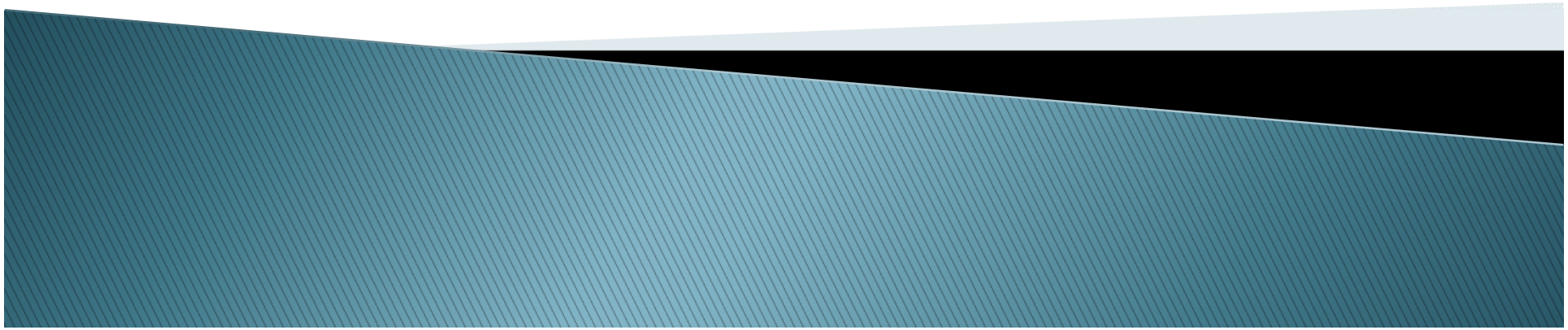
An Optimal Buffer Assignment Method for Streaming Applications on Hybrid Memory with Volatile and Non-volatile RAM

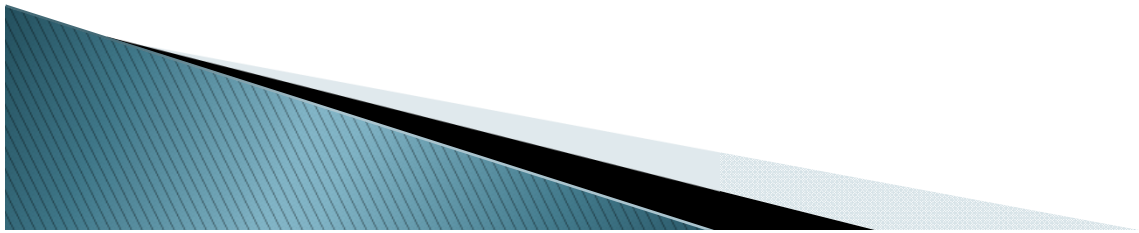
Hyunok Oh, Hanyang Univ.

2012.10.16

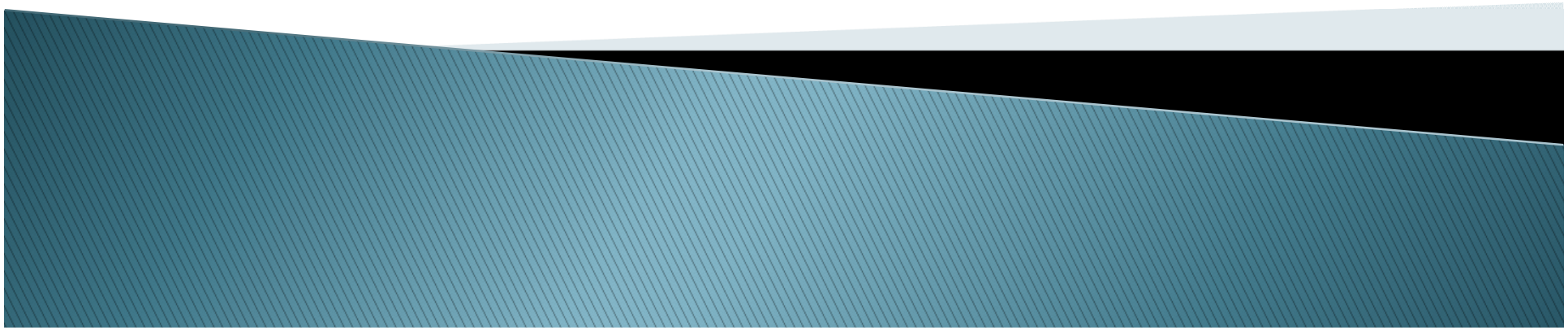


Outline

- Theoretical formulation
- Design tools

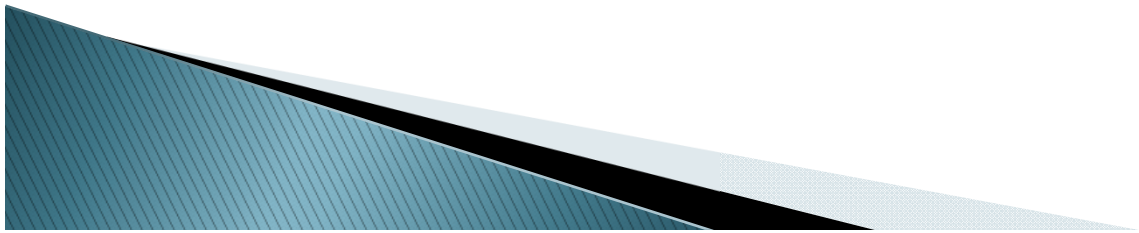


Theoretical Formulation



Outline

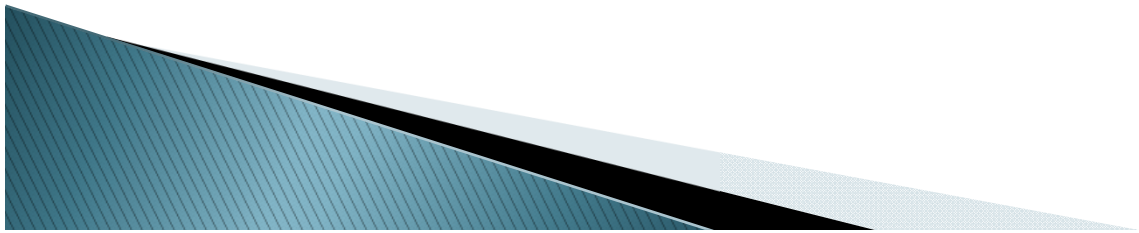
- **Introduction**
  - Non-volatile Memory (NVM)
  - Synchronous dataflow (SDF)
- **Motivational Example**
- **Problem Definition**
  - P1.Maximization of PRAM life-time
  - P2.Minimization of DRAM size
  - P3.Minimization of energy consumption
- **Answer Set Programming**
- **Experiment**
- **Conclusion**



Introduction

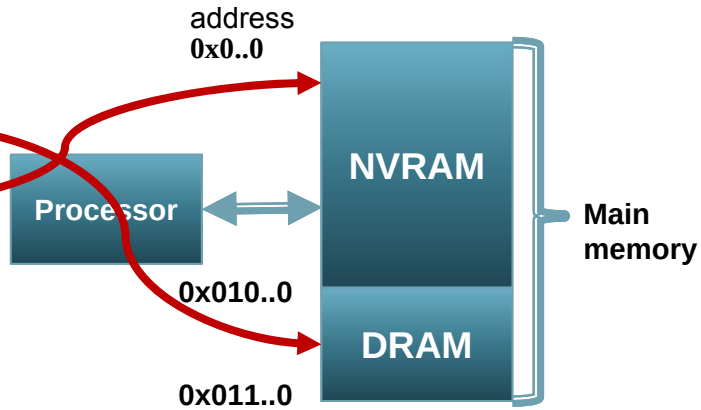
- **Non-Volatile Memory (NVM)**
  - Replace DRAM for main memory

Type	Phase change RAM (PRAM) Spin-transfer torque magneto resistive RAM (STT-MRAM) Ferroelectric RAM(FRAM)
Pros	High density **Low static energy consumption**
Cons	Long write latency **Short lifetime** (PRAM : 10^5 ~ 10^8 times)



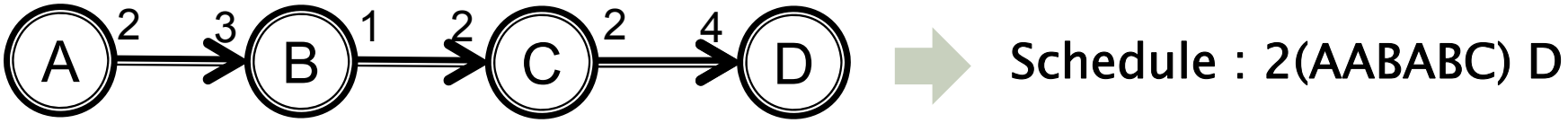
Introduction

- **Hybrid memory architecture – DRAM/NVRAM**
  - Distributed data allocation
    - DRAM: frequently updated data
    - NVRAM: frequently read data
  - Increase lifetime of NVRAM
  - Reduce energy consumption



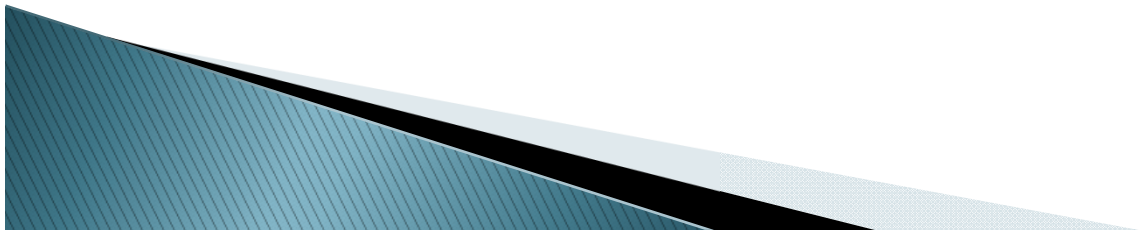
Introduction

- **Synchronous dataflow (SDF)**
  - represents streaming applications like multimedia that require frequent memory access
  - Node(Actor) - functional algorithm
  - Edge - communication between two actors
  - Producing / Consuming rate
    - the number of produced and consumed samples
  - Rate is fixed

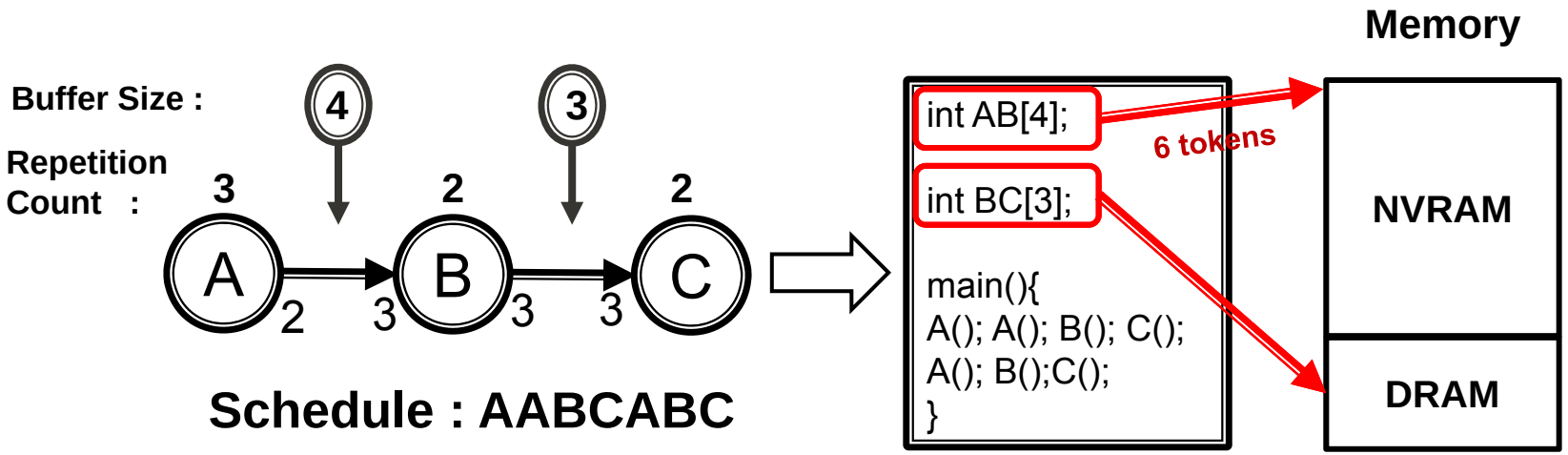


Outline

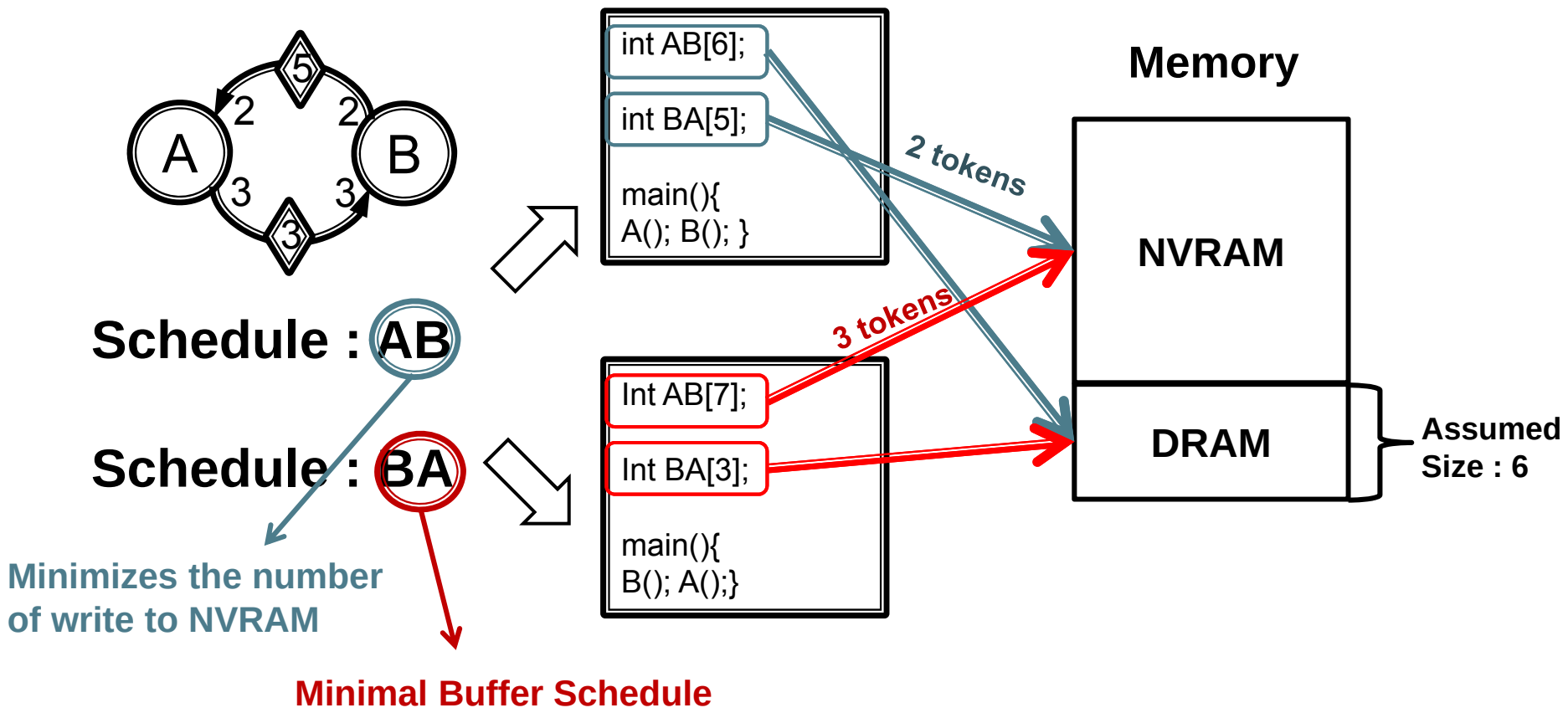
- Introduction
  - Non-volatile Memory (NVM)
  - Synchronous dataflow (SDF)
- **Motivational Examples**
- Problem Definition
  - P1.Maximization of PRAM life-time
  - P2.Minimization of DRAM size
  - P3.Minimization of energy consumption
- Answer Set Programming
- Experiment
- Conclusion



Motivational Example

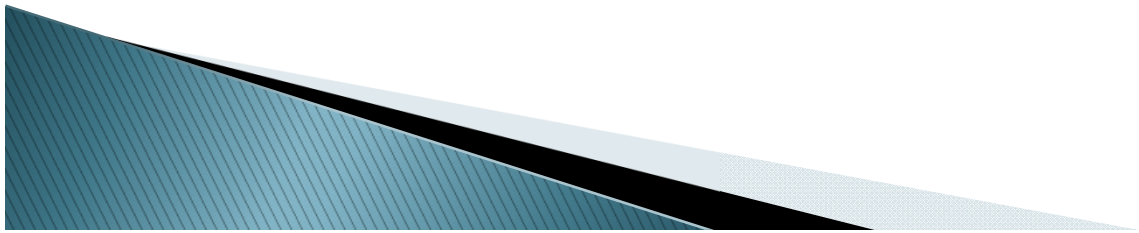


Motivational Example



Outline

- Introduction
  - Non-volatile Memory (NVM)
  - Synchronous dataflow (SDF)
- Motivational Example
- **Problem Definition**
  - P1.Maximization of NVRAM life-time
  - P2.Minimization of DRAM size
  - P3.Minimization of energy consumption
- Answer Set Programming
- Experiment
- Conclusion

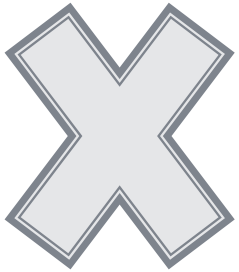


Problem Definition

(1) Maximize the lifetime of NVRAM memory

(2) Minimize the DRAM memory size

(3) Minimize the energy consumption



(a) For a given schedule

(b) By constructing an optimal schedule.

P1.Maximization of PRAM Life-time

- **Formulation for given schedule**

$$\text{Minimize} \sum_{e_i \in E} p_i * s_i$$

The sum of the produced tokens assigned to NVRAM

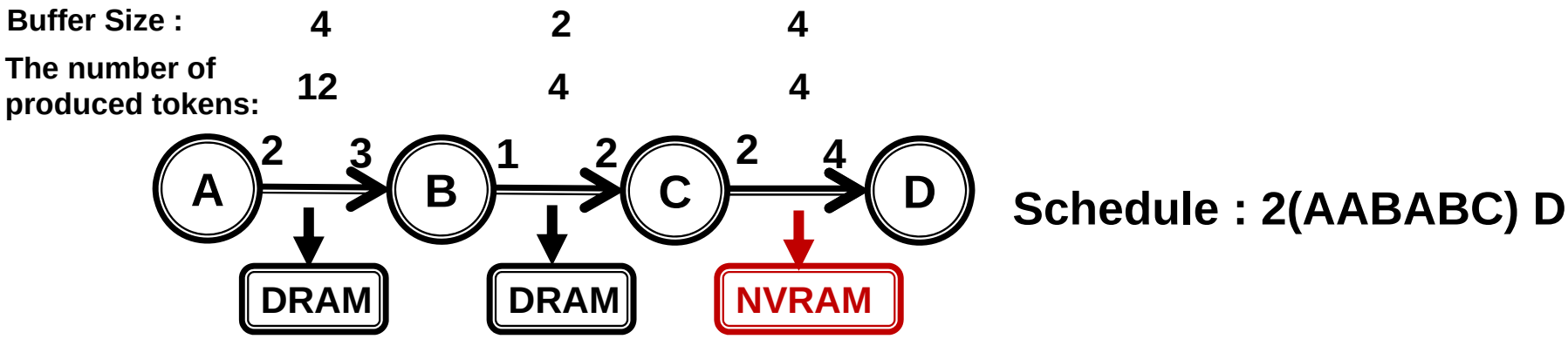
$$\text{Subject to} \sum_{e_i \in E} d_i * b_i \leq S_{DRAM}$$

Total sum of the buffer size assigned to DRAM

Constraint	Meaning
p_i	Binary variable to indicate whether the buffer on edge is assigned to NVRAM or not (p_i = 1- d_i)
s_i	The number of produced tokens on edge per iteration
b_i	Binary variable to indicate whether the buffer on edge is assigned to DRAM or not (It is 1 if assigned to DRAM)
d_i	The buffer size on edge
E	A set of edges

P1.Maximization of PRAM Life-time

- Example



  - Minimize $(12p_{AB}+4p_{BC}+4p_{CD})$
  - Subject to $(4d_{AB}+2d_{BC}+4d_{CD}) \leq 6\ (S_{DRAM})$
- $p_e + d_e$ (A buffer on each edge is assigned to either NVRAM or DRAM)

P1.Maximization of NVRAM Life-time

- Constructing an optimal schedule
  - Computes buffer size on each edge for each schedule to obtain an optimal schedule
  - Compared to previous example's optimal schedule
    - 2(3A2B)2CD - the number of produced tokens on NVRAM is **8**
      - Subject to ($\mathbf{6}\boldsymbol{d_{AB}}+2d_{BC}+4d_{CD}$) ≤ 6
    - 6A 4B 2C D - the number of produced tokens on NVRAM is **16**
      - Subject to ($12d_{AB}+\mathbf{4}\boldsymbol{d_{BC}}+4d_{CD}$) ≤ 6
    - **2(AABABC)D** - the number of produced tokens on NVRAM is **4**
    - Optimal schedule - 2(AABABC)D
- It is applied to P2 and P3

P2.Minimization of DRAM Size

- Revise the endurance maximization problem
  - Minimize $\sum_{e_i \in E} d_i * s_i$

    The sum of the buffer size assigned to DRAM
  - Subject to $\sum_{e_i \in E} p_i * s_i \leq \frac{P * R * S_{PRAM}}{LT}$

    Total sum of the produced tokens assigned to NVRAM

    A given minimum endurance of NVRAM

Constraint	Meaning
P	The period of an application
R	The endurance or the maximum reliable number of write operations per NVRAM cell.
LT	Lifetime of a system to be guaranteed

P3.Minimization of Energy Consumption

- Similarly to the minimization of DRAM Size
- Minimize $E_{NVRAM} \sum_{e_i \in E} p_i * s_i + E_{DRAM} \sum_{e_i \in E} d_i * s_i + S_{DRAM} * SE_{DRAM}$

Total energy consumption

Dynamic energy consumption on NVRAM

Dynamic energy consumption on DRAM

Static energy consumption on DRAM

- Subject to $\sum_{e_i \in E} p_i * s_i \le \frac{P * R * S_{PRAM}}{LT}$, $\sum_{e_i \in E} d_i * b_i \le S_{DRAM}$

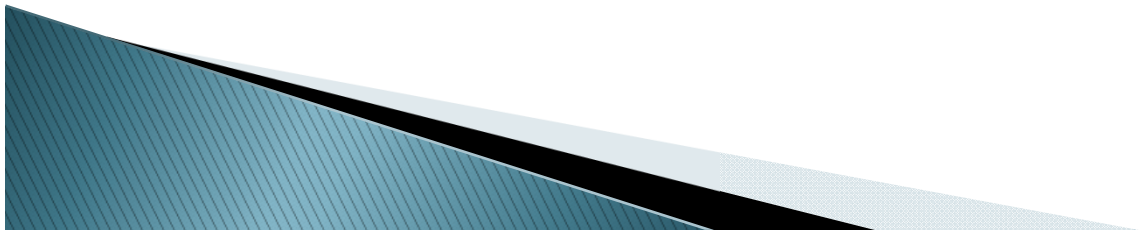
NVRAM endurance

DRAM Size

Constraint	Meaning
E_{NVRAM} , E_{DRAM}	Sum of energy consumptions of byte read and byte write per second on NVRAM, DRAM
SE_{DRAM}	Static energy consumption per byte on DRAM

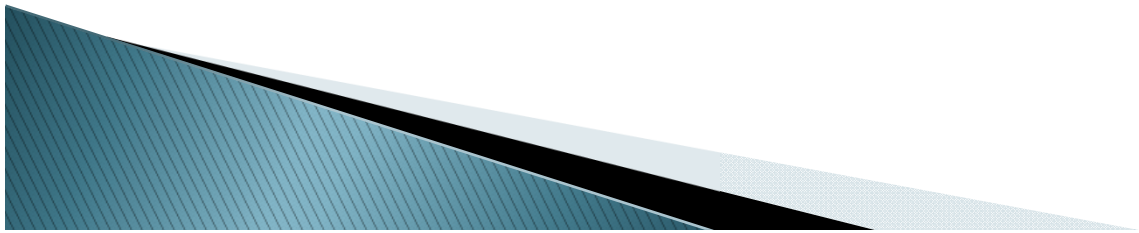
Outline

- Introduction
  - Non-volatile Memory (NVM)
  - Synchronous dataflow (SDF)
- Motivational Example
- Problem Definition
  - P1.Maximization of PRAM life-time
  - P2.Minimization of DRAM size
  - P3.Minimization of energy consumption
- **Answer Set Programming**
- Experiment
- Conclusion

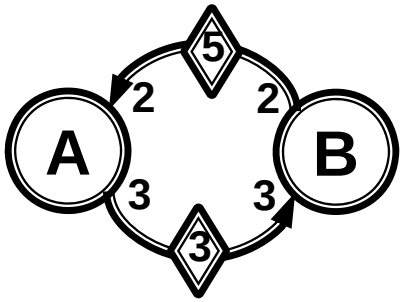


Answer Set Programming (ASP)

- **Declarative approach for NP problems**
- Problem - logic predicates **“ AND”**
- Solutions - answer sets
- Easy to understand the formulation
- Fast ASP solvers have been introduced



Problem Solving Flow



node(1..2).
edge(1,1,2,3,3,3).edge(2,2,1,2,2,5).
repetition(1,1).repetition(2,1).

SDF Graph

```
#begin_lua
function gcd(a,b)
if a==0 then return b
else    return gcd(b%a, a)
end
end
#end_lua.
minimumBufSize(E,M) :- edge(E,A,B,P,C,I), M=P+C-@gcd(P,C).
bufSize(E,U) :- U = #max[sample(E,S,_)=S, minimumBufSize(E,M)=M], edge(E,_,_,_,_,_).
numSamples(E,S) :- S=P*X+I, edge(E,A,B,P,C,I), repetition(A,X).
0 { dram(E) : edge(E,_,_,_,_,_) }.
dramSum(E,S) :- dram(E), bufSize(E,S).
sample(E,I,0) :- edge(E,_,_,_,_,I).
sample(E,S+P,T) :- fire(A,T), edge(E,A,_,P,C,I), sample(E,S,T-1), time(T).
sample(E,S-C,T) :- fire(B,T), edge(E,_,B,P,C,I), sample(E,S,T-1), S>= C, time(T).
sample(E,S,T) :- sample(E,S,T-1), not fire(A,T), not fire(B,T), edge(E,A,B,_,_,_), time(T).
```

```
#sum[dramSum(E,S)=S : edge(E,_,_,_,_,_) : S=1..M : numSamples(E,M)] dramSize.
#maximize [dram(E)=N : numSamples(E,N)].
```

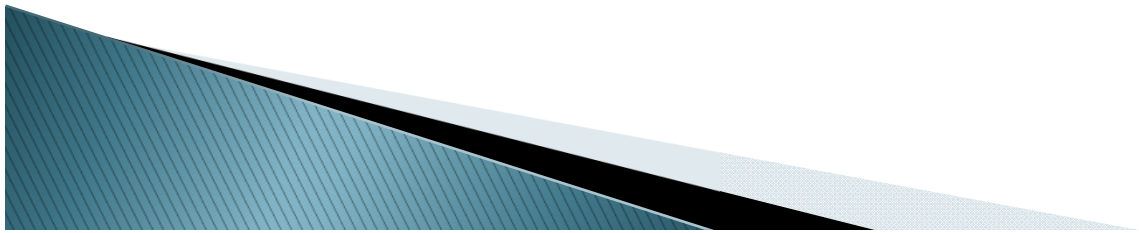
ASP Formulation

```
Answer: 1

bufSize(2,5),bufSize(1,6) numTotalSamplesOnPRAM(13)

Optimization: 13

Answer: 2

bufSize(2,5),bufSize(1,6)

dram(1) numTotalSamplesOnPRAM(7)

Optimization: 7

Answer: 3

bufSize(2,5),bufSize(1,6)

dram(2) numTotalSamplesOnPRAM(6)

Optimization: 6

OPTIMUM FOUND

Models      : 1

 Enumerated: 3

 Optimum    : yes

Optimization: 6
```

Result

Outline

- Introduction
  - Non-volatile Memory (NVM)
  - Synchronous dataflow (SDF)
- Motivational Example
- Problem Definition
  - P1.Maximization of PRAM life-time
  - P2.Minimization of DRAM size
  - P3.Minimization of energy consumption
- Answer Set Programming
- Experiment
- Conclusion

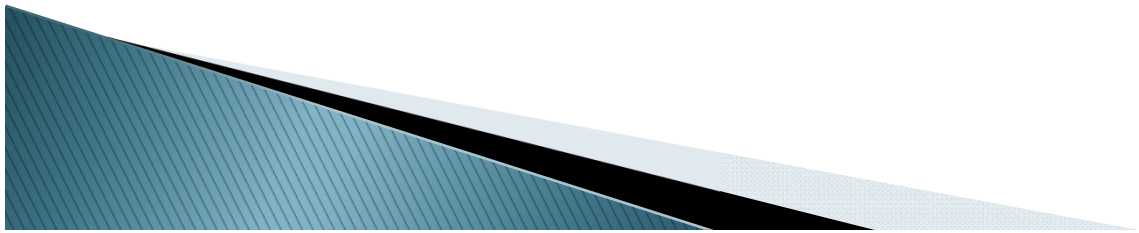


Experiments

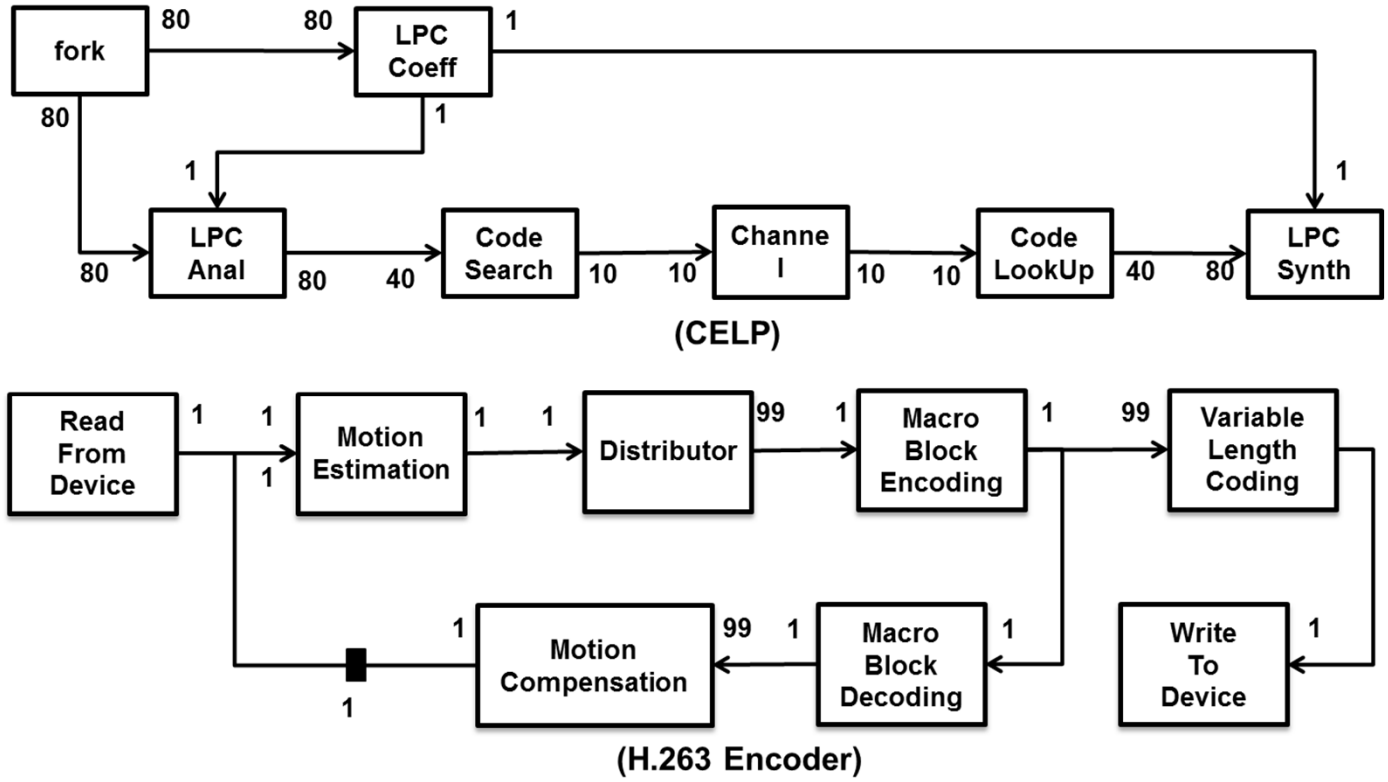
- Synthetic examples
- Real life applications
  - Part of CELP
  - H.263 encoder / decoder
  - MP3 decoder

CPU	Intel i7 2.7Ghz
RAM	6GB
OS	Ubuntu Linux
ASP Solver	Clingo 3.0

- 25 randomly generated examples
  - 3 to 6 actors
  - 1 to 3 repetition per actor



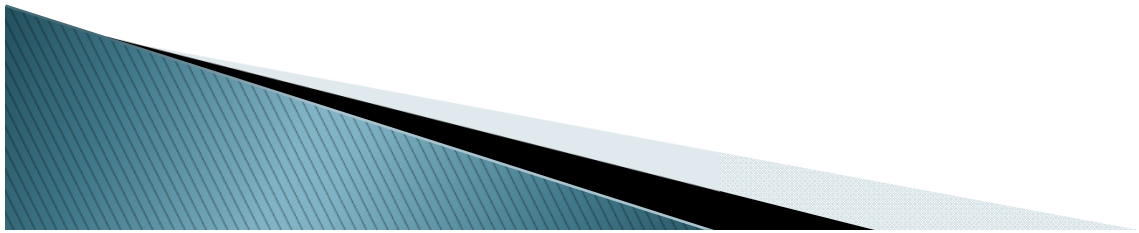
SDF Graphs


fork
LPC Coeff
LPC Anal
Code Search
Channe l
Code LookUp
LPC Synth
80
80
1
80
1
1
1
80
80
40
10
10
10
10
40
80
(CELP)
Read From Device
Motion Estimation
Distributor
Macro Block Encoding
Variable Length Coding
Motion Compensation
Macro Block Decoding
Write To Device
1
1
1
1
1
1
99
1
1
99
1
1
1
1
1
99
1
1
(H.263 Encoder)

PRAM Lifetime Maximization
for Real Life Applications

Graph	DRAM size	C1	C2	C3	C2 time	C3 time	(C1-C2)/C2
CELP	150	280	240	240	0.02	0.02	**0.16**
H.263 Encoder	150	523	396	396	0.42	30.82	**0.32**
H.263 Decoder	10	35	22	22	0.18	1.68	**0.59**
MP3	15	24	11	11	0.04	0.5	**1.18**

- The number of produced tokens
  - C1- Non-optimized approach with a given minimal buffer schedule
  - C2- Lifetime maximization with a given minimal buffer schedule
  - C3- Lifetime maximization with constructing an optimal schedule
- C2,C3 results
  - Acceptable to use existing scheduling algorithms to construct a schedule rather than constructing the optimal schedule
- Improvement of NVRAM lifetime by **63%** compared with a non-optimized algorithm



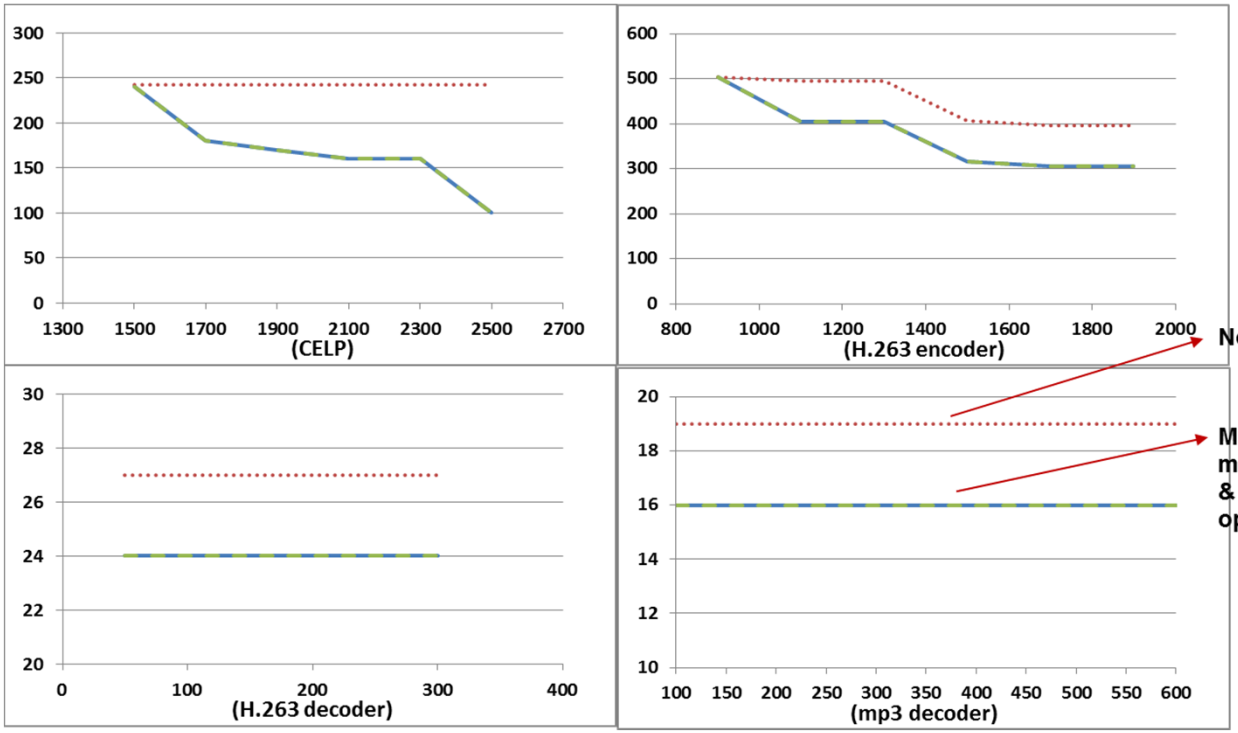
PRAM Lifetime Maximization

for 25 Randomly Generated Examples

	(C1-C2)/C2	(C2-C3)/C3
Max. Lifetime	48.7%	0%
Min. DRAM Size	100.0%	0%

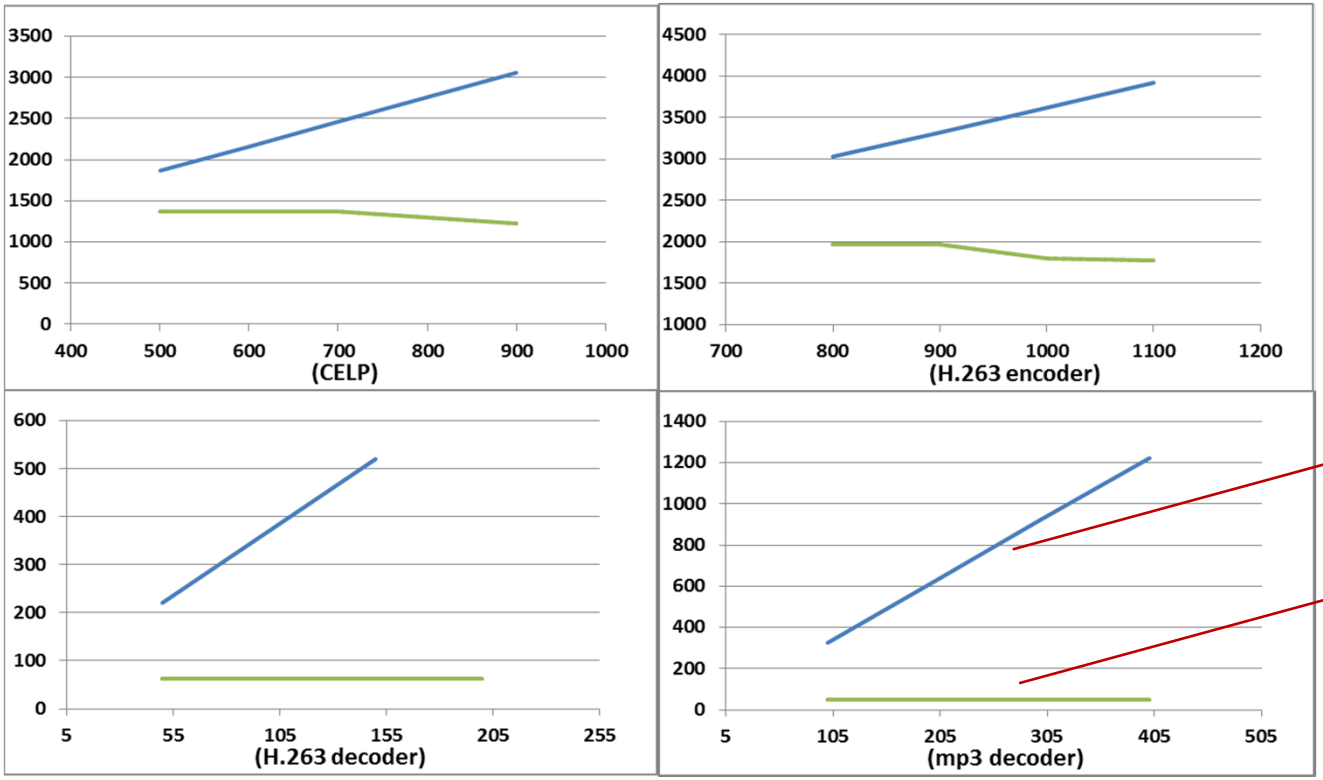
- Average improvement of NVRAM lifetime and the DRAM size with average computation times
- Similarly to the real-life applications, the optimizations with a given schedule and constructing an optimal schedule produce the same optimal results.

DRAM Size Minimization



- Minimization for a given NVRAM size with satisfying a given NVRAM lifetime constraint
- X axis - a given NVRAM size
- Y axis - the minimum DRAM size
- Minimization of DRAM size by 48.8% compared with a non-optimized algorithm

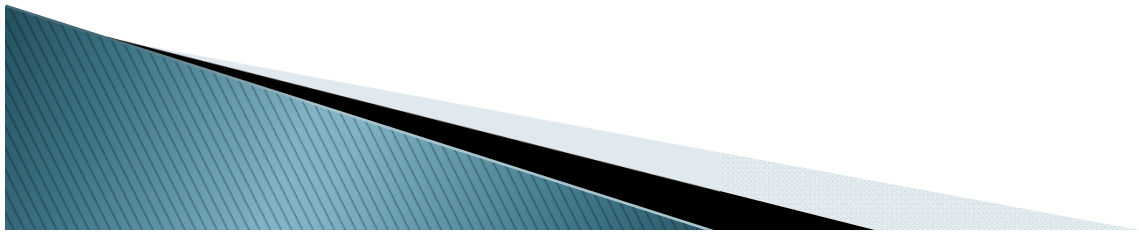
Energy Minimization



- Minimization of the energy consumption by varying the NVRAM size
- X axis : a given NVRAM size and
- Y axis : the minimal DRAM size
- Energy consumption saving by 165%

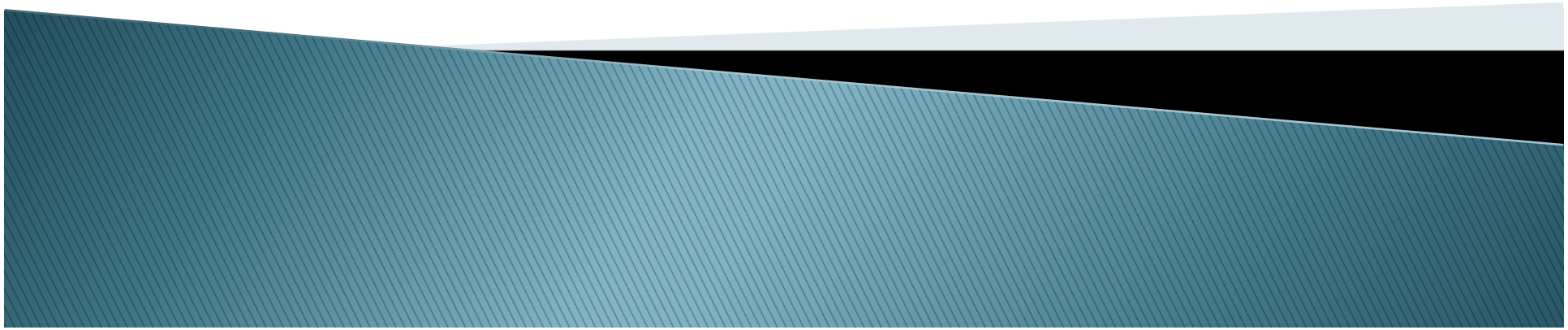
Summary

- Optimal buffer assignment algorithm for hybrid memory
- NVRAM lifetime maximization (63%)
- DRAM size minimization (48.8%)
- Energy consumption saving (165%)



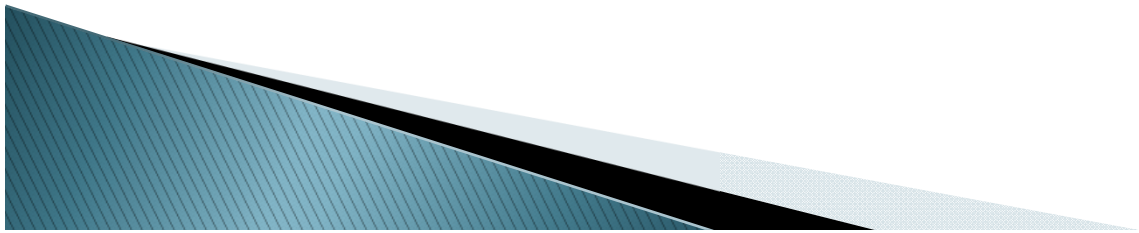
Design Tools :

Dataflow in Answer Set Programming (DASP)



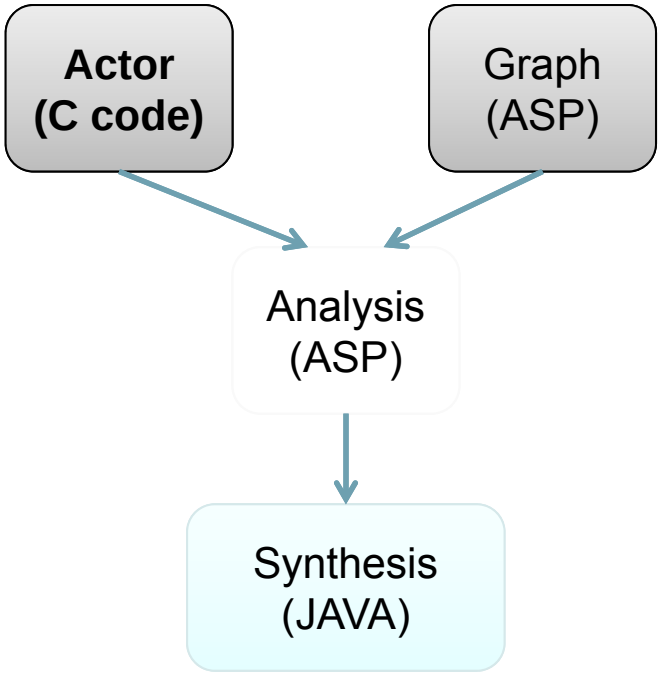
Goals of DASP

- Easy to use
  - No learning (C code based)
- Easy to install
  - Small size
- Easy to debug
  - C based
  - No code translation
- Easy to manage
  - Textual representation



DASP : Dataflow in Answer Set Programming

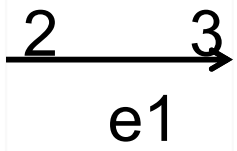
- Specification
  - Actor code
    - C code with macro keywords
  - Graph representation
    - Text (ASP)
- Mapping/Scheduling
  - Answer Set Programming
- Code synthesis
  - Java



DASP : Dataflow in Answer Set Programming

ActorA.c

```
#include <stdio.h>

#define WIDTH 176
#define HEIGHT 144

typedef struct {
      char pixel[WIDTH*HEIGHT];
} Image;

INFO_BEGIN
      OUT_PORT(output,Image,2)
INFO_END

VARS_BEGIN
      int current;
VARS_END

INIT_BEGIN
      VAR(current) = 0;
INIT_END

GO_BEGIN
      PORTn(output,0).pixel[0] = VAR(current);
      VAR(current)++;
      PORTn(output,1).pixel[0] = VAR(current);
      VAR(current)++;

      printf("send %d %d\n",PORTn(output,0).pixel[0],
PORTn(output,1).pixel[0]);
GO_END

WRAPUP_BEGIN
WRAPUP_END
```



ActorB.c

```
#include <stdio.h>

#define WIDTH 176
#define HEIGHT 144

typedef struct {
      char pixel[WIDTH*HEIGHT];
} Image;

INFO_BEGIN
      IN_PORT(input,Image,3)
INFO_END

VARS_BEGIN
VARS_END

INIT_BEGIN
INIT_END

GO_BEGIN
      printf("receive %d %d %d\n", PORTn(input,0).pixel[0],
PORTn(input,1).pixel[0], PORTn(input,2).pixel[0]);
GO_END

WRAPUP_BEGIN
WRAPUP_END
```

Graph

```
schematic(prodcons).
actor(actorA). actor(actorB).
instance(a,actorA). instance(b,actorB).
connect(e1,a,output,b,input).
```

Analysis and Scheduling

- Answer Set Programming
- CreateGraph
- Repetition
- Schedule

Answer: 1

port_info(a,output,"Image",25344,output,2)
port_info(b,input,"Image",25344,input,3) schematic(prodcons) actor(actorA)
actor(actorB) instance(a,actorA) instance(b,actorB)
connect(e1,a,output,b,input) time(1) node(b) node(a) edge(e1,a,b,2,3,0)
numEdges(1) sdfedgegcd(e1,1) possibleRepetition(a,3,0)
possibleRepetition(b,2,0) possibleRepetition(a,3,1) possibleRepetition(b,2,1)
repetition(a,3) repetition(b,2) totalInvocations(5) time(2) time(3) time(4)
time(5) sample(e1,0,0) fire(b,5) fire(b,4) fire(a,3) fire(a,2) fire(a,1)
sample(e1,2,1) sample(e1,4,2) sample(e1,6,3) sample(e1,3,4)
sample(e1,0,5) notfireable(b,2) notfireable(b,1) buffer_size(e1,6)
index_management(b,input,linear) index_management(a,output,linear)
top_schedule(a,1) top_schedule(a,2) top_schedule(a,3) top_schedule(b,4)
top_schedule(b,5)

Code Generation

a.h

```
#define a_output_port(n) e1[n+e1_output_index]
#define DECLARE_BUFFERS Image e1[6];

#define DECLARE_INDEXES int e1_output_index;
#define INIT_INDEXES e1_output_index=0;
#define UPDATE_INDEXES e1_output_index=
(e1_output_index+2)%6;
```

b.h

```
#define b_input_port(n) e1[n+e1_input_index]
#define DECLARE_BUFFERS extern Image e1[6];

#define DECLARE_INDEXES int e1_input_index;
#define INIT_INDEXES e1_input_index=0;
#define UPDATE_INDEXES
e1_input_index=(e1_input_index+3)%6;
```

ActorA.c

```
#include <stdio.h>

#define WIDTH 176
#define HEIGHT 144

typedef struct {
    char pixel[WIDTH*HEIGHT];
} Image;

INFO_BEGIN
    OUT_PORT(output,Image,2)
INFO_END

VARS_BEGIN
    int current;
VARS_END

INIT_BEGIN
    VAR(current) = 0;
INIT_END

GO_BEGIN
    PORTn(output,0).pixel[0] = VAR(current);
    VAR(current)++;
    PORTn(output,1).pixel[0] = VAR(current);
    VAR(current)++;

    printf("send %d %d\n",PORTn(output,0).pixel[0], PORTn(output,1).pi
GO_END

WRAPUP_BEGIN
WRAPUP_END
```

```
void a_init(); void a_go(); void a_wrapup();
void b_init(); void b_go(); void b_wrapup();

int main()
{
    int prodcons_count0;
    a_init(); b_init();

for(prodcons_count0=0;prodcons_count0<3;p
rodcons_count0++) {
        a_go();
        a_go();
        a_go();
        b_go();
        b_go();
    }
    a_wrapup(); b_wrapup();
}
```

main.c

ActorB.c

```
PORTn(input,0).pixel[0], PORTn(input,1).pixel[0],
```

Result

```
[hoh-ui-MacBook-Air:trunk/examples/prodcons] hoh% ./run_script
rm -f port_token_type.lp
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --type a actorA.c >> port_token_type.lp
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --type b actorB.c >> port_token_type.lp
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --buffer a port_token_type.lp graph.lp > a.h
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --buffer b port_token_type.lp graph.lp > b.h
java -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --info graph.lp > genInfo.c
gcc -g -c genInfo.c
gcc -g -o a_info.o -c actorA.c -DTASK_NAME=a -DTASK_INFO_GENERATION -I../../include -include task.h -include a.h
gcc -g -o b_info.o -c actorB.c -DTASK_NAME=b -DTASK_INFO_GENERATION -I../../include -include task.h -include b.h
gcc -o prodcons_info genInfo.o a_info.o b_info.o
% warning: iport/5 is never defined
% warning: initialDelay/2 is never defined
% warning: -index_management/3 is never defined
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --buffer_schedule a port_token_type.lp
schedule.result > a.h
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --buffer_schedule b port_token_type.lp
schedule.result > b.h
gcc -g -c prodcons.c
gcc -g -o a.o -c actorA.c -DTASK_NAME=a -I../../include -include task.h -include a.h
gcc -g -o b.o -c actorB.c -DTASK_NAME=b -I../../include -include task.h -include b.h
gcc -o prodcons prodcons.o a.o b.o
send 0 1
send 2 3
send 4 5
receive 0 1 2
receive 3 4 5
send 6 7
send 8 9
send 10 11
receive 6 7 8
receive 9 10 11
send 12 13
send 14 15
send 16 17
receive 12 13 14
receive 15 16 17
```

Support of Intermediate Port

```
INFO_BEGIN
      OUT_PORT(output,int,1)
      IN_PORT(input,int,1)
      IPORT(output,default,0)
      IPORT(input,default,1)
INFO_END

VARS_BEGIN
      int current;
VARS_END

INIT_BEGIN
      VAR(current) = 0;
INIT_END

GO_BEGIN
      int i;
      PORT(output) = VAR(current);
      for(i=0; i<3; i++) {
            SEND(default,0)
            RECEIVE(default,1)
            PORT(output) = PORT(input);
      }
      VAR(current)+=10;
GO_END

WRAPUP_BEGIN
WRAPUP_END
```

```
#include <stdio.h>

INFO_BEGIN
      IN_PORT(input,int,1)
      OUT_PORT(output,int,1)
INFO_END

INIT_BEGIN
INIT_END

GO_BEGIN
      PORT(output) = PORT(input)*2;
      printf("actor B : %d\n",PORT(input));
GO_END

WRAPUP_BEGIN
WRAPUP_END
```

Kmeans specified in DASP

graph.lp

```
schematic(kmeans).

instance(readFile,readFile).
instance(fork_point,fork_point).
instance(closestCentroid,closestCentroid).
instance(updateCentroid,updateCentroid).
instance(fork_point2,fork_point).
instance(print,print).

connect(e1,readFile,output,fork_point,input).
connect(e2,fork_point,output1,closestCentroid,point).
connect(e3,fork_point,output2,updateCentroid,point).
connect(e4,closestCentroid,index,updateCentroid,index).
connect(e5,updateCentroid,centroids,fork_point2,input).
connect(e6,fork_point2,output1,closestCentroid,centroid).
initialDelay(e6,10).
connect(e7,fork_point2,output2,print,input).
```

readFile.c

```
#include <stdio.h>
#include "kmeans.h"

INFO_BEGIN
        OUT_PORT(output,Point,1)
INFO_END

VARS_BEGIN
        FILE *ifp;
VARS_END

INIT_BEGIN
    VAR(ifp)=fopen("data.txt","r");
INIT_END

GO_BEGIN
    // read a point and generate it
    //Point output;
    //fread(&output, sizeof(int),dimension,ifp);
        PORT(output).p[0] = fgetc(VAR(ifp))-'0';
        PORT(output).p[1] = fgetc(VAR(ifp))-'0';
    printf("output point %f %f\n", PORT(output).p[0],PORT(output).p[1]);
GO_END

WRAPUP_BEGIN
WRAPUP_END
```

closestCentroid.c

updateCentroid.c

print.c

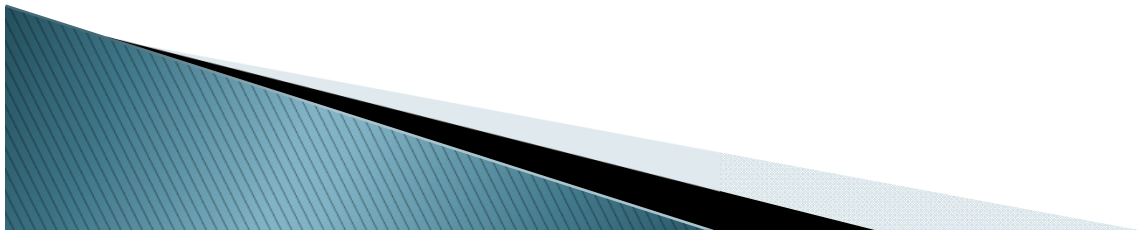
```
#include <stdio.h>
#include "kmeans.h"

INFO_BEGIN
        IN_PORT(input,Point,1)
INFO_END

VARS_BEGIN
VARS_END

INIT_BEGIN
INIT_END

GO_BEGIN
        printf("Centroids\n");
        int j;
        for(j=0; j<dimension; j++) {
                printf("%f ", PORT(input).p[j]);
        }
        printf("\n");
GO_END

WRAPUP_BEGIN
WRAPUP_END
```

Running Result

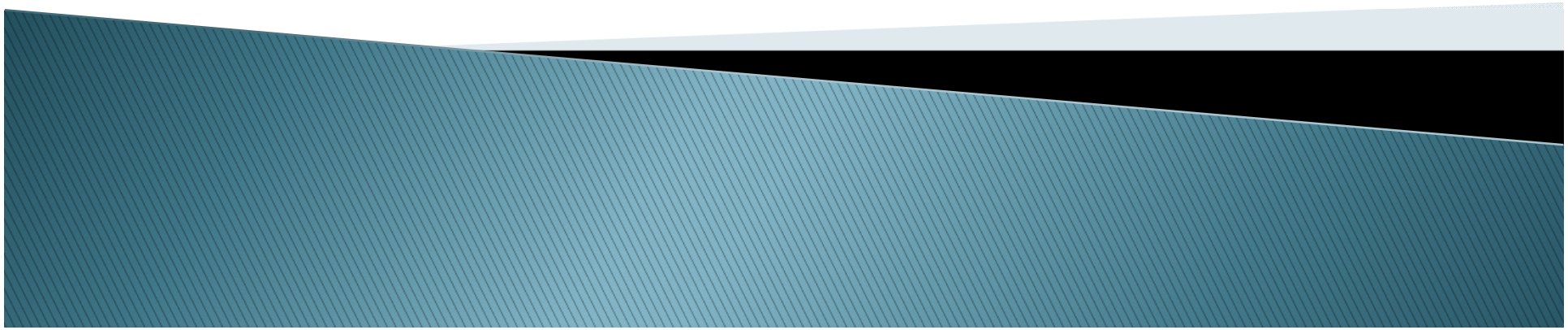
```
gcc -g -o closestCentroid_info.o -c closestCentroid.c -DTASK_NAME=closestCentroid -DTASK_INFO_GENERATION -I../../include -include task.h -include closestCentroid.h
gcc -g -o updateCentroid_info.o -c updateCentroid.c -DTASK_NAME=updateCentroid -DTASK_INFO_GENERATION -I../../include -include task.h -include updateCentroid.h
gcc -o kmeans_info genInfo.o readFile_info.o fork_point_info.o closestCentroid_info.o updateCentroid_info.o fork_point2_info.o print_info.o
% warning: iport/5 is never defined
% warning: -index_management/3 is never defined
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --buffer_schedule readFile port_token_type.lp schedule.result > readFile.h
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --buffer_schedule fork_point port_token_type.lp schedule.result > fork_point.h
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --buffer_schedule closestCentroid port_token_type.lp schedule.result > closestCentroid.h
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --buffer_schedule updateCentroid port_token_type.lp schedule.result > updateCentroid.h
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --buffer_schedule fork_point2 port_token_type.lp schedule.result > fork_point2.h
java  -classpath /Users/hoh/Documents/Research/devel/dasp/trunk/synthesis TaskInformation --buffer_schedule print port_token_type.lp schedule.result > print.h
gcc -g -c kmeans.c
gcc -g -o closestCentroid.o -c closestCentroid.c -DTASK_NAME=closestCentroid -I../../include -include task.h -include closestCentroid.h
gcc -g -o updateCentroid.o -c updateCentroid.c -DTASK_NAME=updateCentroid -I../../include -include task.h -include updateCentroid.h
gcc -o kmeans kmeans.o readFile.o fork_point.o closestCentroid.o updateCentroid.o fork_point2.o print.o
centroid[0].p[0]=7.000000]
centroid[0].p[1]=9.000000]
centroid[1].p[0]=3.000000]
centroid[1].p[1]=8.000000]
centroid[2].p[0]=0.000000]
centroid[2].p[1]=2.000000]
centroid[3].p[0]=4.000000]
centroid[3].p[1]=8.000000]
output point 0.000000 0.000000
output point 1.000000 1.000000
output point 2.000000 2.000000
output point 3.000000 3.000000
output point 4.000000 4.000000
output point 5.000000 5.000000
output point 6.000000 6.000000
output point 7.000000 7.000000
output point 8.000000 8.000000
output point 9.000000 9.000000
read point 0.000000 0.000000
minIndex : 2, minDistance 2.000000
read point 1.000000 1.000000
minIndex : 2, minDistance 1.414214
read point 2.000000 2.000000
minIndex : 2, minDistance 2.000000
read point 3.000000 3.000000
minIndex : 2, minDistance 3.162278
read point 4.000000 4.000000
minIndex : 3, minDistance 4.000000
read point 5.000000 5.000000
minIndex : 3, minDistance 3.162278
read point 6.000000 6.000000
minIndex : 3, minDistance 2.828427
read point 7.000000 7.000000
minIndex : 0, minDistance 2.000000
read point 8.000000 8.000000
minIndex : 0, minDistance 1.414214
read point 9.000000 9.000000
minIndex : 0, minDistance 2.000000
Centroids
8.000000 8.000000
Centroids
0.000000 0.000000
Centroids
1.500000 1.500000
Centroids
5.000000 5.000000
```

Project : DASP

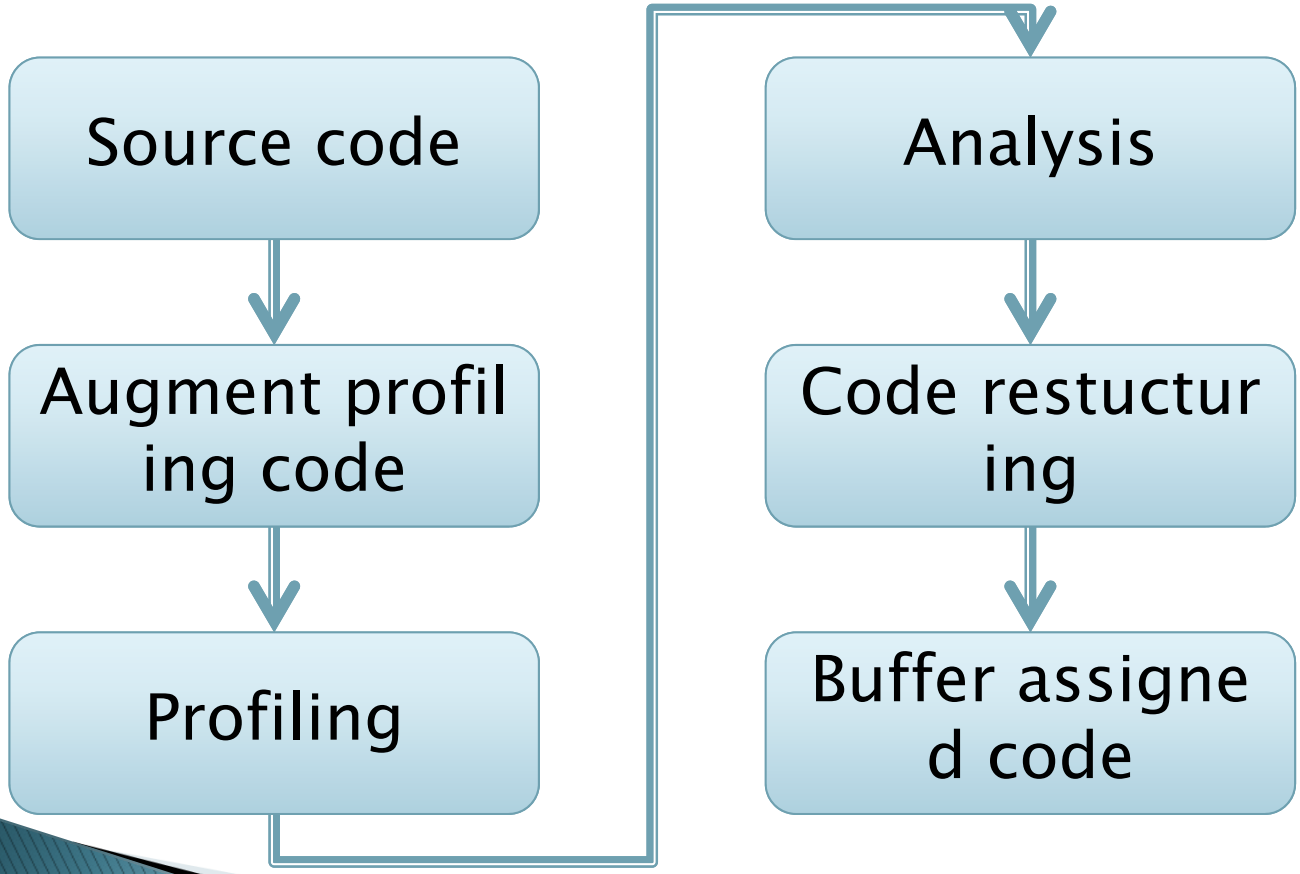
- http://dev.naver.com/projects/dasp
- On-going works
  - Support multi-core
  - Support distributed computing (MapReduce …)
  - Implement multimedia applications
    - H.263 encoder/decoder, H.264/AVC encoder/decoder, HeVC encoder/decoder
  - Implement data mining applications
    - Clustering algorithm
  - Integrate all the developed algorithms
    - Buffer sharing, buffer index management, scheduling, hybrid memory



Design Tools : Compiler Approach



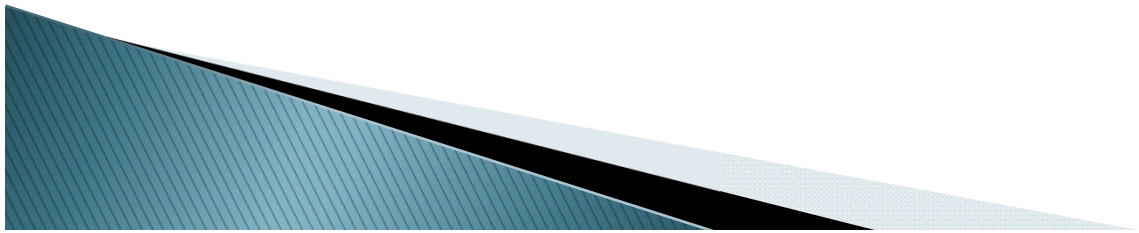
Buffer Assignment onto Hybrid Memory for FFMpeg



Read intensive : NVRAM
Write intensive : DRAM

Profiling

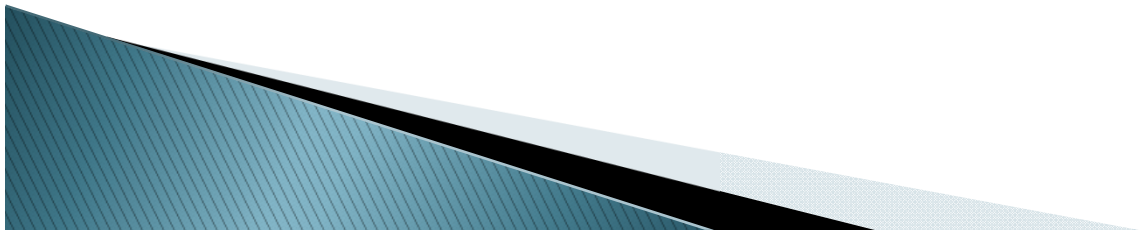
- MpegEncContext memory size : 10,468 Bytes
  - Total read Bytes : 93,418,145
  - Total write Bytes: 32,423,460



MpegEncContext struct Members	Type	Read Count	Write Count	
->v_edge_pos	int	447201		
->current_picture.f.linesize[o]	int	894402		
->current_picture.f.linesize[o]	int	547200		
->mb_x	int	447201		
->mb_x	int	447201		
->mb_x	int	547200		
->mb_y		447201		
->mb_y	int	447201		
->mb_y	int	547200		
->mb_stride	int	447201		
->mb_stride	int	547200		
->mb_skipped	int		376931	
->mb_skipped	int		376931	
->interlaced_dct	int	547200		
->dest[0]	uint8_t	547200		
->mv_dir	int		376931	
->mv_type	int		547200	
->mv_type	int		447201	
->block	short	1093792		
->linesize	int			
->y_dc_scale	int	399996		
->c_dc_scale	int	199998		
->current_picture.f.qscale_table	int8_t	599994		
->dsp.idct_permutation[]	uint8_t	8399916		
->h263_pred	int	547200		
->block_last_index[]	int		2261586	
->current_picture.f.mb_type[]	uint32_t		547200	
->mcsel	int		376931	
->mv[][][]	int		517471	
->mb_intra	int		119869	
->ac_pred	int	99999	99999	
->use_intra_dc_vlc	int		99999	

Spliting the MpegEncContext struct

- MpegEncContext is most used struct in the FFmpeg application.
- Manually placing variable access counters.
- Move read intensive variable to second new struct
- Second struct is pre-allocated in memory in the global segment.
- Pre-allocated struct pointer is assigned to the MpegEncContext at the point of initialization.



MpegEncContext with pointer to second struct

```
typedef struct MpegEncContext {
.
.

MpegEncContext_Persistent_  * Persistent_  ;

} MpegEncContext;
```

Second struct with read intensive members

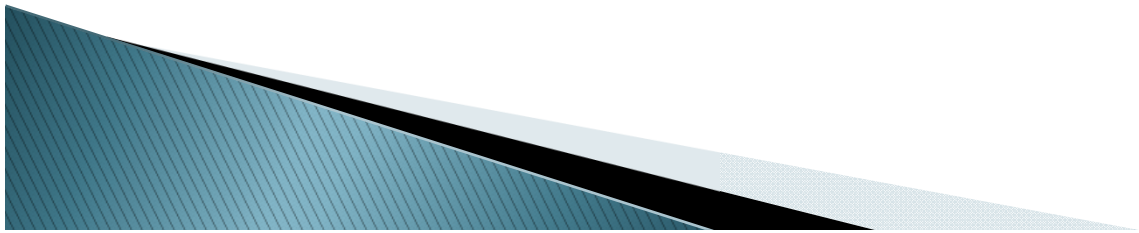
```
typedef struct MpegEncContext_Persistent_{
  int h_edge_pos, v_edge_pos
  int mb_x, mb_y;
  int mb_stride;
  DCTELEM (*block)[64];
  int y_dc_scale, c_dc_scale;
  int h263_pred;
  int ac_pred;
}MpegEncContext_Persistent_;
```

Second struct pre-allocated global memory, and function that assigns it to MpegEncContext at the point where MpegEncContext is initialized

```
MpegEncContext_Persistent_ pre_allocation_[4] ;
int MpegEncContext_Persistent_Allocated_Count = 0;

void assign_allocated_Persistent_struct(void * s_struct)
{
  MpegEncContext * ptr = (MpegEncContext *) s_struct;
  ptr->Persistent_ = & testGlobal[MpegEncContext_Persistent_Allocated_Count ++ ] ;
}
```

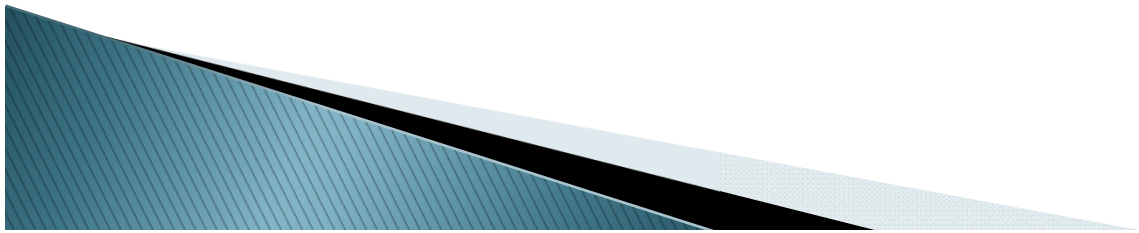
Extend Analysis to all heavily used structs in the FFmpeg

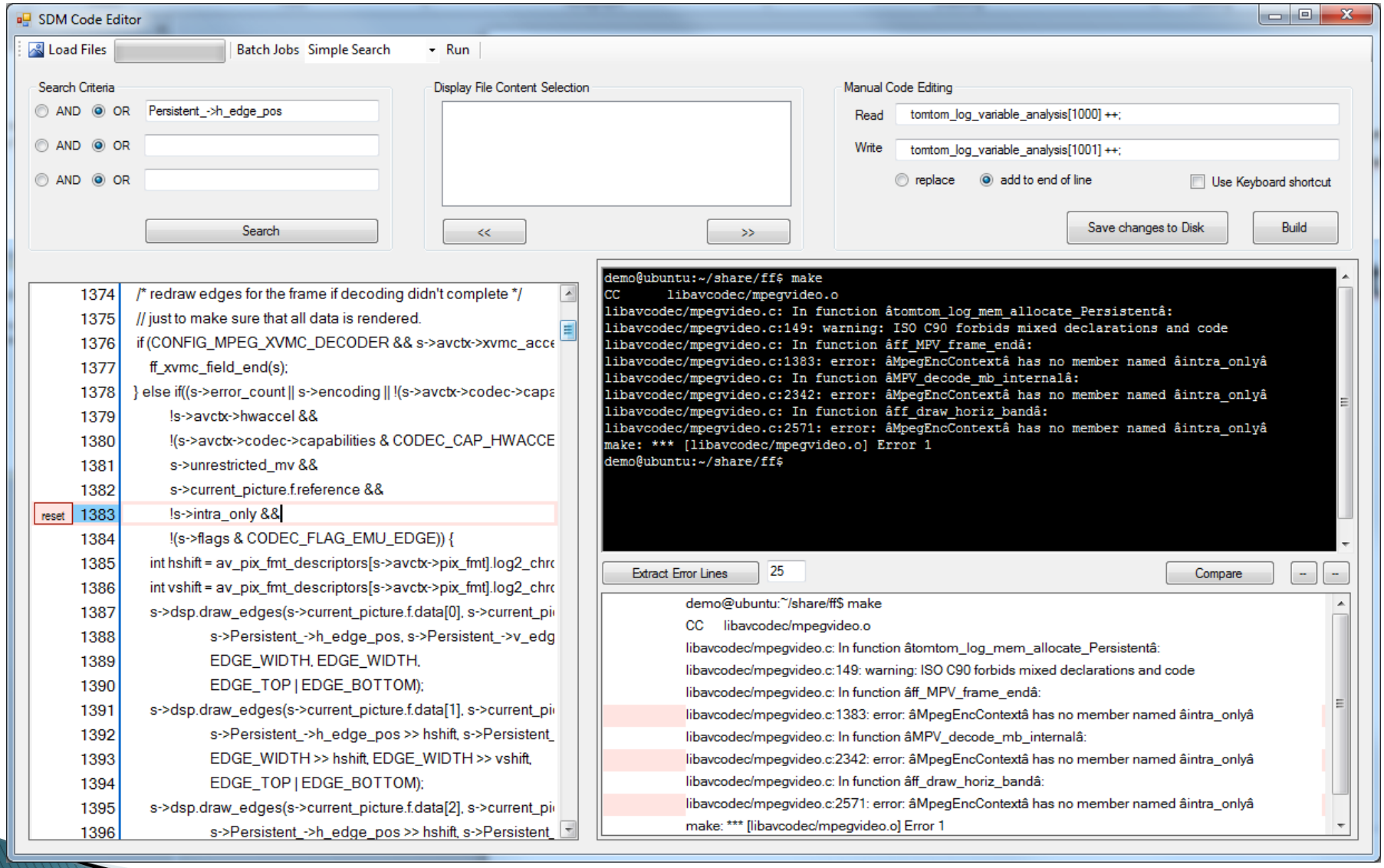
- Challenges
  - The FFmpeg had over 550,000 lines of C codes.
  - Each struct had a lot of members, which had to be t raced individually, eg. The MpegEncContext struct had about 200 member to be traced.
  - Some structs had members with same name, for thi s reason a simple find and replace may not work.



Solution – Created a custom code edition App

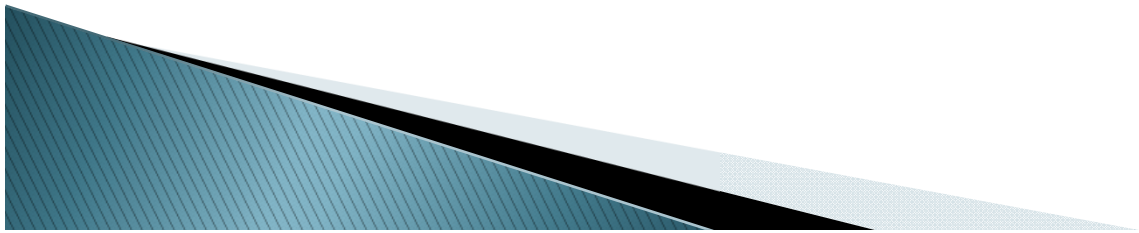
- Searching with complex criteria.
- Execute build command, and with one click o pens up the source file and scrolls to error lin e.
- Automatically place variable access counters at the appropriate position in the source code




SDM Code Editor
Load Files
Batch Jobs
Simple Search
Run
Search Criteria
AND
OR
Persistent_->h_edge_pos
Search
Display File Content Selection
<<
>>
Manual Code Editing
Read
tomtom_log_variable_analysis[1000] ++;
Write
tomtom_log_variable_analysis[1001] ++;
replace
add to end of line
Use Keyboard shortcut
Save changes to Disk
Build
1374 /* redraw edges for the frame if decoding didn't complete */
1375 // just to make sure that all data is rendered.
1376 if (CONFIG_MPEG_XVMC_DECODER && s->avctx->xvmc_acce
1377 ff_xvmc_field_end(s);
1378 } else if((s->error_count || s->encoding || !(s->avctx->codec->capa
1379 !s->avctx->hwaccel &&
1380 !(s->avctx->codec->capabilities & CODEC_CAP_HWACCE
1381 s->unrestricted_mv &&
1382 s->current_picture.f.reference &&
reset 1383 !s->intra_only &&
1384 !(s->flags & CODEC_FLAG_EMU_EDGE)) {
1385 int hshift = av_pix_fmt_descriptors[s->avctx->pix_fmt].log2_chro
1386 int vshift = av_pix_fmt_descriptors[s->avctx->pix_fmt].log2_chro
1387 s->dsp.draw_edges(s->current_picture.f.data[0], s->current_pi
1388 s->Persistent_->h_edge_pos, s->Persistent_->v_edg
1389 EDGE_WIDTH, EDGE_WIDTH,
1390 EDGE_TOP | EDGE_BOTTOM);
1391 s->dsp.draw_edges(s->current_picture.f.data[1], s->current_pi
1392 s->Persistent_->h_edge_pos >> hshift, s->Persistent_
1393 EDGE_WIDTH >> hshift, EDGE_WIDTH >> vshift,
1394 EDGE_TOP | EDGE_BOTTOM);
1395 s->dsp.draw_edges(s->current_picture.f.data[2], s->current_pi
1396 s->Persistent_->h_edge_pos >> hshift, s->Persistent_
demo@ubuntu:~/share/ff$ make
CC libavcodec/mpegvideo.o
libavcodec/mpegvideo.c: In function âtomtom_log_mem_allocate_Persistentâ:
libavcodec/mpegvideo.c:149: warning: ISO C90 forbids mixed declarations and code
libavcodec/mpegvideo.c: In function âff_MPV_frame_endâ:
libavcodec/mpegvideo.c:1383: error: âMpegEncContextâ has no member named âintra_onlyâ
libavcodec/mpegvideo.c: In function âMPV_decode_mb_internalâ:
libavcodec/mpegvideo.c:2342: error: âMpegEncContextâ has no member named âintra_onlyâ
libavcodec/mpegvideo.c: In function âff_draw_horiz_bandâ:
libavcodec/mpegvideo.c:2571: error: âMpegEncContextâ has no member named âintra_onlyâ
make: *** [libavcodec/mpegvideo.o] Error 1
demo@ubuntu:~/share/ff$
Extract Error Lines
25
Compare
demo@ubuntu:~/share/ff$ make
CC libavcodec/mpegvideo.o
libavcodec/mpegvideo.c: In function âtomtom_log_mem_allocate_Persistentâ:
libavcodec/mpegvideo.c:149: warning: ISO C90 forbids mixed declarations and code
libavcodec/mpegvideo.c: In function âff_MPV_frame_endâ:
libavcodec/mpegvideo.c:1383: error: âMpegEncContextâ has no member named âintra_onlyâ
libavcodec/mpegvideo.c: In function âMPV_decode_mb_internalâ:
libavcodec/mpegvideo.c:2342: error: âMpegEncContextâ has no member named âintra_onlyâ
libavcodec/mpegvideo.c: In function âff_draw_horiz_bandâ:
libavcodec/mpegvideo.c:2571: error: âMpegEncContextâ has no member named âintra_onlyâ
make: *** [libavcodec/mpegvideo.o] Error 1

Conclusion

- Application optimization on hybrid memory
- Supporting tools
  - DASP : dataflow tool
  - Compiler approach
    - Interactive approach



Thank you!