Modelling a Multi-Core Media Processor Using JCSP

Anna Kosek, Napier University, UK

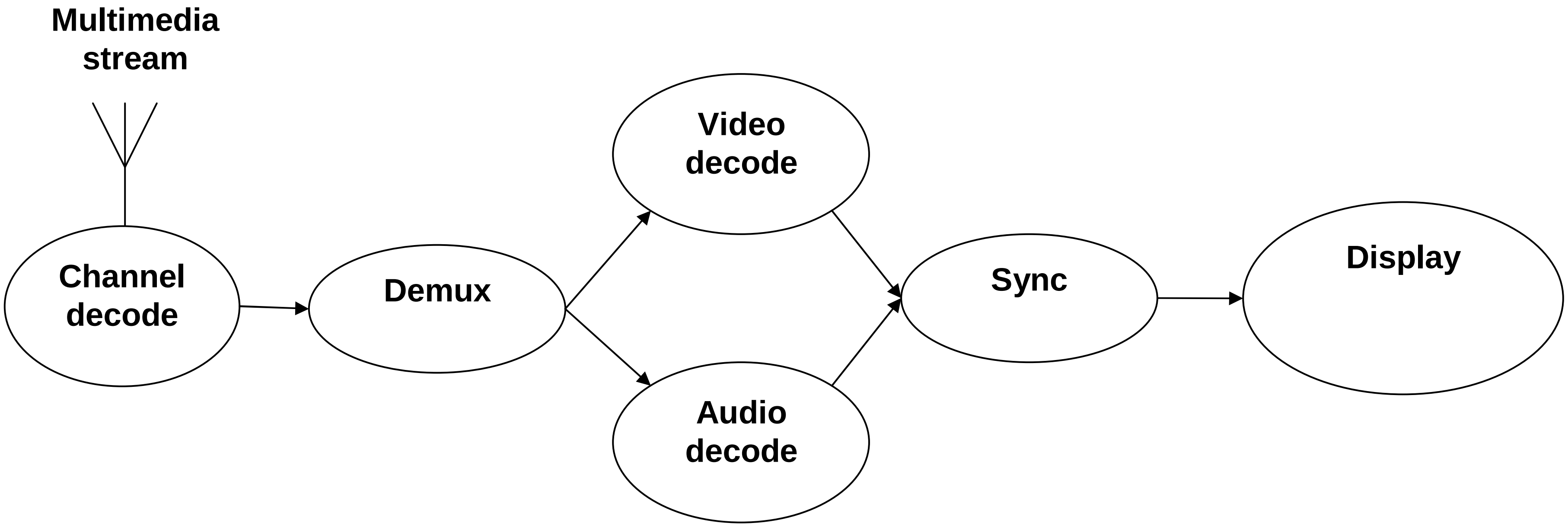
Jon Kerridge, Napier University, UK

Aly Syed, NXP, The Netherlands

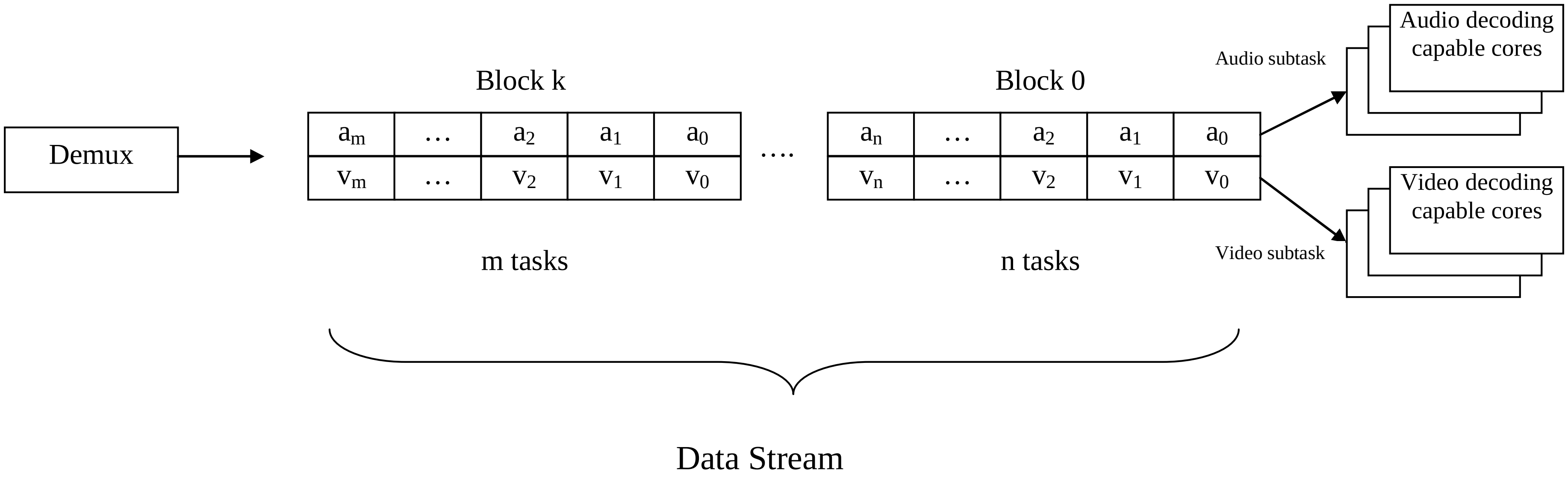
Motivation

- Use CSP model capabilities to create a parallel system
- Use JCSP to simulate a multi-core audio/video processor
- Use of mobile processes to implement allocation of tasks to cores
- Use allocation to control heat generation
  - For hand held devices

Original architecture



Data stream



Block k
Block 0
Demux
a_m … a_2 a_1 a_0
v_m … v_2 v_1 v_0
….
a_n … a_2 a_1 a_0
v_n … v_2 v_1 v_0
Audio subtask
Video subtask
Audio decoding capable cores
Video decoding capable cores
m tasks
n tasks
Data Stream

Data structure

Name	Description	Special values
b	Block number	Starts from 0
t	Task number	Starts from 0
c	Subtask category	Audio = 0 Video = 1
r	Requested subtask type	Starts from 0
w	Amount of work of requested subtask type (units of work)	Starts from 1
o	Variable reserved for actual core performance (outcome) time needed to execute requested subtask	By default equals 0, but changes after subtask processing

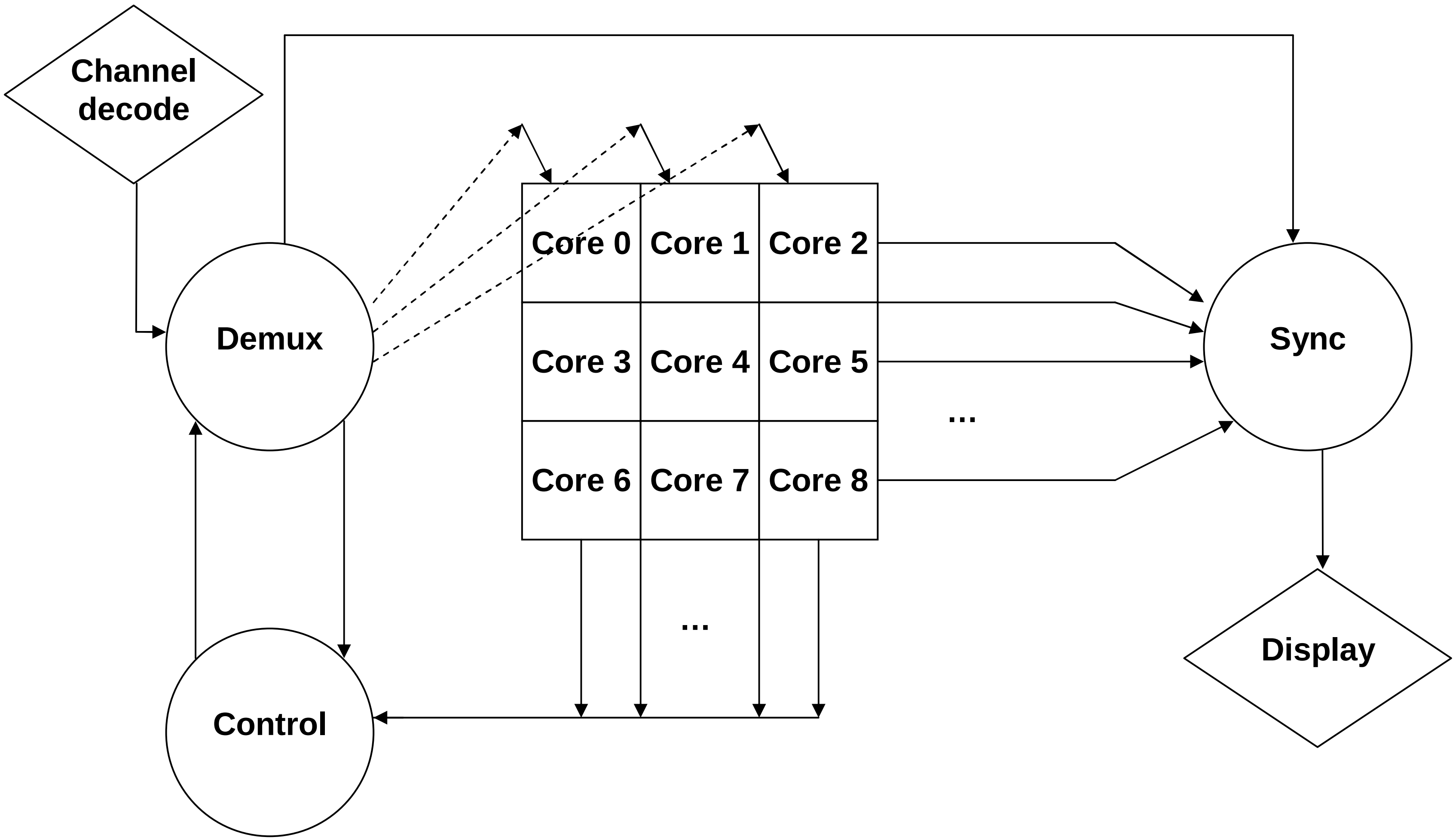
JCSP Components

- Processes
- Channels
- Alternative
- Network Channels (jcsp.net)
- Mobile processes

Mobile process

- Moves around the system
- Can be connected to any process and be run in parallel with it
- Mobile process carries data set representing a subtask

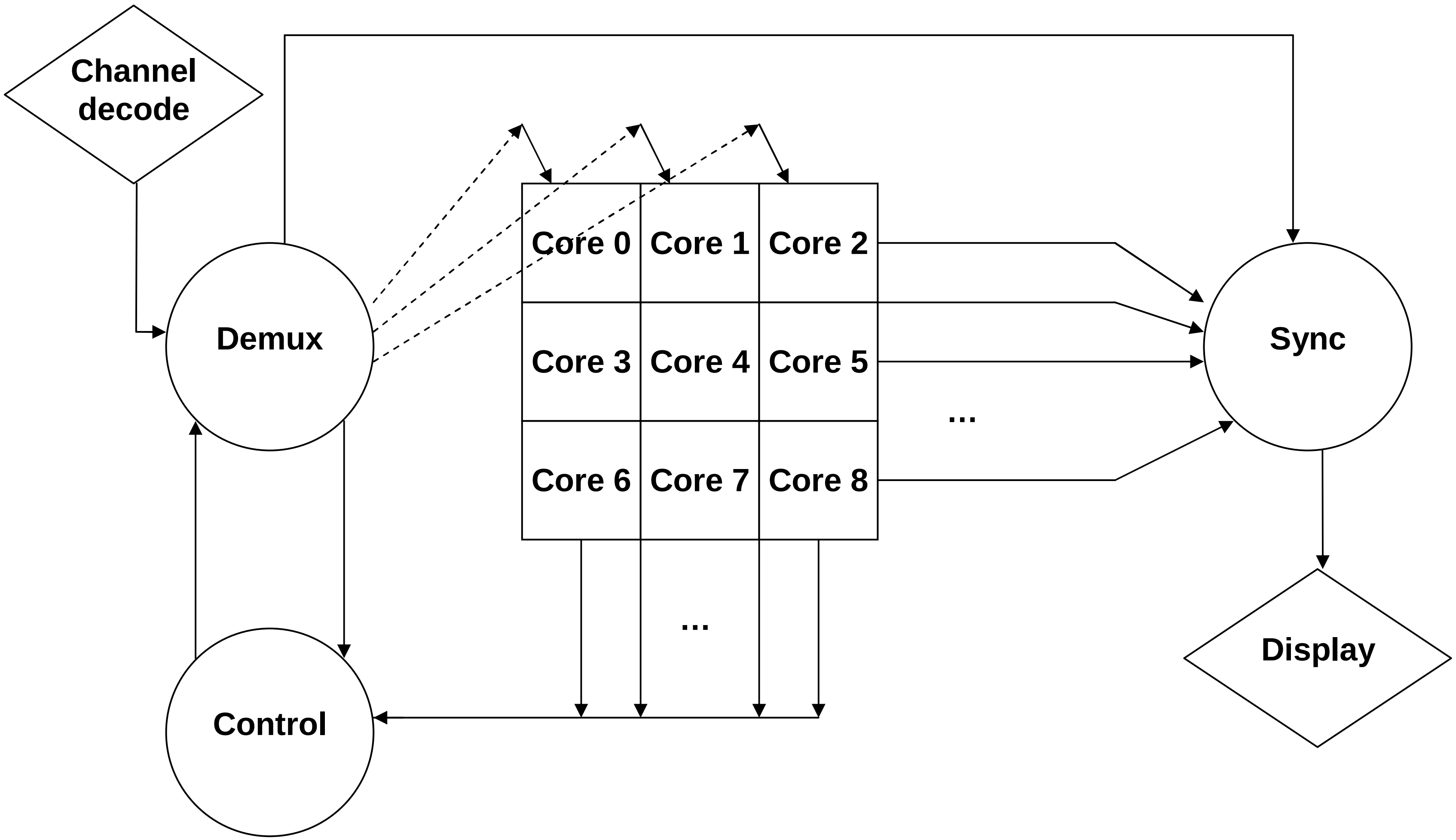
Architecture



Channel Decode

- Reads data stream
- Sends the task, divided into subtasks, to the Demux process

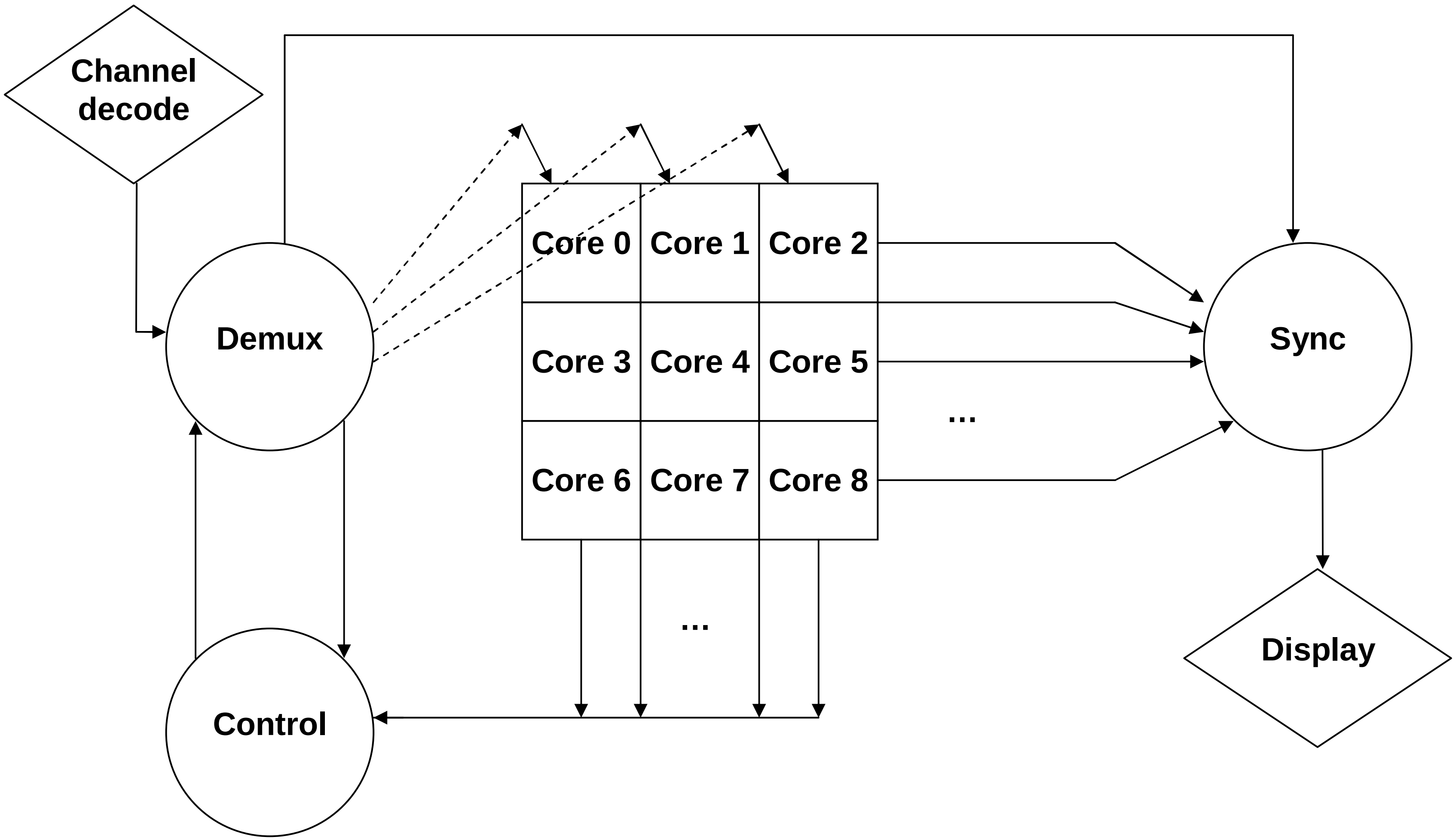
Architecture



Demux

- Requests core location from the Control process
- Creates connection and sends subtask to that Core
- Sends information about subtask ordering to the Sync process to enable synchronisation

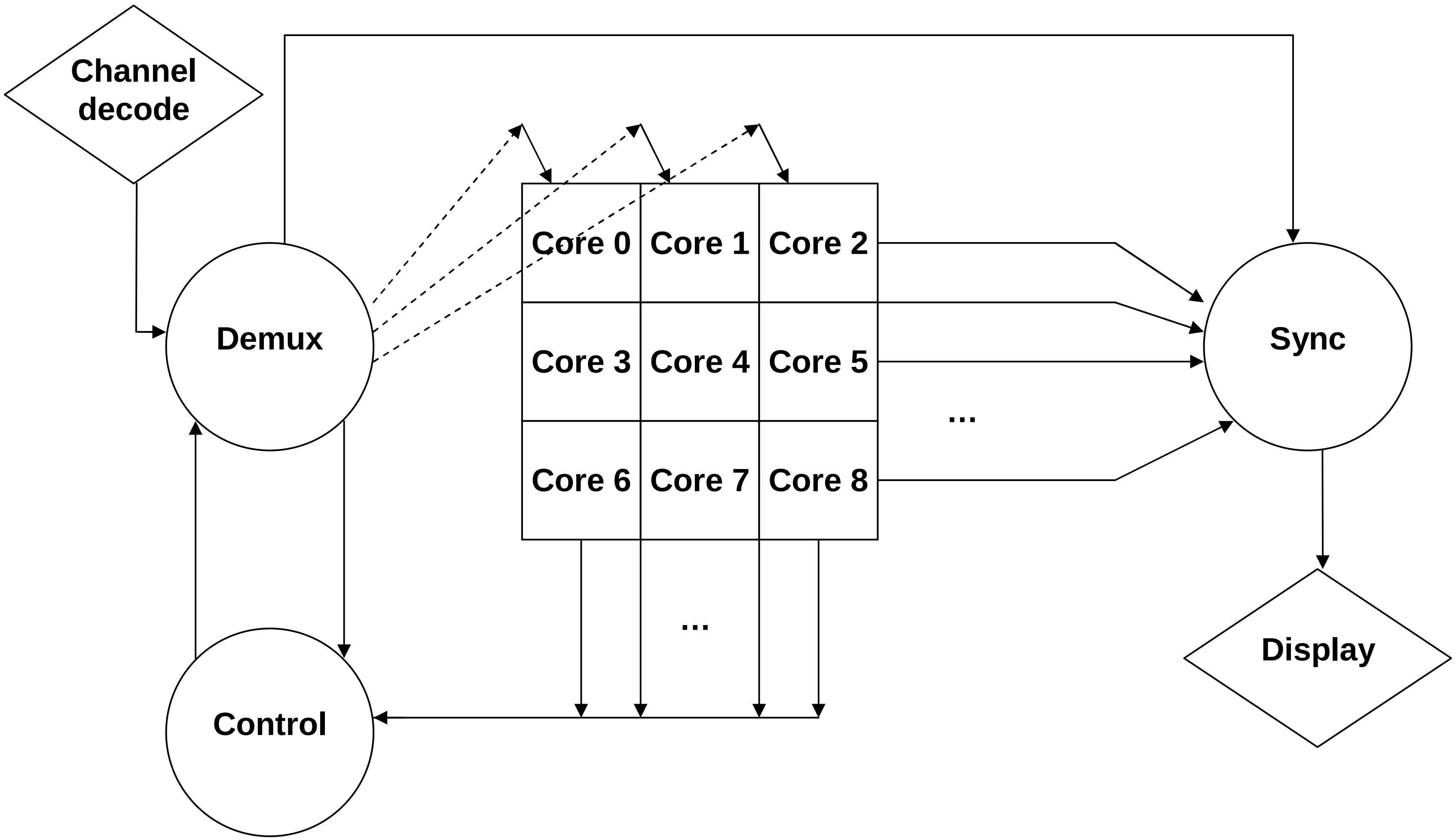
Architecture



Control

- Decides where to allocate a subtask
- Cores register their state and capabilities with the Control process

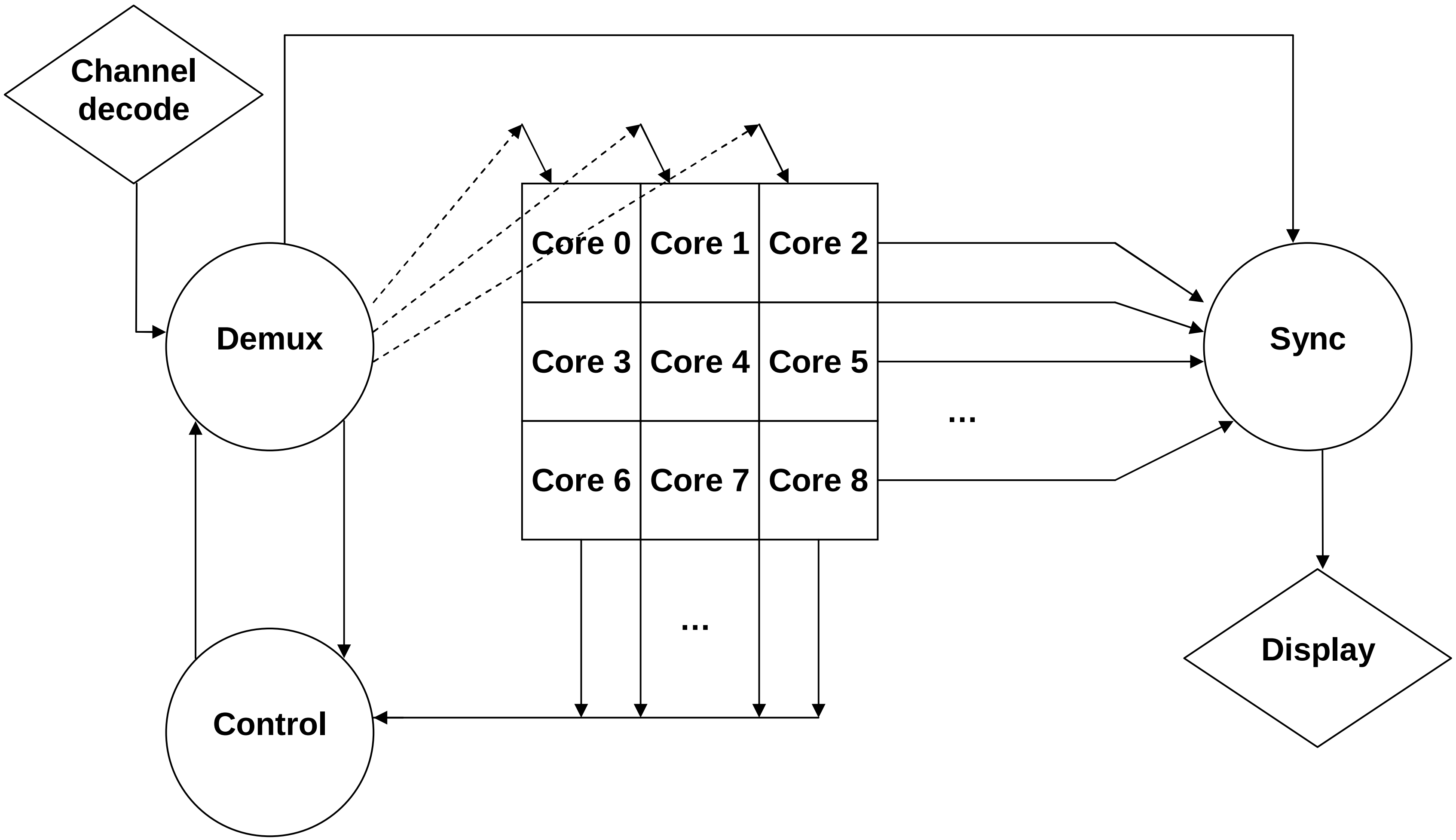
Architecture



Core

- Has its own capabilities
- Sends information to the Control process about: capabilities, avaliability and location
- Receives a subtask and executes it in a time depending on the core’s capabilities
- All cores run in parallel
- A core gains heat if the subtask’s execution takes more than 0,1 sec
- A hot core takes longer to process a subtask
- Heated cores cool down

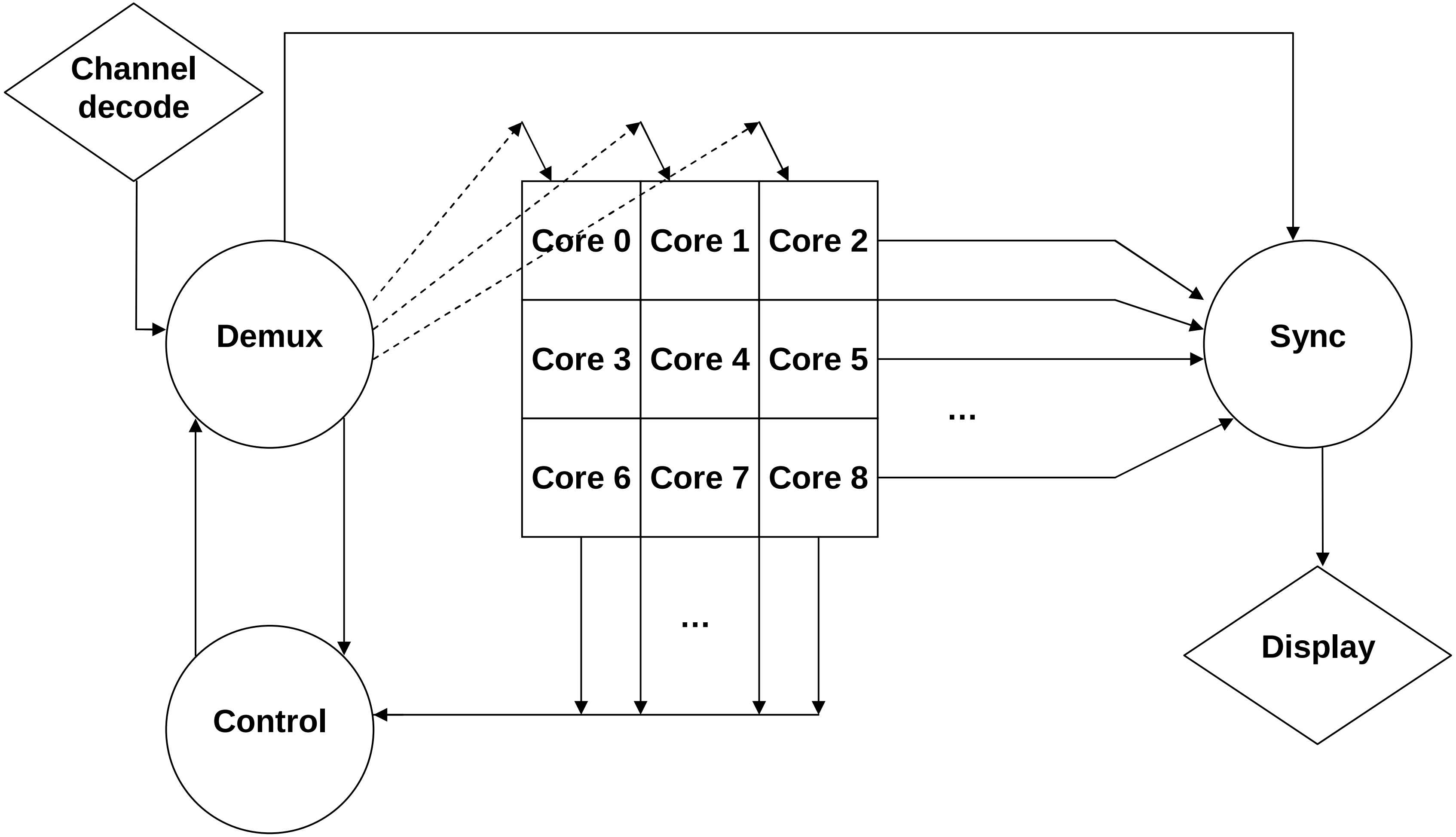
Architecture



Sync

- Synchronises audio and video subtasks
- Sends synchronised subtasks to the Dispay process
- Outputs some basic information concerning where subtasks were processed

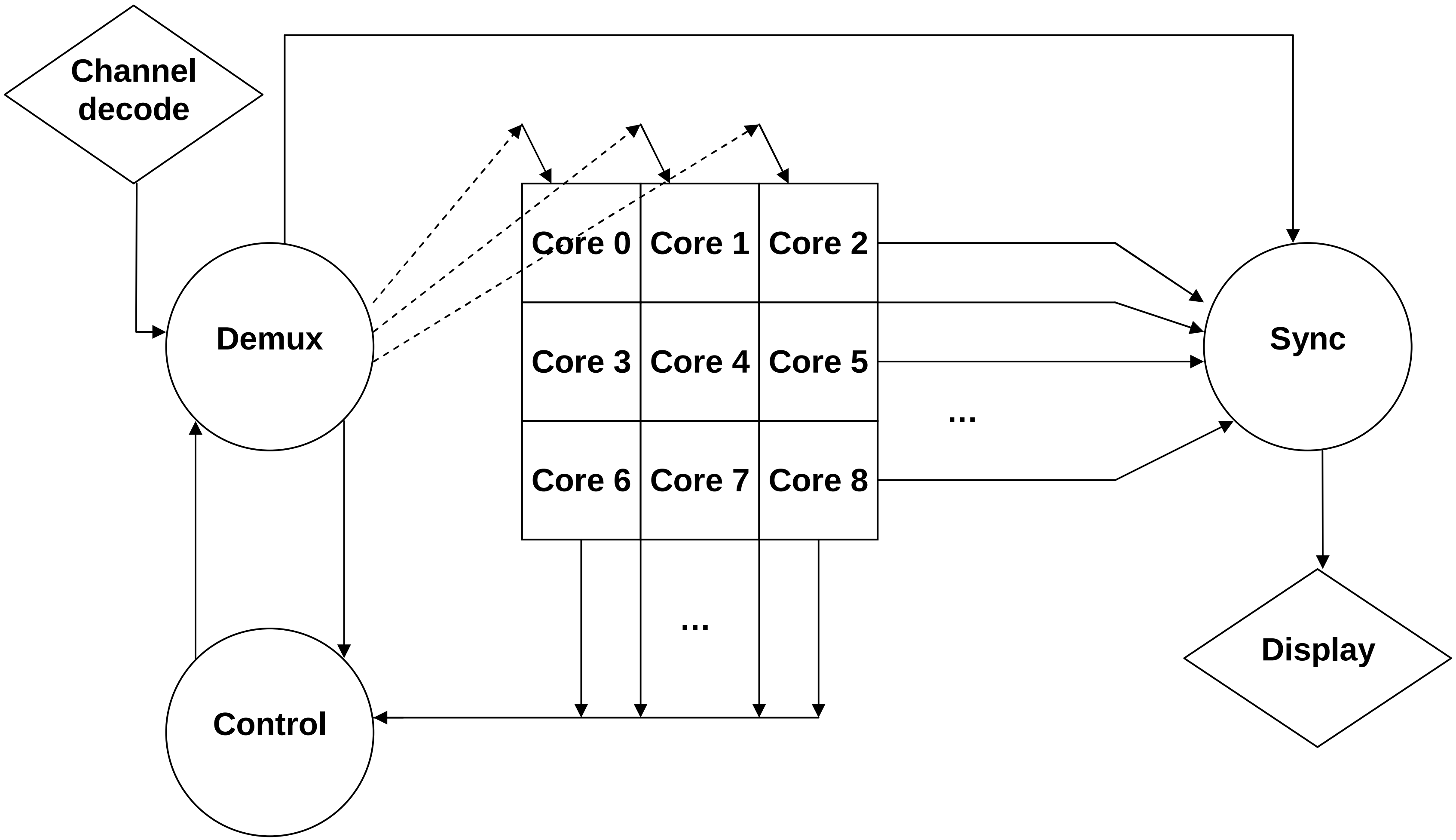
Architecture



Display

- Displays output information such as:
  - Block number
  - Audio / Video subtask
  - Subtask number
  - Optimum time
  - Actual time

Architecture



Typical output

Block:	A/V:	subtask:	optimum time:	actual time:
0	A	0	50	100
0	V	0	100	700
0	A	1	100	100
0	V	1	200	200
0	A	2	70	140
0	V	2	200	1800
1	A	0	50	500
1	V	0	100	1000
1	A	1	100	1000
1	V	1	200	200
1	A	2	70	70
1	V	2	200	1400
1	A	3	70	700
1	V	3	200	400

Corresponding Sync output

```
Sync: task: 0 (task type:  0, category: 0), block: 0, core: 7, processing time: 100
Sync: task: 0 (task type: 10, category: 1), block: 0, core: 7, processing time: 700
Sync: task: 1 (task type:  6, category: 0), block: 0, core: 7, processing time: 100
Sync: task: 1 (task type: 19, category: 1), block: 0, core: 1, processing time: 200
Sync: task: 2 (task type:  4, category: 0), block: 0, core: 1, processing time: 140
Sync: task: 2 (task type: 11, category: 1), block: 0, core: 7, processing time: 1800
Sync: task: 0 (task type:  9, category: 0), block: 1, core: 4, processing time: 500
Sync: task: 0 (task type: 14, category: 1), block: 1, core: 4, processing time: 1000
Sync: task: 1 (task type:  7, category: 0), block: 1, core: 4, processing time: 1000
Sync: task: 1 (task type: 19, category: 1), block: 1, core: 1, processing time: 200
Sync: task: 2 (task type:  6, category: 0), block: 1, core: 7, processing time: 70
Sync: task: 2 (task type: 10, category: 1), block: 1, core: 7, processing time: 1400
Sync: task: 3 (task type:  1, category: 0), block: 1, core: 4, processing time: 700
Sync: task: 3 (task type: 12, category: 1), block: 1, core: 1, processing time: 400
```

Conclusions

- A basic implementation capturing:
  - Distribution of work
  - Selection of cores
  - Core heating and cooling
  - Synchronisation
  - Limited output and performance capture

Future work

- Improve the input format
- Improve the monitoring capability
- Automating functional testing
- Core failure
- More complex heating and cooling strategies
- Multiple input streams
- Switching between cores to complete one subtask
- Removing need for a single Control process