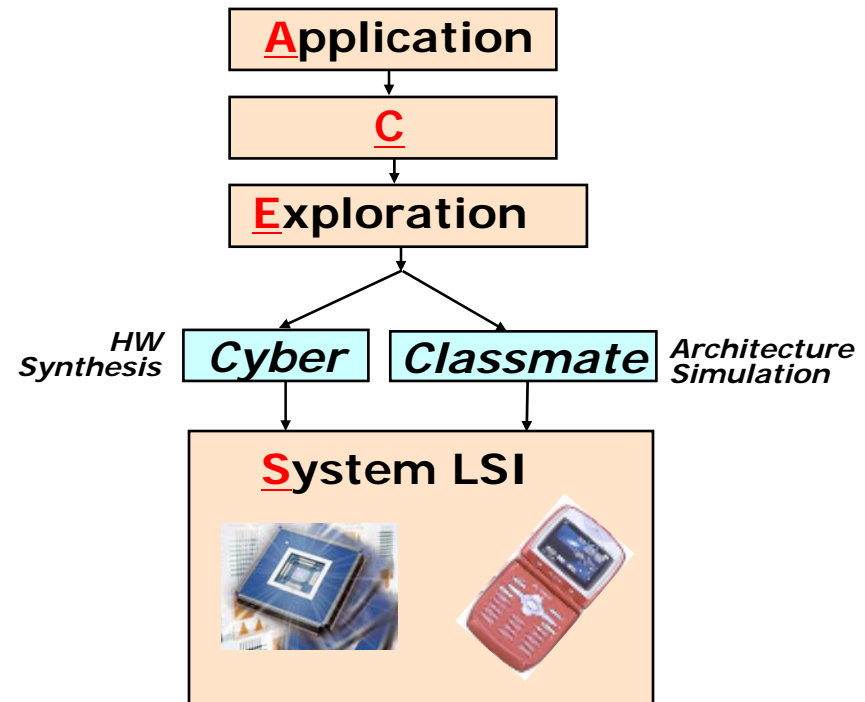


# Automatic Softening of System-On-Chip Hardware



**ACES**

Abhishek Mitra  
UC Riverside



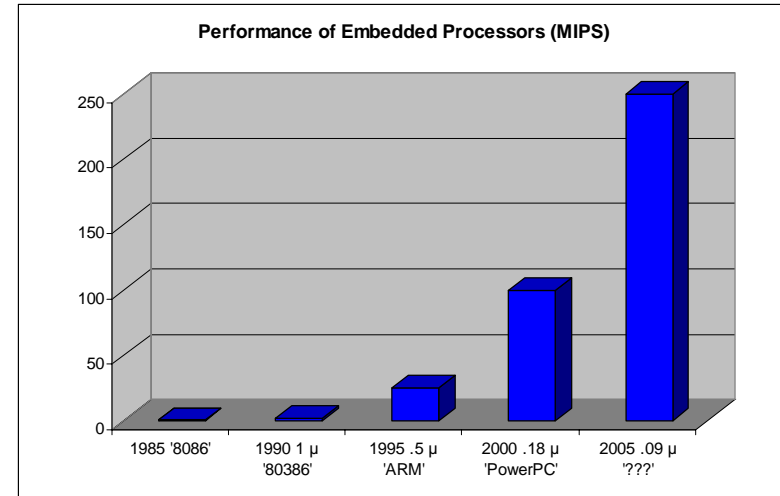
# Outline

---

- **Softening: Motivation**
- **Softening Methodology**
  - **Target Selection**
  - **Interface Synthesis**
  - **Code generation**
  - **Softened task scheduling**
- **Case Study: 802.11b Wireless LAN Subsystem**
- **Experimental Results**

# Software Content in System LSIs

Sustained improvements in semiconductor technology allows for increased migration of system functionality from hardware to software



- Growth in software content in modern Systems
  - Typical code size : 100K lines in 1995 to 1000K lines in 2002  
(Jerry Fiddler, Wind River Systems, Keynote address, Design Automation Conf. 2002.)
- Moore's Law (chip complexity/performance 2x ~ 18 months)
- But standards evolve at a far slower rate (every few yrs), e.g. Five years between WEP(802.11b) and WPA(802.11i)
- Exploiting the improvements in embedded processor performance is laborious
  - Largely manual conversion of **hardware oriented** designs to mixed HW/SW designs

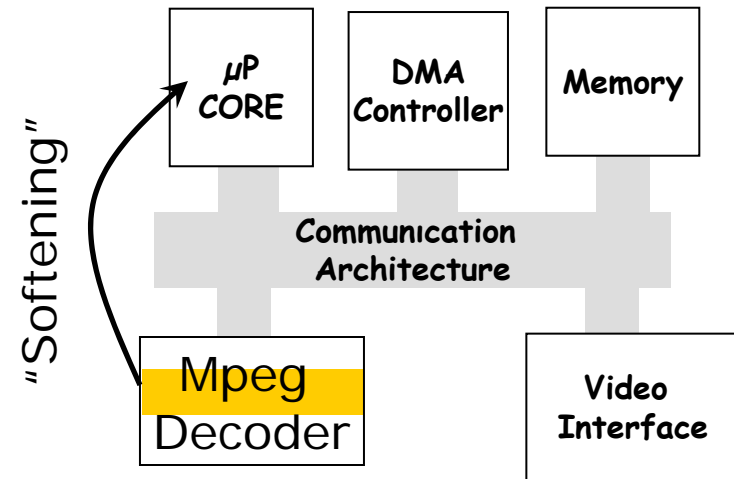
# Research Objectives

---

- Enable automatic migration of system functionality from HW to SW
  - **“Softening”** of system-on-chip hardware

- Advantages:

- Shorter time to market
- Reduced hardware cost
- Upgradeability/flexibility



# Hardware Softening Overview

---

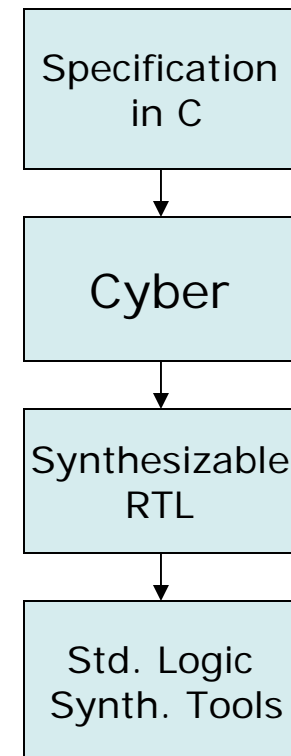
- **Implementation flow:**
  - **Given a hardware block, providing a semi-automated flow for migration to software**
    - Selection of softening target (manual)
    - Data dependence Analysis
    - HW/SW Interface Synthesis
    - Software Code Generation
    - Scheduler design
- **Analysis flow:**
  - **Given a system description, identifying candidate hardware blocks for potential migration to software**
    - Ongoing work

# Softening Target Specification

## □ Example: C-based specification for CRC-32

```
.....
doneCrcCalcOut = 0;
wait(startCrcln);
noOfBytes = bufferSizeIn;
/*SW-BEGIN*/
crc_temp = 0xFFFFFFFF; //Initialize the CRC
index = (crc_temp >> 24) & 0xFF;
crc_temp = (crc_temp << 8) ^ crc_table[index];
i=0;
while ( i < noOfBytes) {
    if (selectMemIn==0) data = plainText[i];
    else if (selectMemIn==1) data = cipherText[i];
    index = ((crc_temp >> 24) ^ data) & 0xFF;
    crc_temp = (crc_temp << 8) ^ crc_table[index];
    i=i+1;
}
wait(1);
crc_temp = crc_temp ^ 0xFFFFFFFF; //EXOR with the Final Word
crcTMP[0] = crc_temp;//32bit Output (Using Shared Memory)
/*SW-END*/
crcOut = crcTMP[0];
doneCrcCalcOut = 1;
```

### Synthesis System Flow



# Extraction of Data Dependencies

- Detect variables that are *read after* but *defined before* "SW-begin"

- SW inputs

- Detect variables that are *written before* but *read after* "SW-end"

- SW outputs

```
doneCrcCalcOut = 0;  
wait(startCrcIn);  
noOfBytes = bufferSizeIn;
```

■ SW Local variables

■ HW -> SW I/F variables

```
/*SW-BEGIN*/  
crc_temp = 0xFFFFFFFF; //Initialize the CRC  
index = (crc_temp >> 24) & 0xFF;  
crc_temp = (crc_temp << 8) ^ crc_table[index];  
i=0;  
while ( i < noOfBytes) ←  
{  
  if (selectMemIn==0)  
    data = plainText[i];  
  else if (selectMemIn==1)  
    data = cipherText[i];  
  index = ((crc_temp >> 24) ^ data) & 0xFF;  
  crc_temp = (crc_temp << 8) ^ crc_table[index];  
  i=i+1;  
}  
wait(1)  
crc_temp = crc_temp ^ 0xFFFFFFFF;  
/*SW-END*/
```

```
crcOut = crc_temp;  
doneCrcCalcOut = 1;
```

# Automatic Manipulation of Hardware Descriptions

```
doneCrcCalcOut = 0;
wait(startCrcIn);
noOfBytes = bufferSizeIn;
/* Passing values to software */
noOfBytesOut = noOfBytes; /* To SW */
selectMemInOut = selectMemIn; /* To SW */
/* Begin Software */
startGetCrcSWOut = 1;
/*SW-BEGIN*/
//(omitted) crc_temp = 0xFFFFFFFF; //Initialize the CRC
//(omitted) index = (crc_temp >> 24) & 0xFF;
//(omitted) crc_temp = (crc_temp << 8) ^ crc_table[index];
//(omitted) i=0;
---
//(omitted) crc_temp = crc_temp ^ 0xFFFFFFFF; //EXOR with the Final Word
//(omitted) crcTMP[0] = crc_temp;
//(omitted)
/*SW-END*/
wait(doneGetCrcSWIn);
startGetCrcSWOut = 0;
/* End Software */
/* Getting results from software */
crcOut = crc_tempIn; /* From SW */
doneCrcCalcOut = 1;
```

1. Set up Software Inputs

2. Assert SW Start Signal

3. Wait for SW to complete

4. Clear SW start signal

3. Read Result

# Software Code Generation

□ Pointer declaration for memory mapped interface variables

□ Dereference pointers to perform intended arithmetic/logic operations

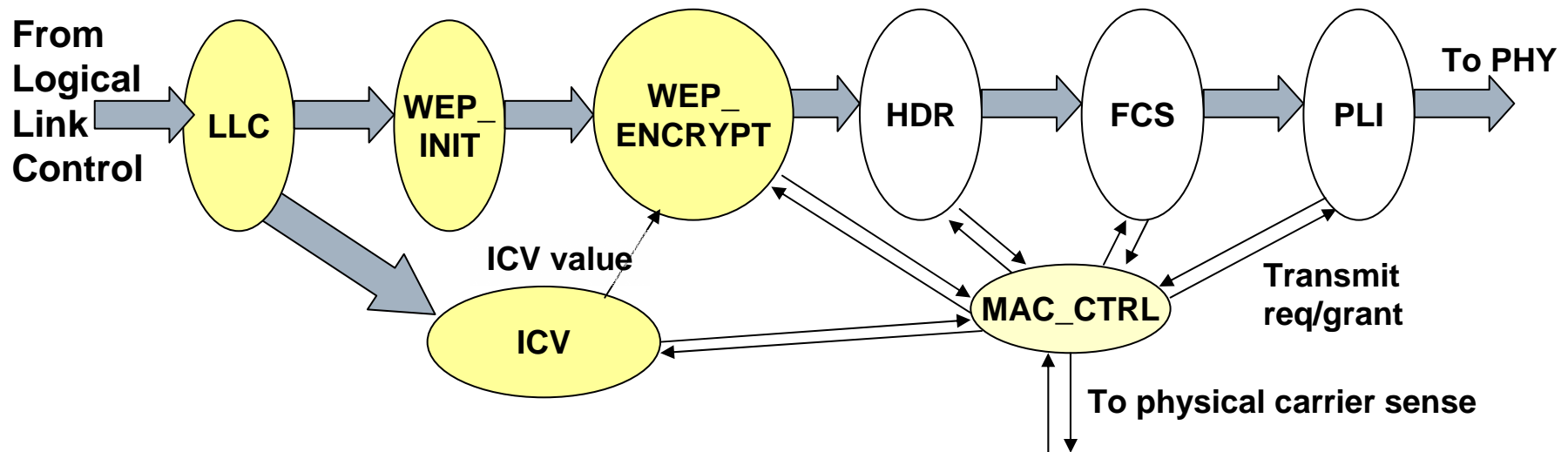
□ Syntax translation of hw-specific constructs

- Remove explicit statements that model hardware behavior

```
#define noOfBytesIn (__ABS_MEMORY+0x8000)
#define crc_temp (__ABS_MEMORY+0x8002)
#define noOfBytes (*v_noOfBytesIn)
#define crc_temp (*v_crc_tempOut)
#define wait(d)
void GetCrcSW(){
    v_noOfBytesIn = (short int *) noOfBytesIn;
    v_selectMemInIn = (short int *) selectMemInIn;
    v_crc_tempOut = (int *) crc_tempOut;
    v_doneGetCrcSWOut = (short int *) doneGetCrcSWOut;
    /*SW-BEGIN*/
    int index = 0; int i = 0; int data = 0; int crc_temp = 0;
    crc_temp = 0xFFFFFFFF; /*Initialize the CRC*/
    ...
    ...
    else if (selectMemIn==1) data = cipherText[i];
    index = ((crc_temp >> 24) ^ data) & 0xFF;
    crc_temp = (crc_temp << 8) ^ crc_table[index];
    i=i+1;
}
    wait(1)
    crc_temp = crc_temp ^ 0xFFFFFFFF;
    * v_doneGetCrcSWOut = ONE;
/*SW-END*/ }
}
```

# Case Study: 802.11 MAC Layer

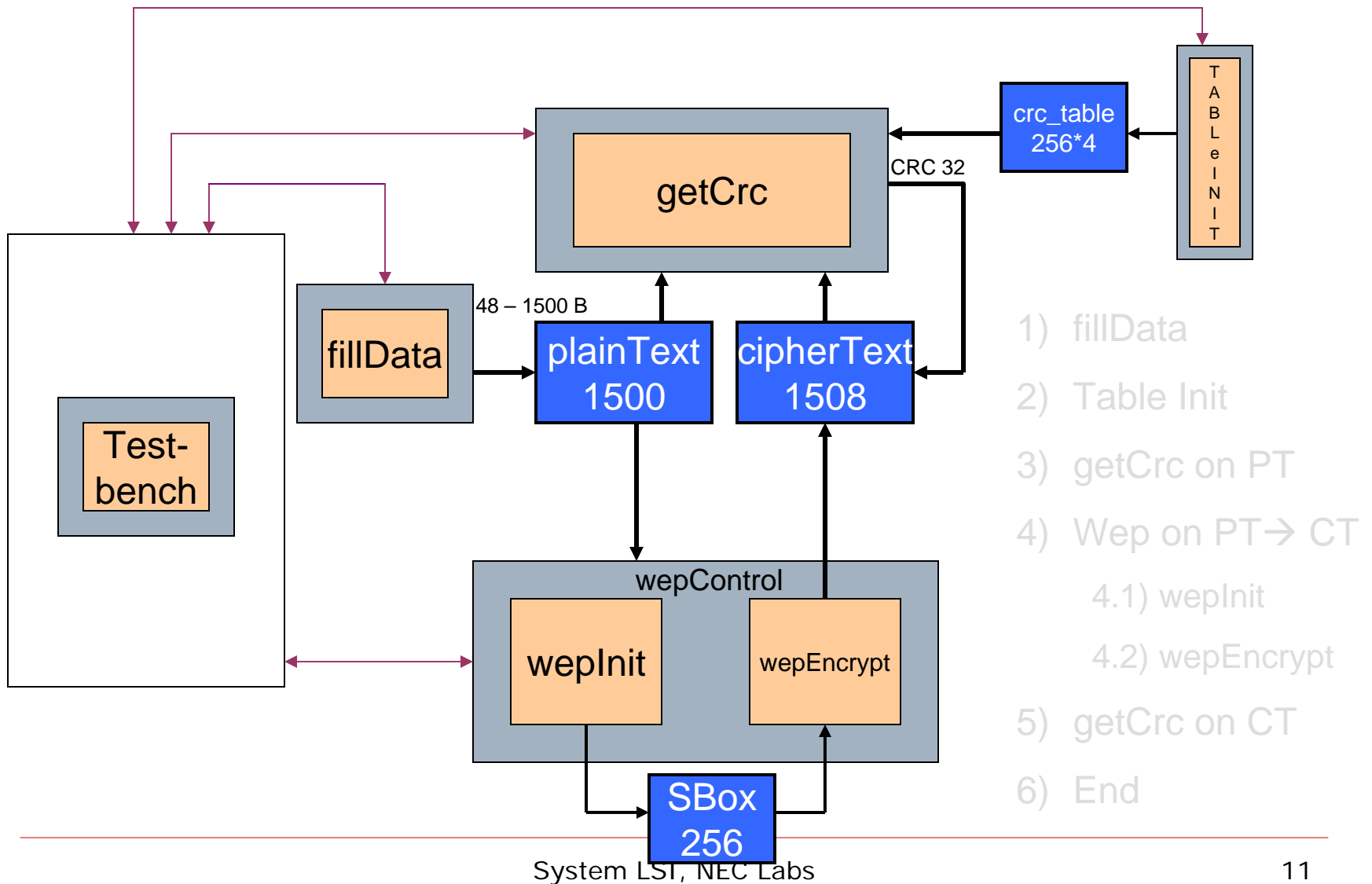
## □ Functional specification



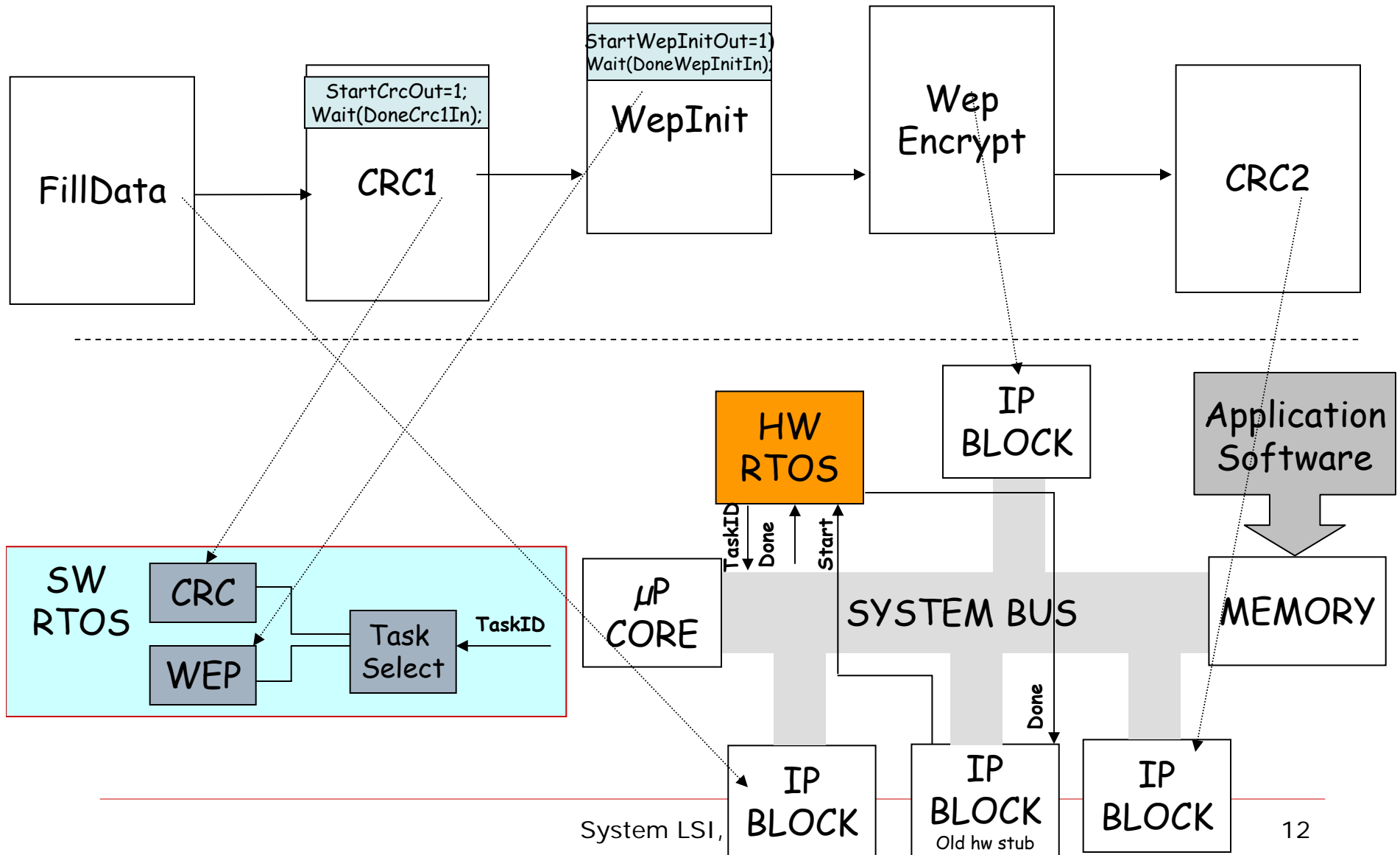
### For each MAC frame:

- LLC: Writes incoming frame to memory
- WEP\_INIT: Initializes a substitution box
- WEP\_ENCRYPT: Performs RC4 encryption
- ICV: Computes CRC-32 checksum over plain text.
- FCS: Computes CRC-32 checksum over cipher text

# 802.11 MAC Sub-system HW Architecture



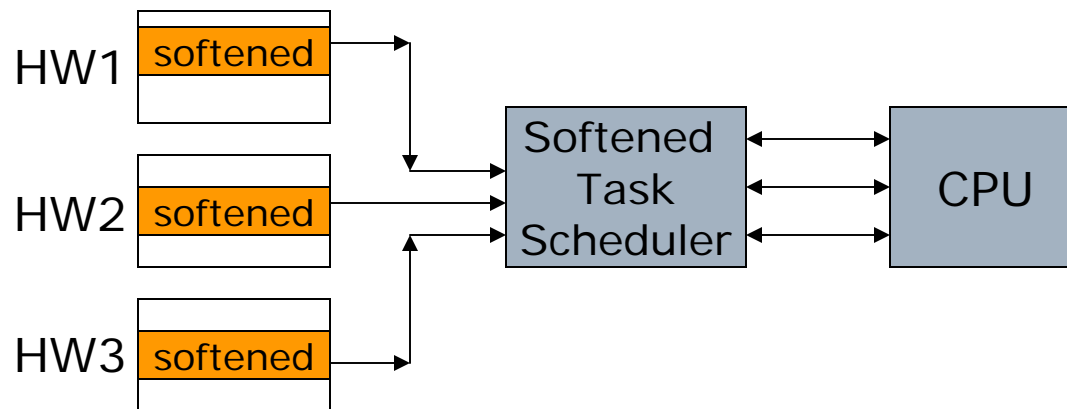
# "Softened" 802.11 MAC Sub-system



# Scheduler for Softened Tasks

---

- ❑ Multiple softened blocks may result in simultaneously pending tasks for the CPU
- ❑ Scheduling support is needed in order to sequentialize the execution of softened tasks

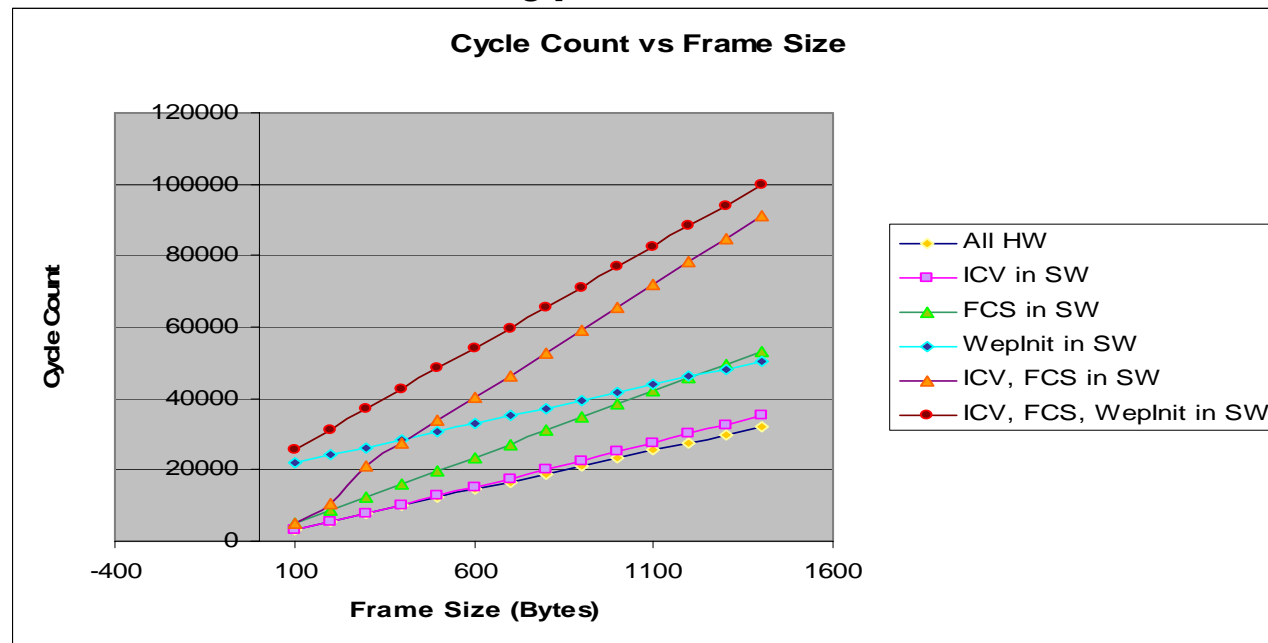


- ❑ Softened task scheduler: HW/SW co-design of a simple RTOS-like entity
  - Implement scheduling algorithm in HW (e.g., round robin, EDF, etc)
  - SW is only responsible for task invocation

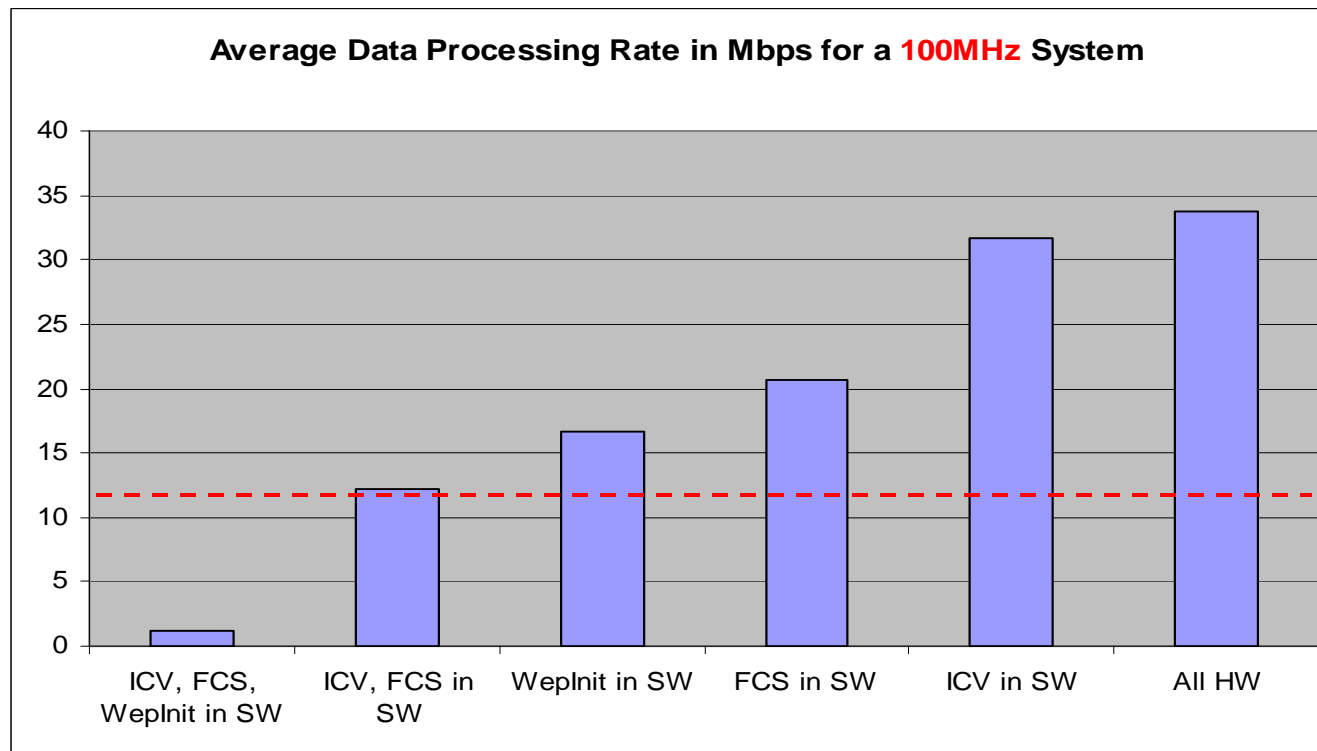
# Preliminary Results

## □ Experimental Methodology:

- Manual selection of candidate blocks in the 802.11 MAC sub-system for softening
- Automatic flow successfully applied for
  - ICV, FCS computation (CRC-32)
  - WEP\_INIT (RC4 encryption)



# Comparison of Different Softening Alternatives



- ❑ Numerous alternatives exist for softening, depending on performance requirements
- ❑ Need to carefully select softened tasks, based on hardware savings, and performance impact

# Conclusion

---

## □ Ongoing work

- Develop systematic techniques for selecting blocks to soften

## □ Current Status

- Data dependences extraction
- Interface Synthesis
- Performed a case study with the 802.11 MAC subsystem