Towards Multi-Objective Dynamic SPM Allocation

Shashank Jadhav  Heiko Falk

shashank.jadhav / heiko.falk@tuhh.de

Institute of Embedded Systems, TUHH
21st International Workshop on Worst-Case Execution Time Analysis (WCET), Vienna, Austria

July 11, 2023
Overview

1 Motivation

2 Dynamic SPM Allocation (DSA)

3 Multi-Objective DSA-based Optimization

4 Evaluation

5 Conclusion
Motivation

- Worst-Case Execution Time (WCET)
- Energy consumption
- Static SPM allocation constrained by small SPM size
Overview

1. Motivation

2. Dynamic SPM Allocation (DSA)

3. Multi-Objective DSA-based Optimization

4. Evaluation

5. Conclusion
Dynamic SPM Allocation (DSA)

1. Determine Memory Objects
2. Perform Liveness Analysis
3. Allocation of Memory Objects
4. Solve Address Assignment Problem
5. Code Transformation for DMA
6. Final Code Generation and WCET/Energy Analyses
DSA: Memory Objects

- Determine Memory Objects
- Perform Liveness Analysis
- Allocation of Memory Objects
- Code Transformation for DMA
- Solve Address Assignment Problem
- Final Code Generation and WCET/Energy Analyses

**Flash**

- main()
  - MO_3
  - Loop_1
- call Foo_1(...)  
- MO_4
  - Loop_2

**SPM**

- Empty

**Memory Objects:**
- Functions: Foo_1() and Foo_2()
- Loops: Loop_1, Loop_2, Loop_3, and Loop_4

© Shashank Jadhav, Heiko Falk
DSA: Liveness Analysis

Memory Objects:
- Functions: Foo_1() and Foo_2()
- Loops: Loop_1, Loop_2, Loop_3, and Loop_4

Liveness conflicts:
- Foo_2(), Loop_3, and Loop_4
- Foo_1() and Foo_2()
DSA: Memory Objects Allocation

- **Determine Memory Objects**
- **Perform Liveness Analysis**
- **Allocation of Memory Objects**
- **Code Transformation for DMA**
- **Solve Address Assignment Problem**
- **Final Code Generation and WCET/Energy Analyses**

**Memory Objects:**
- **Functions:** Foo_1() and Foo_2()
- **Loops:** Loop_1, Loop_2, Loop_3, and Loop_4

**Liveness conflicts:**
- Foo_2(), Loop_3, and Loop_4
- Foo_1() and Foo_2()

**Memory Object Allocation:**
- **Flash:** Foo_1() and Loop_3()
- **SPM:** Foo_2(), Loop_1, Loop_2, and Loop_4

---

© Shashank Jadhav

Heiko Falk
DSA: Address Assignment

Memory Objects:
- Functions: Foo_1() and Foo_2()
- Loops: Loop_1, Loop_2, Loop_3, and Loop_4

Liveness conflicts:
- Foo_2(), Loop_3, and Loop_4
- Foo_1() and Foo_2()

Memory Object Allocation:
- Flash: Foo_1() and Loop_3()
- SPM: Foo_2(), Loop_1, Loop_2, and Loop_4

Address Assignment:
- First-Fit Heuristic
- Best-Fit Heuristic
At Compile Time

- Determine Memory Objects
- Perform Liveness Analysis
- Allocation of Memory Objects
- Code Transformation for DMA
- Solve Address Assignment Problem

Final Code Generation and WCET/Energy Analyses

B0
FLASH

B1
FLASH

B2
FLASH

B3
FLASH

B4
FLASH

B5
FLASH

Memory Object (MO)
DSA: Code Transformation

Determine Memory Objects → Perform Liveness Analysis → Allocation of Memory Objects → Code Transformation for DMA → Solve Address Assignment Problem → Final Code Generation and WCET/Energy Analyses

At Compile Time

- B0
  - FLASH
    - B0_call Insert Memcpy/Call
    - B0_jump Fix Jumps

- B1
  - FLASH
  - B1
    - FLASH
      - B2
        - FLASH
        - B2
          - FLASH
          - B4
            - FLASH
            - B4_jump Fix Jumps
            - B4
              - FLASH
              - B5
                - FLASH
                - B5
DSA: Code Transformation

- Determine Memory Objects
- Perform Liveness Analysis
- Allocation of Memory Objects
- Code Transformation for DMA
- Solve Address Assignment Problem
- Final Code Generation and WCET/Energy Analyses

**At Compile Time**

- **B0**
  - FLASH
  - B0_call
  - Insert Memcpy Call
  - B0_jump
  - Fix Jumps

- **B1**
  - FLASH
  - Memory Object (MO)

- **B2**
  - FLASH

- **B3**
  - FLASH

- **B4**
  - FLASH
  - B4_jump
  - Fix Jumps

- **B5**
  - FLASH

**During Runtime**

- **B0**
  - FLASH
  - B0_call
  - Call Memcpy & Copy MO to SPM
  - B0_jump
  - Jump from B0_call in Flash to B1 in SPM

- **B1**
  - SPM
  - Memory Object (MO)

- **B2**
  - SPM

- **B3**
  - SPM

- **B4**
  - SPM
  - B4_jump
  - Jump from B4 in SPM to B5 in Flash

- **B5**
  - FLASH
DSA: Code Generation and Analyses

1. Determine Memory Objects
2. Perform Liveness Analysis
3. Allocation of Memory Objects
4. Solve Address Assignment Problem
5. Code Transformation for DMA
6. Fix Literal Pool Placements
7. Temporarily Resize SPM
8. Perform WCET and Energy Analyses
10. Statically Allocate Code
11. Insert Temporary NOP Basic Blocks Within Main Memory
12. Final Code Generation and WCET/Energy Analyses
Overview

1 Motivation

2 Dynamic SPM Allocation (DSA)

3 Multi-Objective DSA-based Optimization

4 Evaluation

5 Conclusion
Multi-Objective Optimization Problem

- **Search Space:**
  
  \[ x \in X \in \{0, 1\}^d \]
Multi-Objective Optimization Problem

- **Search Space:**
  \[ x \in X \in \{0, 1\}^d \]

- **Objective Space:**
  \[ \Theta = \{ F(x) = (F_1(x), F_2(x)) | x \in X \} \]
  Where, \( F_1(x) = \) WCET objective and \( F_2(x) = \) Energy objective
Multi-Objective Optimization Problem

- **Search Space:**
  \[ x \in X \in \{0, 1\}^d \]

- **Objective Space:**
  \[ \Theta = \{F(x) = (F_1(x), F_2(x)) | x \in X\} \]
  Where, \( F_1(x) = \) WCET objective and \( F_2(x) = \) Energy objective

- **Minimization function:**
  \[ \min_{x \in X} F(x) \]
Multi-Objective Optimization Problem

- **Search Space:**
  
  \[ x \in X \subseteq \{0, 1\}^d \]

- **Objective Space:**
  
  \[ \Theta = \{ F(x) = (F_1(x), F_2(x)) \mid x \in X \} \]
  
  Where, \( F_1(x) \) = WCET objective and \( F_2(x) \) = Energy objective

- **Minimization function:**
  
  \[ \min_{x \in X} F(x) \]

- **Search Space Constraint:**
  
  \[ x_{(F+1):(F+L)} = x_{(F+1):(F+L)} + \tau \]
  
  Where,

  \[ \tau_l = \begin{cases} 
  1, & \text{if } x_{F+1} = 0 \& (\exists f \mid \lambda_{F+1} \subseteq \lambda_f \in \Lambda_{1:F}) \& x_f = 1 \\
  0, & \text{otherwise}
  \end{cases} \]
Multi-Objective Optimization Problem

- **Search Space:**
  \* \( x \in X \in \{0, 1\}^d \)

- **Objective Space:**
  \* \( \Theta = \{ F(x) = (F_1(x), F_2(x)) | x \in X \} \)
  Where, \( F_1(x) \) = WCET objective and \( F_2(x) \) = Energy objective

- **Minimization function:**
  \* \( \min_{x \in X} F(x) \)

- **Search Space Constraint:**
  \* \( x_{(F+1):(F+L)} = x_{(F+1):(F+L)} + \tau \)
  Where,
  \( \tau_l = \begin{cases} 
  1, & \text{if } x_{F+l} = 0 \text{ and } (\exists f | \lambda_{F+l} \subseteq \lambda_f \in \Lambda_{1:F}) \text{ and } x_f = 1 \\
  0, & \text{otherwise} 
\end{cases} \)

- **Address Assignment Algorithm Constraint:**
  \* \( (\mathcal{T} - \eta) = 0 \)
Overview

1. Motivation
2. Dynamic SPM Allocation (DSA)
3. Multi-Objective DSA-based Optimization
   - Multi-Objective Optimization Problem
4. Evaluation
5. Conclusion
To solve multi-objective DSA-based optimization problem, we use:

- Flower Pollination Algorithm (FPA)
- Strength Pareto Evolutionary Algorithm (SPEA)

**Algorithm** Multi-Objective DSA-based optimization

1: Collect \textit{memObj}, perform Liveness Analysis, and randomly initialize initial population of size $N$
2: \textbf{for} $n = 1 : N$ \textbf{do}
3: \hspace{1em} Generate DSA code
4: \textbf{while} Stopping criteria is not reached \textbf{do}
5: \hspace{1em} Update Individual using respective update operators
6: \hspace{1em} \textbf{for} Each updated Individual \textbf{do}
7: \hspace{2em} Generate DSA code
8: \hspace{1em} Update to next generation using selection operator
9: \hspace{1em} \textbf{return} Pareto-optimal solution set
Overview

1. Motivation

2. Dynamic SPM Allocation (DSA)

3. Multi-Objective DSA-based Optimization

4. Evaluation

5. Conclusion
Comparison between...

- Proposed multi-objective DSA-based optimization \((M_{OD})\)  
  \(\rightarrow\) Solved using:
  - FPA
  - SPEA

- Multi-objective static SPM allocation-based optimization \((M_{OS})\)  
  \(\rightarrow\) Solved using:
  - FPA
  - SPEA

- ILP-based single objective dynamic SPM allocation \((SO_{D})\)
Overview

1. Motivation
2. Dynamic SPM Allocation (DSA)
3. Multi-Objective DSA-based Optimization
4. Evaluation
   - Pareto fronts
5. Conclusion
Figure 1: Solutions Obtained from $MO_S$, $MO_D$, and $SO_D$ optimization runs
Pareto fronts

The following percent of solutions were on the final Pareto front

- $MO_S$–FPA: 3.62%
- $MO_S$–SPEA: 5.26%
- $SO_D$–ILP: 0.66%
- $MO_D$–FPA: 70.4%
- $MO_D$–SPEA: 20.1%

$MO_D$–FPA found most number of solution on the final Pareto front
Overview

1. Motivation
2. Dynamic SPM Allocation (DSA)
3. Multi-Objective DSA-based Optimization
4. Evaluation
   - Quality Indicators
5. Conclusion
Quality Indicators

- Coverage: \( C = 1 - \frac{|\{a \in A : \exists p \in P, a < p\}|}{|A|} \)
- Non-Dominance Ratio: \( NDR = \frac{|P \cap A|}{|P|} \)
- Non-Dominated Solutions: \( NDS = \frac{|a \in A : a \in P|}{|A|} \)
Quality Indicators

- **Coverage**: $C = 1 - \frac{|\{a \in A: \exists p \in P, a \prec p\}|}{|A|}$

- **Non-Dominance Ratio**: $NDR = \frac{|P \cap A|}{|P|}$

- **Non-Dominated Solutions**: $NDS = \frac{|a \in A: a \in P|}{|A|}$

From overall Evaluations, in terms of Quality Indicators:

- $MO_D$ performed much better than $SO_D$
- $MO_D$ performed slightly better than $MO_S$
Quality Indicators

- Coverage: $C = 1 - \frac{|\{a \in A : \exists p \in P, a \preceq p\}|}{|A|}$
- Non-Dominance Ratio: $NDR = \frac{|P \cap A|}{|P|}$
- Non-Dominated Solutions: $NDS = \frac{|a \in A : a \in P|}{|A|}$

From overall Evaluations, in terms of Quality Indicators:
- $MO_D$ performed much better than $SO_D$
- $MO_D$ performed slightly better than $MO_S$

Overheads due to dynamic copying in $MO_D$ optimization run:
- WCET overheads on average: 24.39%
- Energy overheads on average: 22.65%
Overview

1. Motivation
2. Dynamic SPM Allocation (DSA)
3. Multi-Objective DSA-based Optimization
4. Evaluation
5. Conclusion
Conclusion

- Proposed compiler-level DSA-based multi-objective optimization
- WCC performs WCET and energy analysis of DSA code
- $MO_D$ is solved using FPA and SPEA
- $MO_D$ outperforms $SO_D$
- $MO_D$ performs slightly better than $MO_S$

Future Work

- Reducing the WCET and energy overheads by using DMA
- Reducing the compilation time needed by multi-objective DSA-based optimization
Thank You