	1544	14	4	66.46
kArray_derv	437	1074	1	18	55.44
kArray_evaluation	460	1148	1	18	55.41

ECIT The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

DPN: Applying Transformations



	Transformation	Description
1	Actor fusion	Combines multiple actors into a single actor.
2	Actor fission	Load balance data between replicas of an actor.
3	Loop fission	Load balance data between replicas of a loop.
4	Actor pipelining	Pipelines an actor's instructions into separate actors.
5	Task parallelism	Decomposes an expression into separate sub-expression.
6	Loop elimination	Replaces loops with on-the-fly streaming.
7	FSM simplification	Re-writes FSMs so that all states are always live.
8	Built-in constructs	Use constructs with optimised FPGA implementations.

Other computational transformations are considered for FPGAs, notably Floating vs Fixed point implementation and the use of LUTs

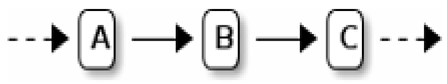
ECIT | The Institute of Electronics, Communications and Information Technology

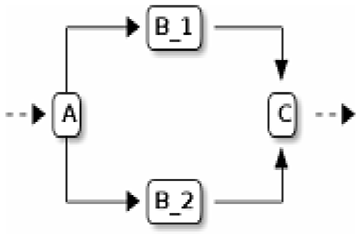
EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

Example 1: Data Parallelism & Actor Fission

Transformation 2 *Data parallelism with actor fission*





```
ACTOR A In ⟹ Out :
  action In:[x] ⟹ Out:[x]

ACTOR B In ⟹ Out :
  action In:[x] ⟹ Out:[F x]

ACTOR C In ⟹ Out :
  action In:[x] ⟹ Out:[x]
```

$\Longrightarrow$

```
ACTOR A A ⟹ O1,O2 :
  action In:[x1,x2] ⟹ O1:[x1], O2:[x2]

ACTOR B_1 A ⟹ Out :
  action In:[x] ⟹ Out:[F x]

ACTOR B_2 A ⟹ Out :
  action In:[x] ⟹ Out:[F x]

ACTOR C A ⟹ Out :
  action In:[x] ⟹ Out:[x]
```

ECIT The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

Actor Fission: applied to update weights

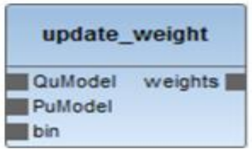
```
Update weights: action ⟹
Do
/* data parallelisable */
Foreach int I in 0 …. (NUMBINS) -1 do
     If (Pu_model_buffer[i] = 0) then
               R[i] := 0;
     Else
               sqrt ((Qu_model_buffer[i]/Pu_model_buffer[i])));
               R[i] := sqrtvalue;
     End
End
/* barrier necessary between two loops – not task parallelisable */
/* data parallelisable, but not cost-effective */
Foreach int x in 0 …. (X_SIZE-1) do
     Foreach int y in 0 …. (Y_SIZE-1) do
               weight_buffer[x][y] := R[bin_buffer{x][y]];
     End
End
End
```

ECIT | The Institute of Electronics, Communications and Information Technology

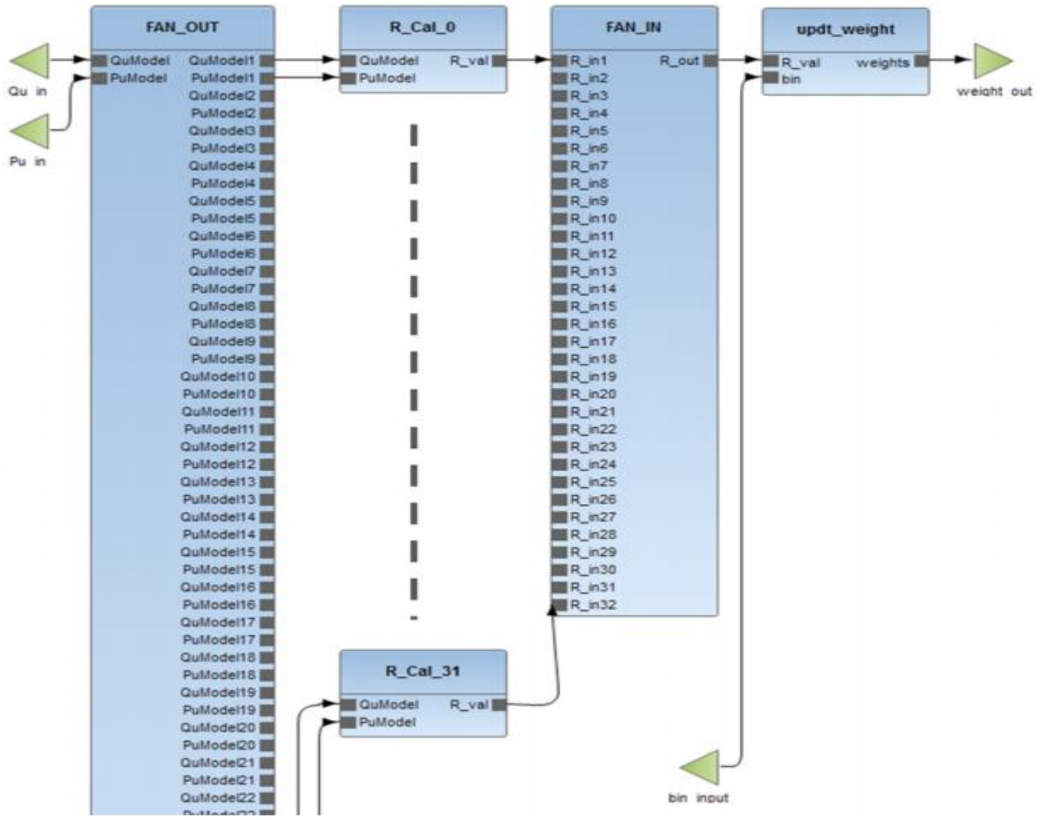
EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

Actor Fission: Update weight (Mean Shift)



$\Longrightarrow$



(a) before

(b) after

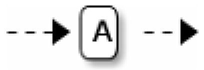
ECIT | The Institute of Electronics, Communications and Information Technology

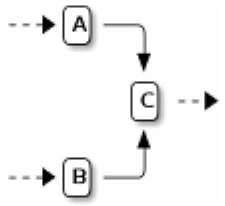
EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

Example 2: Task Parallelism

Transformation 5 *Decompose expression into task parallel actors*



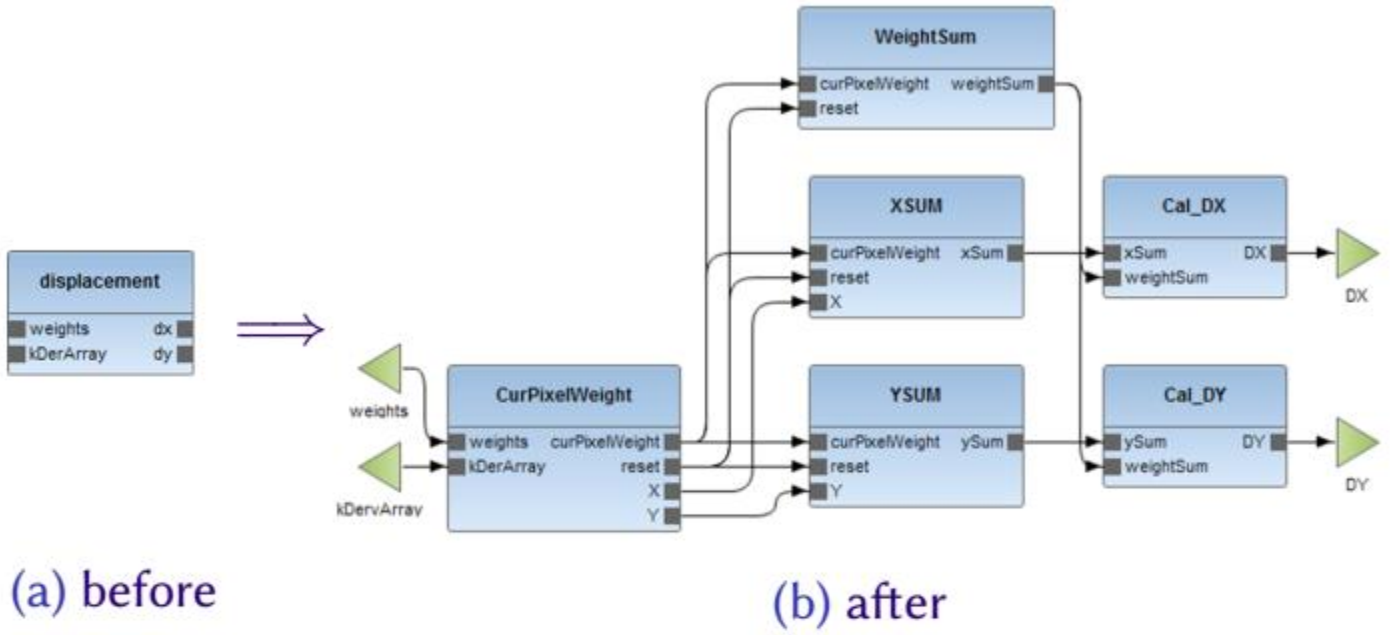


```
ACTOR A ⟹ Out :
  action ⟹ Out:[y]
    y := (e1 · e2 · e3)
```

$\Longrightarrow$

```
ACTOR A ⟹ Out :
  action ⟹ Out:[e1]

ACTOR B ⟹ Out :
  action ⟹ Out:[e2 · e3]

ACTOR C In1, In2 ⟹ Out :
  action In1[x], In2:[y] ⟹ Out:[x · y]
```

Task Parallelism: Displacement (Mean Shift)



(a) before

(b) after

Applying Transformations to Mean Shift

Functionality	Transformation	Registers	Slice LUTs	BRAM	DSP	Clock (MHz)
Stream to YUV	None	90	287	24	0	420.0
	Loop elimination	27	85	0	0	386.7
YUV to RGB	None	300	957	7	0	126.7
	Actor fusion	99	353	0	0	182.8
Displacement	None	545	1326	2	9	73.4
	Task parallelism	791	1210	7	9	110.0
Update weight	None	556	1544	14	4	66.5
	Fission	12352	19878	55	128	72.5
	Just square root (none)	346	548	0	4	72.5
	Square root Lookup	139	227	32	0	368.2
	Combined	7907	38544	1028	0	225.9
k-array derive	None	437	1074	1	18	55.4
	Loop promotion	4447	12484	5	144	52.7

But recall this is 'once only'

Final results: Mean Shift

	Slice LUT	Registers	BRAM	DSP48E	Clock (MHz)
Naive version	2751	6582	86	13	66.5
Optimising HDL for speed	2751	6635	86	13	66.5
Optimising HDL for area	2748	6610	87	13	66.5
Dataflow optimisations	10786	51267	1026	9	110.0

Table 4: Comparison of Dataflow and HDL Level Optimisation Results

Functionality	Transformation	Runtime	FPS
Naive version		2.97s	43.8
Updating the model	FSM simplification	2.06s	63.1
Displacement	Task parallelism	3.17s	41.0
Compute tracking window	Loop elimination	2.96s	43.9
RGB to YUV	Actor fusion	2.67s	48.7
k-array evaluation	Language use	2.34s	55.6
Combined		**1.70s**	**76.5**

Table 5: Transformation effects on CPU results

ECIT | The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

Conclusions (the story so far)

- We have applied DPN transformations to optimise algorithms expressed in the CAL dataflow language.
- This identifies transformations to target FPGAs; e.g. for Mean Shift, the overall clock frequency is increased from 66.5MHz to 110MHz.
- Applying all CPU targeting transformations increases mean throughput from 43fps to 77fps.
- In general, coding is (arguably) much simplified, e.g. a wavelet transformation is 4 lines of RIPL, 34 lines of CAL, and over 1000 lines of VHDL code.
- We have also developed an IpPro architecture, a partially reconfigurable soft core processor that will continue to evolve

ECIT The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY

Future Work

- A key priority is to embed the dataflow transformations as compiler optimisations guided by FPGA simulation and CPU traced-based profiling
- As the project develops, we hope to target the IpPro from both RIPL and Dataflow networks.
- We are developing concurrently new algorithms for dynamic video data analysis,
  - Random Finite Set approach to track multiple targets of two distinct types in clutter (e.g. pedestrians and vehicles, sheep and goats)
  - Crowd density and flow estimation techniques, that we hope to use to improve detection and tracking in sparse and dense populations

R. Stewart, D. Bhowmik, A Wallace, G Michaelson, Profile guided dataflow transformation for FPGAs & CPUs, IEEE Global Conference on Signal and Information Processing, December 2014 + Journal Submit.

ECIT | The Institute of Electronics, Communications and Information Technology

EPSRC Engineering and Physical Sciences Research Council

HERIOT WATT UNIVERSITY