Short-Circuit Compiler Transformation: Optimizing Conditional Blocks

Mohammad Ali Ghodrat, Tony Givargis, Alex Nicolau

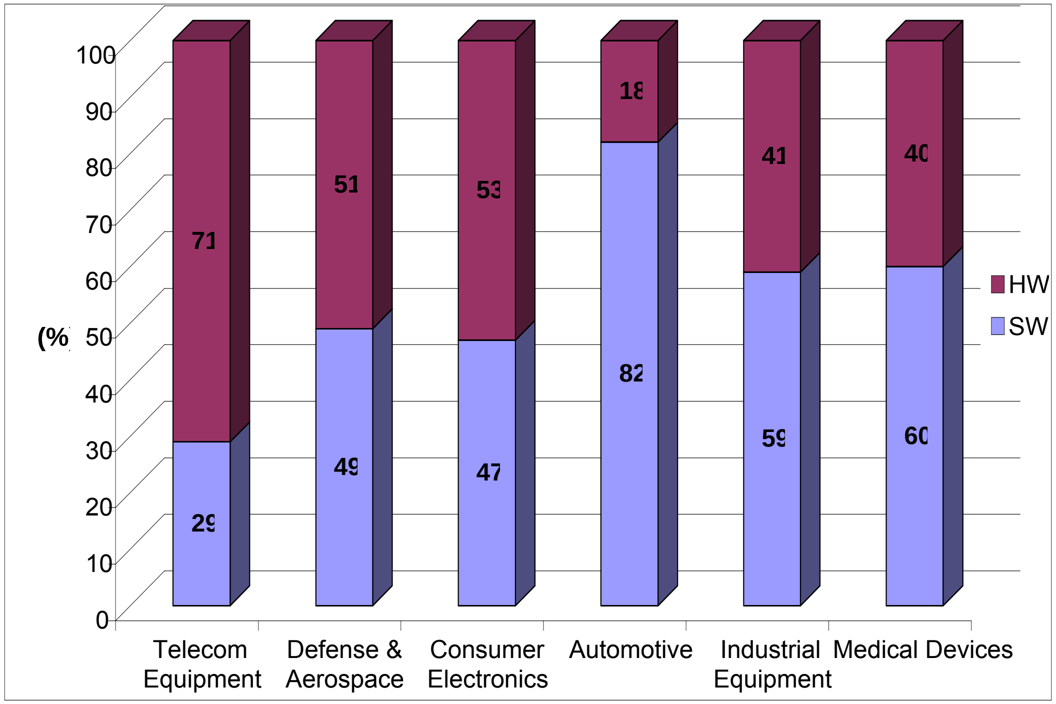
Department of Computer Sciences
Center for Embedded Computing System
University of California, Irvine

Outline

- ❑ Introduction and motivational example
- ❑ Related work and underlying analysis
- ❑ Proposed transformation
- ❑ Measuring/Optimizing delay
- ❑ Results
- ❑ Conclusion

Embedded Software

- Growing importance of embedded software
- Importance of aggressive compiler optimizations for embedded systems
- Long time compilation for embedded software is tolerable



Source: The Boston Consulting Group & Thomson Financial Database, 2005

Motivational example

Original

Transformed

```
for(x=min; x<max; x++) {
 for(y=min; y<max; y++) {
  if( x*x+y*y == x*x*y )
   color(x, y, BLACK);
 }
}
```

```
// expr is false for y<0
for(x=min; x<max; x++) {
 for(y=min; y<max; y++) {
  if( y < 0 )
   continue;
  else if(x*x+y*y == x*x*y )
   color(x, y, BLACK);
 }
}
```

Bypass evaluation of expressions when possible

Related work

- ❑ Lazy evaluation of Boolean expressions
  - ➲ Early works
    - →H.D. Huskey et. al [1961]
    - →B.W. Arden et. al [1962]
  - ➲ Ordering Boolean operands
    - →M.Z. Hanani [1977]
  - ➲ Effect on code size
    - →M.H. Clifton [1998]
- ❑ Conditional expression evaluation
  - ➲ CASES-2005, TVLSI-2006

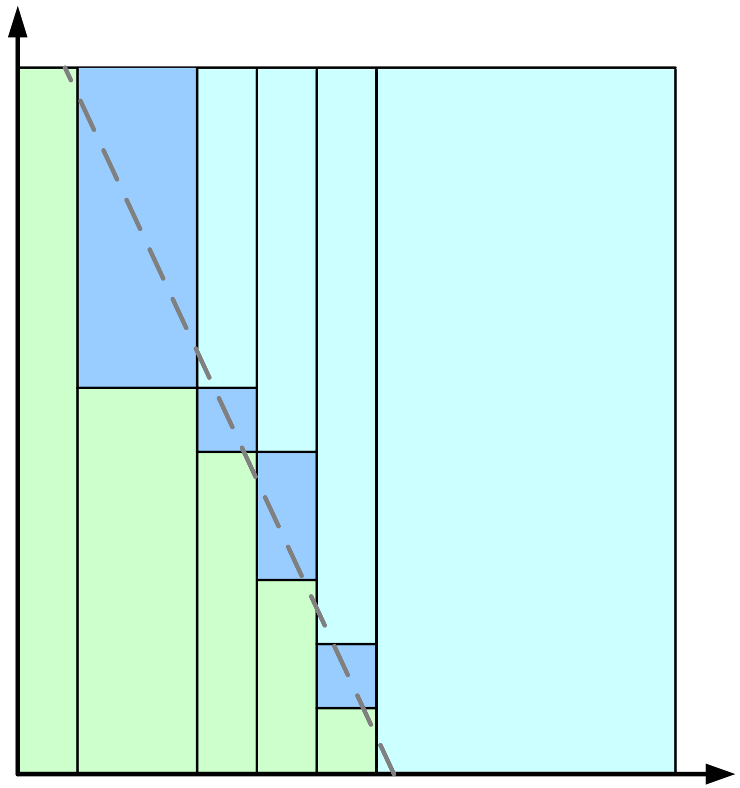
Underlying Analysis

- Given a conditional expression:
  - E.g., $2x_0+x_1+4 > 0$
- n-dimensional space S is a box-shaped region:

$[L_0,U_0] \times [L_1,U_1] \times \ldots \times [L_{n-1},U_{n-1}]$

- Compute, off-line: true, false, & unknown *spaces*
- Domain space partitioning problem solution is given in figure.
- Membership test: if a point X $(x_0,x_1,\ldots,x_{n-1})$ is in a space S:

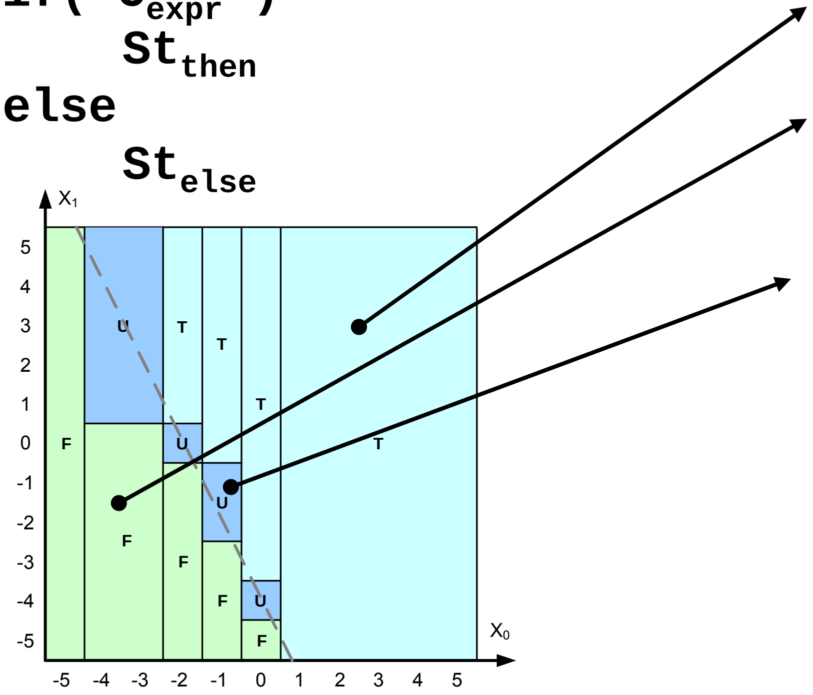
$(L_0 \le x_0 \le U_0)$ && $(L_1 \le x_1 \le U_1)$ && … $(L_{n-1} \le x_{n-1} \le U_{n-1})$



Transformation overview

Original

if(C_{expr})
  St_{then}
else
  St_{else}



Transformed

if(x_0 > 0 && x_0 < 5)
  St_{then}
else if(…)
  St_{else}
…
else {
  if(C_{expr})
    St_{then}
  else
    St_{else}
}

Transformation template

```
if (C     )
    St
else
    St
```

```
if (X∈S )
    St
else if (X∈S  )
    St
    ⋮
else if (X∈S  )
    St
else {



}
```

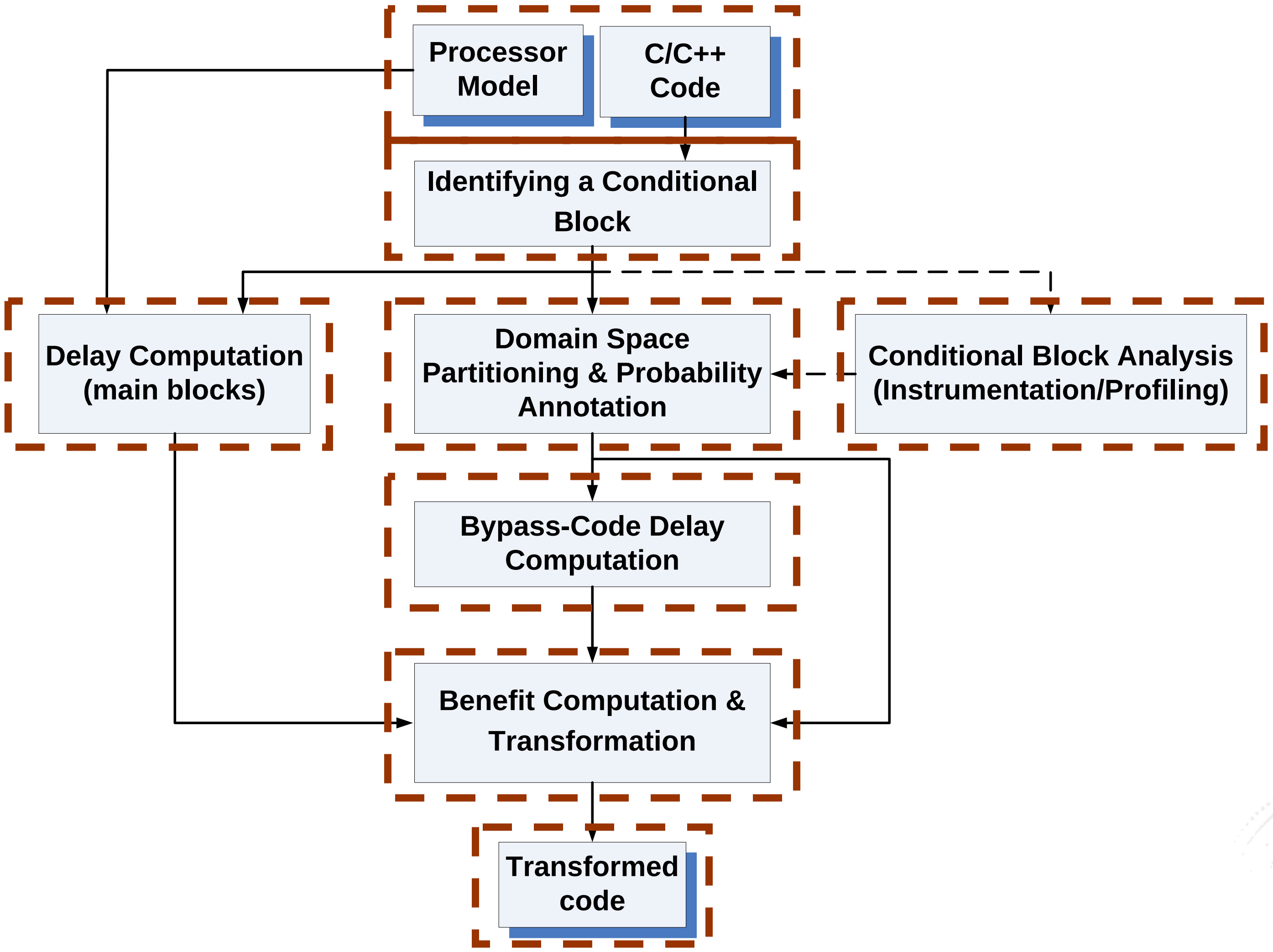
$$St \quad = \begin{cases} St & ; BVi = true \\ St & ; BVi = false \end{cases}$$

Original T

Optimization Strategy

- Pick any conditional block
  - Analyze it
  - Estimate cost of bypasses
  - Estimate cost of conditional block
  - Make decision and mark
  - Repeat until all visited
- Use profiling to improve the estimates above

Overall flow


Processor Model
C/C++ Code
Identifying a Conditional Block
Delay Computation (main blocks)
Domain Space Partitioning & Probability Annotation
Conditional Block Analysis (Instrumentation/Profiling)
Bypass-Code Delay Computation
Benefit Computation & Transformation
Transformed code

Example – MP3

```
for(j=0;j<CBANDS;j++){
        for(i=0;i<CBANDS;i++){
                t1[i][j] += 0.474;
                t3 = 15.811389+7.5*t1[i][j]-17.5*sqrt((double) (1.0+t1[i][j]*t1[i][j]));
                if(t3 <= -100) {
                        s[i][j] = 0;
                }
                else {
                        t3 = (t2[i][j] + t3)*LN_TO_LOG10;
                        s[i][j] = exp(t3);
                }
        }
}
```

* Output of the code segment is s[CBANDS][CBANDS]

Example (Conditional Block Analysis)

C : $15.8 + 7.5 * (t1[i][j] + 0.474) - 17.5 * \sqrt{1 + (t1[i][j] + 0.474)^2} \le -100$

Space	Boolean Value(BV)	Probability
[-4.7, 11.8]	False	0.475
[-30, -4.7]	True	0.312
[11.8, 30]	True	0.160

expr

Example (transformation)

[-4.7,11.862]	False
[-30,-4.702]	True
[11.864,30]	True

```
else
if ( t1[i][j]<= 11.862 && t1[i][j]>= -4.7)
{       t1[i][j] += 0.474;
        t3 = -15.811389+7.5*t1[i][j]-17.5*sqrt((double) (1.0+t1[i][j]*t1[i][j]));
else if ( t1[i][j]>=11.864 && t1[i][j]<=30 )
        t3 = -15.811389+7.5*t1[i][j]-17.5*sqrt((double) (1.0+t1[i][j]*t1[i][j]));
        t3 = (t2[i][j] + t3)*LN_TO_LOG10;
        s[i][j] = exp(t3);
}
```

```
{
        t1[i][j] += 0.474;
        if(t3 < -16.8100)
        s[i][j] = 0;
        else {
                t3 = (t2[i][j] + t3)*LN_TO_LOG10;
                s[i][j] = exp(t3);
        }
}
```

Example (Transformed code)

```
for(j=0;j<CBANDS;j++){
        for(i=0;i<CBANDS;i++){
                if ( t1[i][j]<= 11.862 && t1[i][j]>= -4.7)
                {
                        t1[i][j] += 0.474;
                        t3 = 15.811389+7.5*t1[i][j]-17.5*sqrt((double) (1.0+t1[i][j]*t1[i][j]));
                        t3 = (t2[i][j] + t3)*LN_TO_LOG10;
                        s[i][j] = exp(t3);
                }
                else if ( t1[i][j]<= -4.702 && t1[i][j]>= -30 )
                        s[i][j] = 0;
                else if ( t1[i][j]>=11.864 && t1[i][j]<=30 )
                        s[i][j] = 0;
                else
                {
                        t1[i][j] += 0.474;
                        t3 = 15.811389+7.5*t1[i][j]-17.5*sqrt((double) (1.0+t1[i][j]*t1[i][j]));
                        if(t3 <= -100)
                                s[i][j] = 0;
                        else {
                                t3 = (t2[i][j] + t3)*LN_TO_LOG10;
                                s[i][j] = exp(t3);
                        }
                }
        }
}
```

Benefit of transformation (delay)

```
if (C    )
    St
else
    St
```

```
if (X∈S )
    St
else if (X∈S )
    St
    ⋮
else if (X∈S  )
    St
else {

}
```

$$T_{new} < T_{original}$$

$$T_{original} = C_{expr}.delay$$

$$+ Prob_{then} \times S_{then}.delay$$

$$+ Prob_{else} \times S_{else}.delay$$

Original

Transf

Benefit of transformation (delay)

```
if (C    )
    St
else
    St
```

```
if (X∈S )
    St
else if (X∈S )
    St
  ⋮
else if (X∈S )
    St
else {

}
```

T_{new} = P_1 * ($case_1$.delay + S_{BV1}.delay)

+ P_2 * ($case_1$.delay+$case_2$.delay+S_{BV2}.delay)

+ …

+ P_m*($case_1$.delay+…+$case_m$.delay+S_{BVm}.delay)

+ (1-P_1-…-P_m) * $T_{original}$

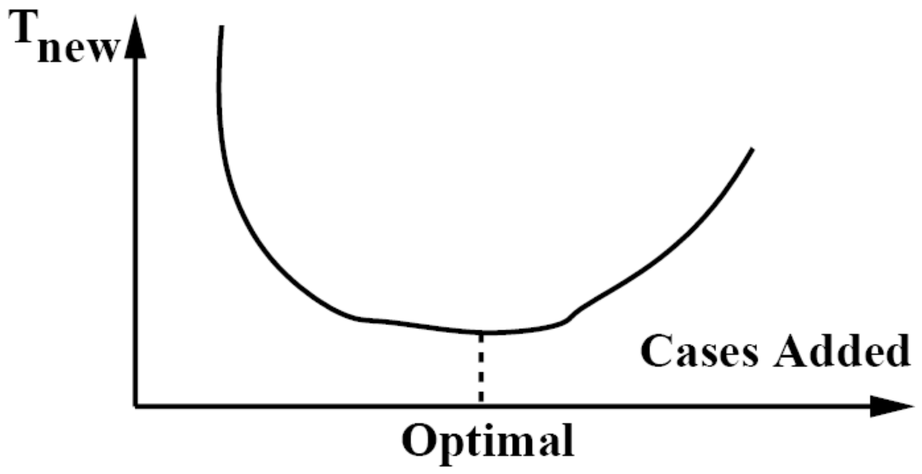
Original

Trans

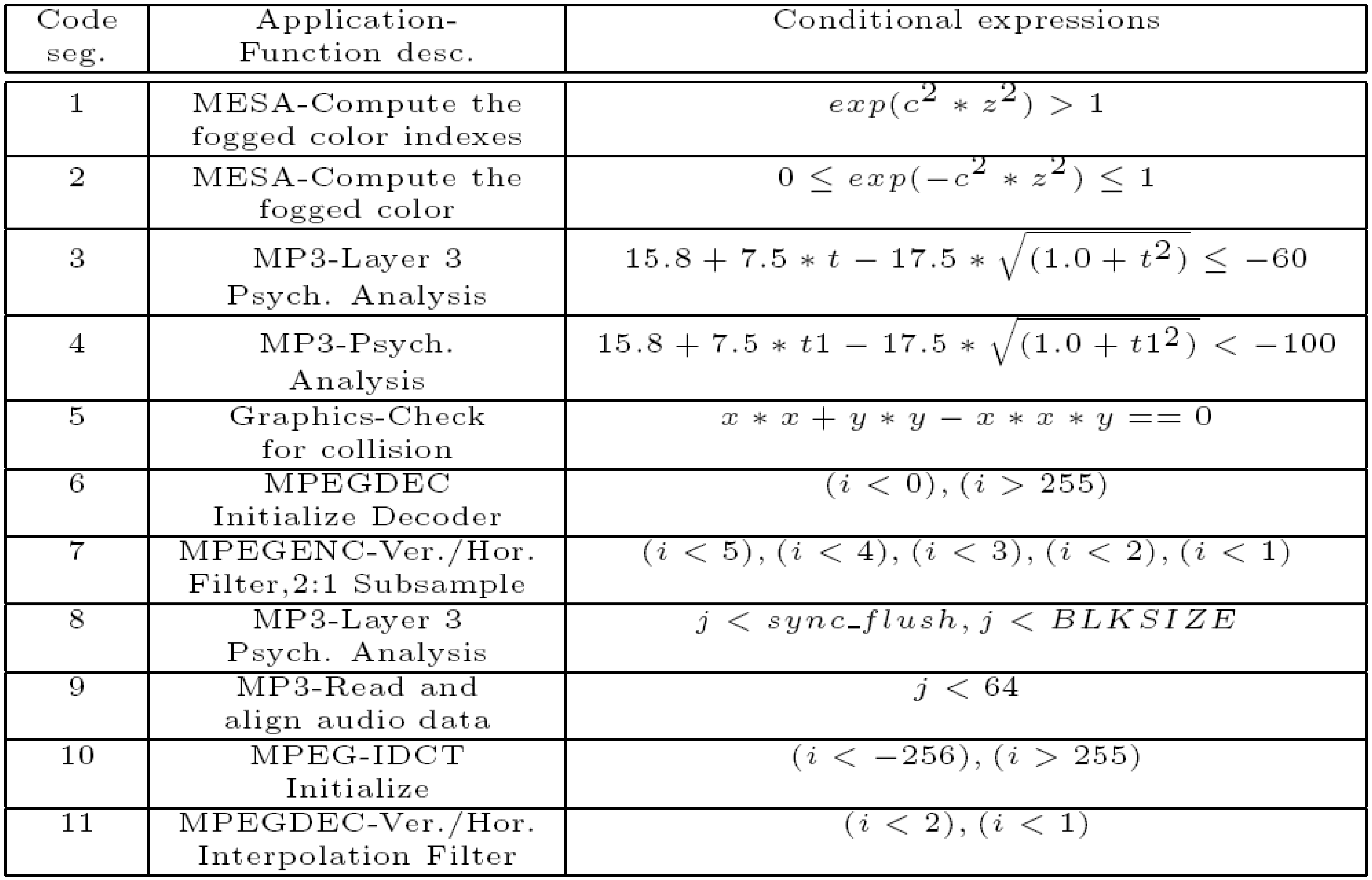
expr

Optimizing if-else chain

- Any or all of the cases may be left out because of the last else block
- As the number of cases that are added grows, T_{new} increases
- Add the next highest probability case as long as T_{new} is strictly decreasing.

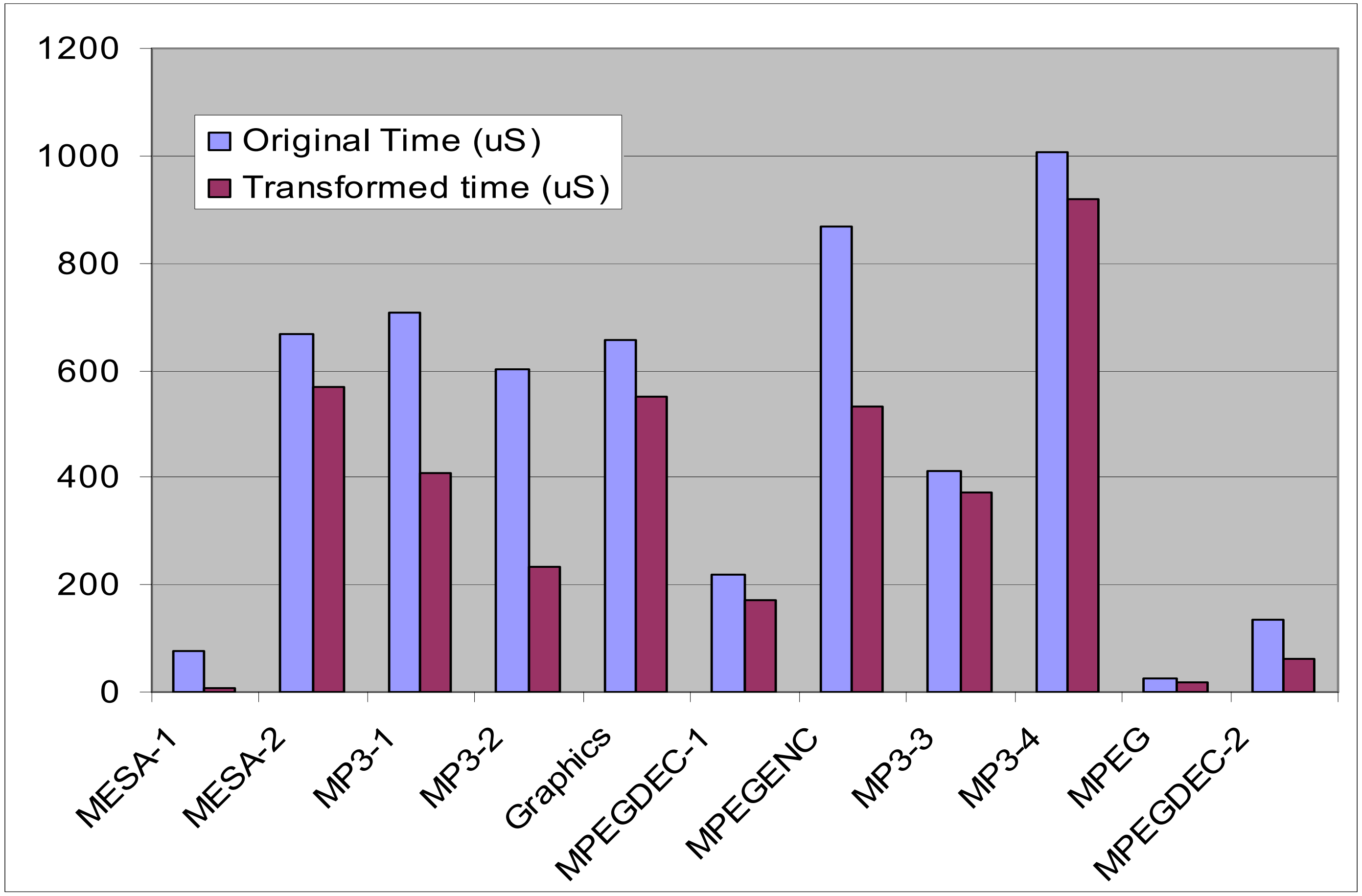


Experimental Results



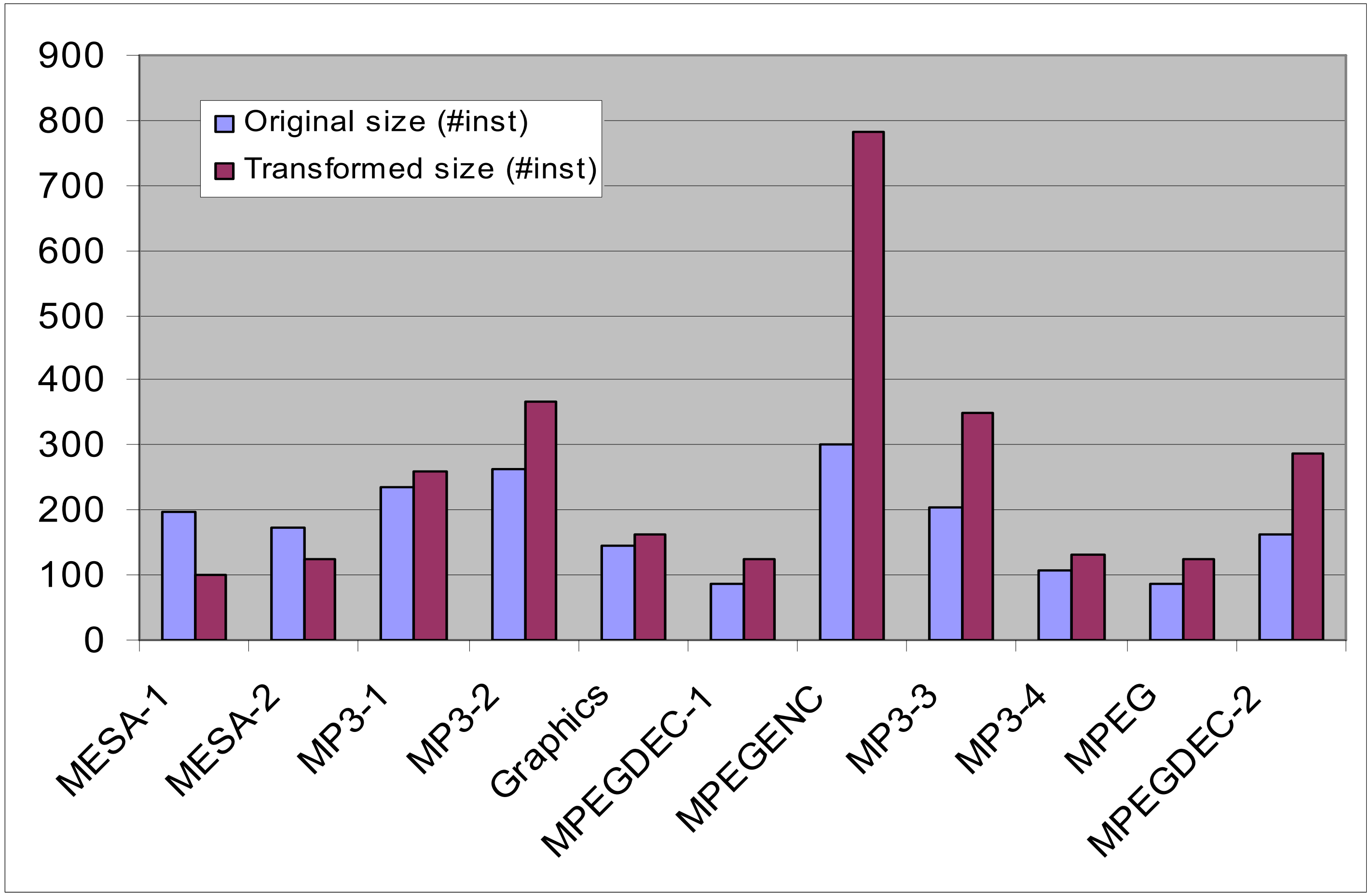
Code seg.	Application-Function desc.	Conditional expressions
1	MESA-Compute the fogged color indexes	$exp(c^2 * z^2) > 1$
2	MESA-Compute the fogged color	$0 \leq exp(-c^2 * z^2) \leq 1$
3	MP3-Layer 3 Psych. Analysis	$15.8 + 7.5 * t - 17.5 * \sqrt{(1.0 + t^2)} \leq -60$
4	MP3-Psych. Analysis	$15.8 + 7.5 * t1 - 17.5 * \sqrt{(1.0 + t1^2)} < -100$
5	Graphics-Check for collision	$x * x + y * y - x * x * y == 0$
6	MPEGDEC Initialize Decoder	$(i < 0), (i > 255)$
7	MPEGENC-Ver./Hor. Filter,2:1 Subsample	$(i < 5), (i < 4), (i < 3), (i < 2), (i < 1)$
8	MP3-Layer 3 Psych. Analysis	$j < sync\_flush, j < BLKSIZE$
9	MP3-Read and align audio data	$j < 64$
10	MPEG-IDCT Initialize	$(i < -256), (i > 255)$
11	MPEGDEC-Ver./Hor. Interpolation Filter	$(i < 2), (i < 1)$

SPARC – Execution time gain



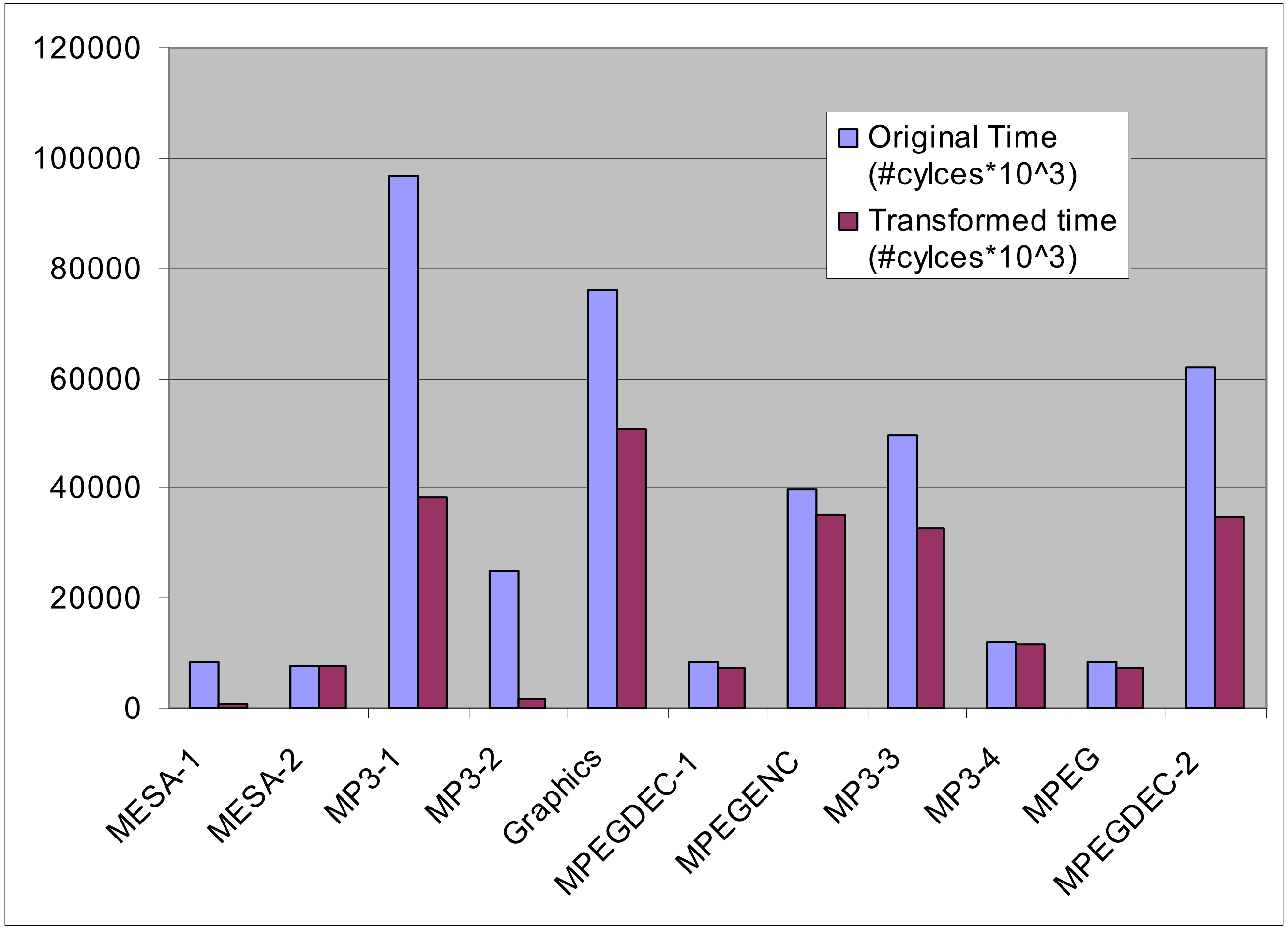
Max: 88.1 %
Min: 8.7 %
Avg: 35.1 %

SPARC – Code size increase



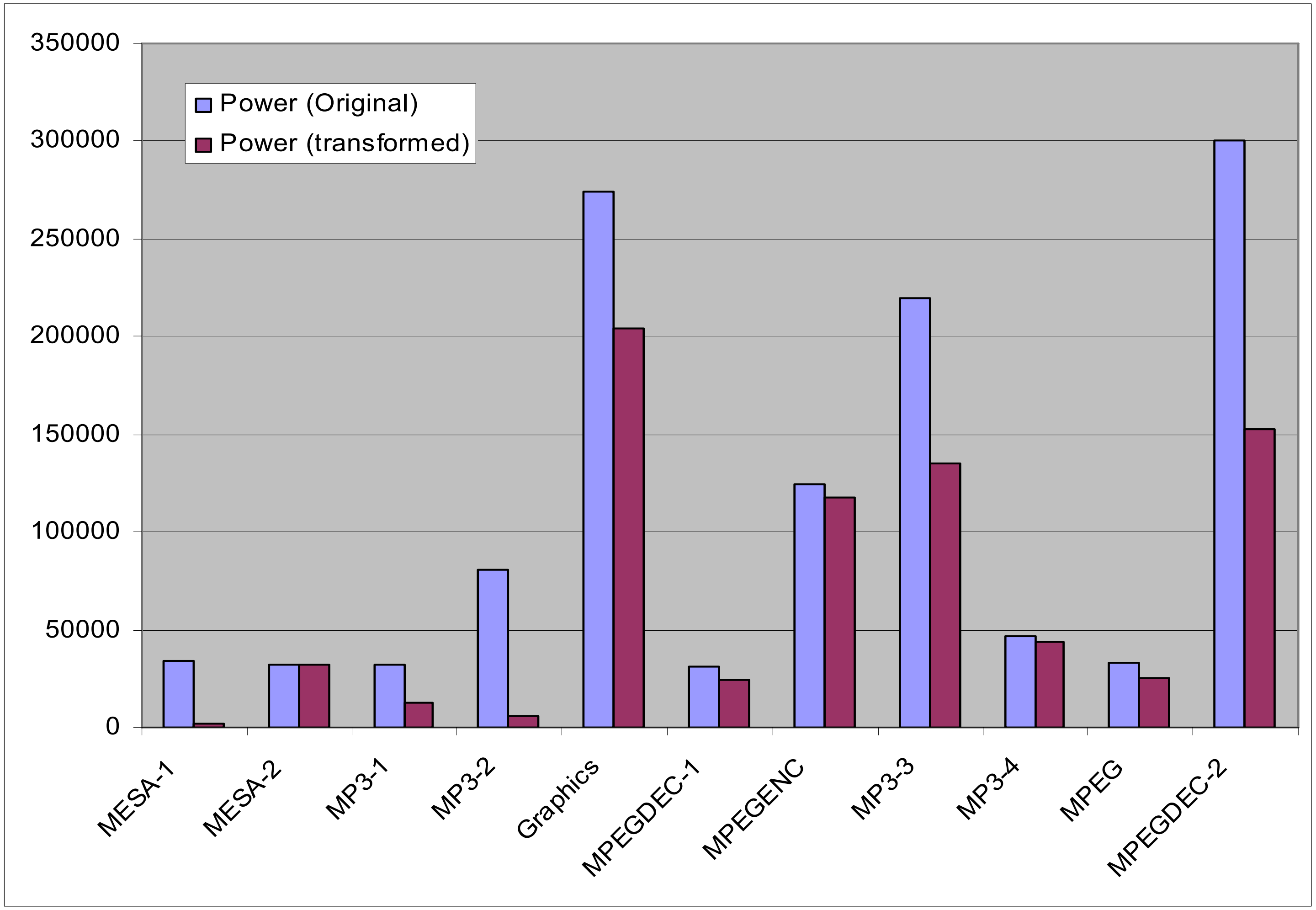
Max: 158 %
Min: -48 %
Avg: 36 %

ARM - Execution time gain



Max: 93 %
Min: 1 %
Avg: 36 %

ARM – Power reduction



Max: 93 %
Min: 0 %
Avg: 36 %

Conclusion

- Short-circuit code transformation technique
- Conditional expression can be statically analyzed
- Bypass the conditional expression evaluation.
- Profile-based optimizations
- Optimization applies to some regions of the code
- For an embedded software this increase in compile time is justified

THANKS